Project no. IST-033576

XtreemOS

Integrated Project
BUILDING AND PROMOTING A LINUX-BASED OPERATING SYSTEM TO SUPPORT VIRTUAL ORGANIZATIONS FOR NEXT GENERATION GRIDS

Evaluation Report and Revision of Application Requirements D4.2.4

Due date of deliverable: November 30^{th}, 2007
Actual submission date: January 11, 2008

Start date of project: June 1^{st} 2006

Type: Deliverable
WP number: WP4.2
Task number: T4.2.2, T4.2.4 and T4.2.5

Responsible institution: SAP
Editor & and editor's address: Bernd Scheuermann
SAP Research, CEC Karlsruhe
Vincenz-Prießnitz-Str. 1
76131 Karlsruhe
Germany

Version 5.0 / Last edited by Bernd Scheuermann / January 11^{th}, 2008

Project co-funded by the European Commission within the Sixth Framework Programme		
Dissemination Level		
PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision history:

Version	Date	Authors	Institution	Section affected, comments
1.0	09/10/07	Bernd Scheuermann	SAP	Initial template
2.0	07/11/07	Bernd Scheuermann	SAP	Included appendix and requirements updates
3.0	20/11/07	WP4.2	BSC, EADS, EDF, SAP, TID, T6, UDUS, XLAB	Included evaluation results, submitted draft to internal reviewers.
4.0	30/11/07	WP4.2	BSC, EADS, EDF, SAP, TID, T6, UDUS, XLAB	Pre-final release finished, submitted to internal reviewers.
5.0	11/01/08	WP4.2	BSC, EADS, EDF, SAP, TID, T6, UDUS, XLAB	Final release finished.

Reviewers:
Antoine Ginies (Mandriva), Zhiwei Xu (ICT)

Tasks related to this deliverable:

Task No.	Task description	Partners involved°
T4.2.2	Revision of requirements and use case scenarios	BSC, EADS, EDF, SAP*, TID, T6, UDUS, XLAB
T4.2.4	Implementing and porting applications to XtreemOS	BSC, EADS, EDF, SAP*, TID, T6, UDUS, XLAB
T4.2.5	XtreemOS experiments and evaluation	BSC, EADS, EDF, SAP*, TID, T6, UDUS, XLAB

°This task list may not be equivalent to the list of partners contributing as authors to the deliverable
*Task leader

Abstract

The work package "Applications, Experiments and Evaluations" (WP4.2) contributes a range of reference applications which are used in two ways. On the one hand, the application references are the basis for the definition and revision of the XtreemOS requirements. On the other hand, the applications in WP4.2 are used to perform experimental evaluation of the available XtreemOS releases and intermediate XtreemOS components thereby providing regular feedback to the developers. This deliverable includes the second revision of the requirements. The revision includes the update of eight existing requirements and the introduction of four new requirements reflecting the ongoing design and development activities and the evolution of the project vision. The deliverable at hand further provides the evaluation of the first internal releases of various XtreemOS components including node-level VO support (from WP2.1), checkpoint and restart (from WP2.1), LinuxSSI (from WP2.2), SAGA API (from WP3.1), XtreemFS (from WP3.4, and CDA (from WP3.5). The evaluation is fully documented, comprising test plans, procedures for each test case, test logs and test results, ensuring traceability and repeatability. Furthermore, the fulfillment status of application requirements is given according to the evaluation results.

Contents

List of test documents

Chapter 1

Introduction

WP4.2 is in charge of defining and revising the requirements which are derived from a range of reference applications from different sectors. Furthermore, WP4.2 continuously conducts experimental evaluations of the XtreemOS components by means of executing reference applications using these components, thereby providing feedback to the developers.

In deliverable D4.2.1 [2], we presented the catalogue of initial requirements for XtreemOS. A first revision of these requirements was published in deliverable D4.2.3 [3]. The deliverable at hand provides a second revision of the requirements. Various requirements have been updated, further requirements have been introduced driven by the ongoing activities in SP2 and SP3 and by the deeper insights into applications' needs. The latest versions and the evolution of requirements are constantly communicated to the consortium for reasons of transparency and acceptance.

Deliverable D4.2.2 [1] reported on the early experiments with the Kerrighed operating system which is the foundation for the cluster flavor of XtreemOS currently being developed. After the state-of-the-art analysis and specification of XtreemOS, the consortium commenced the design and development during the second half of the first project year. From M13 to M17, first internal releases of various individual XtreemOS components became available for experimental evaluations by WP4.2. The test documentation including test plans, test specifications, logs, test results as well as an evaluation of the fulfillment status of application requirements are presented in the current deliverable. The scope of the evaluation has been identified in close cooperation with the respective developers such that the evaluations complement the continuous test and debugging procedures in SP2 and SP3.

The remainder of this deliverable is structured as follows. Chapter 2 summarizes the modifications to requirements and motivates the introduction of additional requirements. The documentation and the results of the experimental evaluation of the available XtreemOS components are presented in Chapter 3 along with an overview of the fulfillment status of application requirements. The conclusion

(Chapter 4) summarizes the main aspects and results of this deliverable and gives an outlook on future work. Appendix A provides a complete list of application requirements including all updates and extensions.

Chapter 2

Revision of Requirements

In close cooperation with the development work packages, we discussed and negotiated the application requirements previously published in deliverable D4.2.3 [3]. The requirements have been modified where appropriate to align them to the ongoing activities in SP2 and SP3 and to reflect the deeper insights into applications' and market needs. Eventually, a total of eight requirements were updated, four new requirements were added. The revision affected general requirements as well as requirements in the sections Federation Management, XtreemOS Interfaces, Highly Available and Scalable Services and Security in Virtual Organizations. We provide a list of new and updated requirements together with explanations motivating these changes. The entirety of all application requirements also including the extended and updated ones are presented in Appendix A. In the following sections, new requirements are listed with the whole text. For updated requirements, we only give a reference to the respective pages in the Appendix.

2.1 General Requirements

In this section of the requirements catalogue, we propose a new requirement (R7) considering the fact that six reference applications in WP4.2 are programmed in Java. Hence, it is essential that Java applications can be installed and executed on XtreemOS. Furthermore, the features and services provided by the XtreemOS project must fully support Java applications. Regarding the migration service, it is sufficient to at least migrate an individual Java Virtual Machine (JVM) and the Java application running in it as a whole.

Requirement R11 was introduced requesting the provision of self-installing software packages of XtreemOS releases. The selection of supported Linux distributions was driven by the needs of the WP4.2 applications and by the fact that two Linux distributors (Mandriva and Red Flag) participate in the XtreemOS consortium.

For reasons of consistency, in R19 the wording "multicast across clusters" was replaced by the more specific "within a cluster".

The revision of requirement R23 was motivated by the evaluation of the role management concepts presented in deliverable D3.5.4 [5]. Complex Linux applications require the possibility to execute different parts of the applications, e.g. the database in a multi-tier business application stack, in the context of different users and different groups. One example is a standard SAP business application which requires eight default application users:

```
> sdb:x:1000:1001:Database Software Owner:/home/sdb:/bin/bash
> lcaadm:x:1001:1001:Owner of Database Instance LCA:/home/lcaadm
  (cont.):/bin/csh
> pv4adm:x:1002:1002:SAP System Administrator:/home/pv4adm:/bin/
  (cont.)csh
> db2pv4:x:1003:1003:Database Administrator:/db2/db2pv4:/bin/csh
> sappv4:x:1004:1005:ABAP Database Connect User:/home/sappv4:/bin
  (cont.)/csh
> sappv4db:x:1005:1005:Java Database Connect User:/home/sappv4db
  (cont.):/bin/csh
> sapadm:x:1006:1002:SAP System Administrator:/home/sapadm:/bin/
  (cont.)csh
> trxadm:x:1007:1002:SAP System Administrator:/home/trxadm:/bin/
  (cont.)bash
```

Therefore it must be ensured that within a VO, it is possible to manage such differentiated users and groups.

List of extensions and changes:

New R7: Execution and Migration of Java Applications

It must be possible to at least migrate an individual JVM and the Java application running in it as a whole.

Previous number (D4.2.3): new
Updated: yes, new
Importance: obligatory
Importance for MDs: not applicable
Implementation order: to be determined

New R11: Packaging of XtreemOS releases

In order to increase the acceptance of XtreemOS in the users and developers community it is required that all intermediate and the final releases of XtreemOS are distributed as self-installing software packages including but not restricted to the rpm and deb format. These packages must be compatible with at least the following Linux distributions: Mandriva, Red Flag, and Debian.

Previous number (D4.2.3): new
Updated: yes, new
Importance: obligatory

Implementation order: to be determined

Updated R19: XtreemOS shall support multicast – see page 213 for new version of full text.

Updated R23: XtreemOS VOs must provide role management – see page 215 for new version of full text.

2.2 Federation Management

The misunderstanding about the selection of LinuxSSI nodes to be used by the application was resolved. A new requirement R41 was added, which specifies some additional variables that the SSI scheduler must be able to take into account. This allows the node selection requirements to be implemented simply by appropriate scheduling policies. Requirements R42 and R43 were thus updated accordingly.

List of extensions and changes:

New R41: Information available to SSI scheduler

The LinuxSSI scheduler must have access (at least) to:

- node-level and cluster-level variables, for example per-node CPU and memory usage,
- application-specified (or process-specified) variables, for example application's requirement to be scheduled only to nodes with certain properties,
- grid-level variables, for example a requirement that two processes must never be scheduled to the same node within the cluster. Such conditions must be coordinated with the grid-level scheduler.

The above will allow the LinuxSSI scheduler to satisfy requirements R42, R43, and R46. It must also be possible to define new variables and new, custom scheduling policies that take them into account.

Previous number (D4.2.3): New
Updated: yes, new
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: To be determined

Updated R42: Specification of service qualities in federations – see page 223 for new version of full text.

Updated R43: Node properties constraints in federations – see page 224 for new version of full text.

2.3 XtreemOS Interfaces

One further requirement (R49) was motivated by the evaluations performed in deliverable D3.5.4 [5]. For the acceptance of XtreemOS in the users and developers communities and for reasons of exploitability, it is essential that XtreemOS allows to install and execute Linux applications, in particular legacy applications. The background is that many complex applications and business solution installers require to make specific settings with privileged access (root) on target hosts. Settings on target hosts need to be customized and tuned for interoperability and efficient support. Configuration variables and registries defining the system behaviour have to be set or disabled in the area of accounts, file systems, network, hardware, licenses, timer etc. (cf. D3.5.4 for more details). Installers require POSIX compliant interfaces to perform these actions. Furthermore, it must be considered that often installations are interactive procedures which require an ongoing monitoring.

List of extensions:

New R49: XtreemOS must support Linux software with no need for modifications

It must be possible to install and execute Linux applications, in particular legacy applications, with no need for modifications neither to installers nor to application code. The Linux extensions and modifications produced by the XtreemOS project must be exploitable by such applications as far as possible. The fulfillment of this requirement largely leverages the acceptance of XtreemOS in the users (in particular industrial users) and developers community.

Previous number (D4.2.3): new
Updated: yes, new
Importance: obligatory
Implementation order: to be determined

2.4 Highly Available and Scalable Grid Services

In this section, four requirements were updated. The re-phrasing of requirement R53 considers the fact that the availability of nodes cannot be guaranteed. Instead it is necessary to provide for a high availability of allocated nodes during the whole job execution using virtual nodes as fault-tolerance mechanism.

Requirement R54 was changed, because depending on the underlying infrastructure, it may not be possible to give a bandwidth guarantee. However, many

Linux applications (including various reference applications in WP4.2) require a reliable minimum bandwidth. Therefore, XtreemOS must at least be able to estimate the available bandwidth with high precision. In this case, the goal is that the network links used by an allocated application provides a stable minimum bandwidth with high probability (user-defined) during the whole application runtime. This also requires a continued monitoring of the network with dynamic reallocation if needed.

In requirements R55 and R56, we added minimum and maximum values of priorities and further explanations for clarification. Requirement R55 was complemented by services to manage VOs. In requirement R56, additional measurements were included which are needed by applications on mobile devices. Further errors and redundancies were corrected.

List of changes:

Updated R53: Availability of nodes for a specified time – see page 227 for new version of full text.

Updated R54: High availability of bandwidth between the allocated nodes during the entire runtime of applications – see page 228 for new version of full text.

Updated R55: Grid services which have to be provided – see page 228 for new version of full text.

Updated R56: Measurement criteria which must be provided to estimate the quality of Grid services – see page 230 for new version of full text.

Chapter 3

Evaluation Report

In this chapter, we report on the setup, the procedure and the results of the experimental evaluation carried out by WP4.2. Section 3.1 positions the evaluations into the context of other test efforts within the consortium, an overview of the available XtreemOS components and their assignment to the various reference applications is given together with a general introduction into the test procedure and documentation. The actual evaluation of the available XtreemOS components is described in sections 3.2 to 3.7 including the test plans, test specifications, logs and test results. These results are summarized in Section 3.8 which also gives an overview of the fulfillment status of all requirements defined by WP4.2.

3.1 Overview

The reference applications in WP4.2 (cf. table 3.1) are used to perform experiments to evaluate selected XtreemOS components or entire XtreemOS prototypes and packages (as soon as they become available). In general, testing of XtreemOS development is carried out by various groups on different levels:

- Developers level: Tests carried out by work packages developing components (SP2 and SP3).
- Packaging level: Tests carried out by WP4.1 (Mandriva and RedFlag), QA with an automatic test platform (reliability, non-regression, etc).
- User/application level: Tests carried out by WP4.2, feedback to the developers (SP2 and SP3) regarding functional and non-functional issues.

The evaluation performed by WP4.2 focuses on the installation of XtreemOS developments and on the execution of the reference applications using the functionalities provided by the components under test. Typically, the evaluation targets aspects like: fulfillment of requirements, stability, performance, scalability and usability. As opposed to the tests carried out on developers or packaging level, the evaluation with reference applications allows for testing the XtreemOS from the

user perspective using input data with practical relevance. The tests are planned in cooperation with the developers which allows to identify the scope of the tests and to provide useful feedback which can be exploited within the continuous development activities. In the case of erroneous system behaviour, the evaluation results give detailed reports on the incidents observed. The efforts also include the analysis of the problem as far as this is possible from the user or application perspective. Code-level problem analysis and debugging do not belong to the scope of WP4.2.

Table 3.1 provides an overview of the teams contributing to WP4.2 and their corresponding applications. Henceforth, the short names of the applications are used instead of the full names.

Partner	**Application Name**	**Short Name**	**Application Area**
BSC	Specweb2005	SPECWEB	Enterprise solutions
BSC	GRID superscalar fastDNAml	GSDNA	Bio-informatics
EADS	Elfipole	ELFIPOLE	Electromagnetics
EADS	jCAE	JCAE	Computer aided engineering
EDF	Moderato	MODERATO	Particle physics
EDF	Simeon	SIMEON	Optimization
EDF	Zephyr	ZEPHYR	Fluid mechanics
EDF	Secured Remote Computing	SRC	Enterprise solutions
SAP	SAP Netweaver Application Server	WEBAS	Enterprise solutions
T6	DBE	DBE	Enterprise solutions
TID	TID Instant Messaging Application	IMA	Instant messaging
TID	Job Management Application	JOBMA	XtreemOS job management
UDUS	Wissenheim	WISS	Virtual Presence
XLAB	Galeb	GALEB	Economics, optimization

Table 3.1: Overview of the applications in WP4.2.

The first entire XtreemOS prototype will be released in M24. Therefore the evaluations for this deliverable restrict to the evaluation of the XtreemOS components which were available and ready for user-/application-level testing in M17. Table 3.2 gives an overview of the XtreemOS components under development and their respective readiness for evaluation. Note that the tested components are still internal or alpha releases which may be incomplete, instable and not yet optimized for performance.

WP	Component Name	Readiness
2.1	Node-level VO support	yes
2.1	Checkpoint and restart	yes
2.2	Kerrighed (32 bit version)	yes
3.1	SAGA API (C++ engine)	yes
3.2	Publish/subscribe service	no

3.2	Node management	no
3.2	Directory service	no
3.2	Virtual nodes	no
3.2	Distributed servers	no
3.3	Scheduler	no
3.3	Monitoring and accounting	no
3.3	Job submission	no
3.3	Interaction with jobs	no
3.3	Checkpoint, restart, migration (Grid-level)	no
3.3	Job self-management	no
3.4	XtreemFS	yes
3.5	Credential Distribution Agency (CDA)	yes
3.5	Virtual Organization Policy Service (VOPS)	no
3.5	Identity Service	no
3.5	Attribute Service	no
3.5	Virtual Organization Membership Service	no

Table 3.2: Overview of components and their readiness for evaluations by WP4.2

Accordingly, table 3.3 gives an overview of the tested XtreemOS components along with the applications used for evaluation.

WP	Component Name	SPECWEB	GSDNA	ELFIPOLE	JCAE	ZEPHYR	WEBAS/MaxDB[1]	DBE	WISS	GALEB	other
2.1	Node-level VO support				X					X	
2.1	Checkpoint and restart	X					X	X			
2.2	LinuxSSI (Kerrighed)			X						X	
3.1	SAGA API					X					
3.4	XtreemFS		X				X		X		
3.5	Credential Distribution Agency (CDA)										X

Table 3.3: Overview of available components and their assignment to applications in WP4.2

[1] In the multi-tier SAP stack, MaxDB is the central database through which the majority of accesses to the file system is carried out. Therefore MaxDB has been chosen as an appropriate benchmark for testing XtreemFS. Typically, MaxDB is used by multiple distributed instances of the SAP Netweaver Application Server (WEBAS).

Each component (except the SAGA API and CDA) was evaluated by at least two applications. This allows to test the components exploiting different application characteristics and heterogeneous testbed setups. Therefore the experiments cover features which are examined by each test application, whereas other features are explicitly tested by a certain application. In the case of SAGA, the goal is to gain first experience with the API due to the lack of adapters for XtreemOS services. More extensive tests are planned for the forthcoming SAGA releases. In the case of the CDA, the normal usage is to call this component rather infrequently (e.g. once a month) to get an XOS Certificate. Furthermore, CDA does not provide an API yet. Therefore, the evaluation of CDA within this deliverable is restricted to preliminary tests using a dedicated Java application and shell commands. For succeeding deliverables the DBE application will be a potential candidate for testing the CDA via API. The DBE application uses an identity system based on certificates, more in depth, the application uses X509 certificates, the same type of certificate generated by the CDA application. The DBE application has been written in Java and uses standard methods in order to work with certificates.

The node-level VO support was tested with JCAE and GALEB. GALEB tests started with the port of Galeb that was made for the demonstration of `xos-ssh` in Pisa in June 2007. The port uses `ssh` for remote process execution and `scp` for file transfer, which makes it a natural choice for access permissions testing. Even if it's no longer the best way to do it, jCAE was several years ago, distributed using `ssh`. This feature was used to test the `xos-ssh`.

The checkpointing and restart tests were performed using three different applications from three different areas. The Specweb benchmark was used to verify the modules capability of handling multithreaded Java applications. The SAP WebAS checkpointing and restart tests are used to evaluate the capabilities in combination with static and dynamic libraries as well as with database handles. This kind of setup is typical for many business applications. Furthermore, the SAP WebAS has several different application stacks which results in several parallel running threads. Thus, the capabilities of the checkpointing mechanisms regarding these issues are addressed. The DBE application maintains several applications running in different threads. The ability of checkpointing and restart those applications when there is a failure is also appreciated.

Most parts of the LinuxSSI were still undergoing heavy development at the time of the testing, thus only migration and checkpoint/restart were tested. Note that the checkpointing and restart mechanisms in LinuxSSI are implemented using the KDDM concept of Kerrighed and is different from the checkpointer implemented for the PC flavor of XtreemOS (extension of BLCR). Elfipole is a multiprocesses parallel application which can benefit from from process migration. It can be configured to used different types of IPC so it can test many system primitives implemented in Kerrighed. SSI clusters are most appropriate for running SMP-type of parallel applications, thus Galeb was adapted to use SysV message queues for communication between the computing processes. Migration in particular was tested thoroughly with Galeb because the ability to migrate individual

processes of a parallel application is crucial to load balancing in the SSI system.

Though SAGA API was not available yet for XtreemOS, some basic tests could be done with the SAGA implementation on a regular Linux system. SAGA API was tested using some simple programs written in C++ that perform job or file management. We expect that these pieces of code will be merged with no major changes to EDF applications : ZEPHYR, MODERATO, or SIMEON (once the wrapping of the API is available in C or Fortran 90 for ZEPHYR or SIMEON).

GSDNA, MaxDB and WISS application tests have been chosen to evaluate the XtreemFS component because they are only a few applications but nicely cover a very wide range of data access characteristics. While the installation procedure of GSDNA is file-metadata intensive, the MaxDB test incur only a few metadata operations, but many data chunk transfers. The chosen applications also have a diverse spacial locality of access within files: GSDNA application has a strong sequential access pattern, typical for scientific applications; the MaxDB database system mainly accesses small chunks in a few big files randomly. WISS and GSDNA tests exhibit concurrent I/O with many file system clients running in parallel; MaxDB is a single-client application. GSDNA and MaxDB only use POSIX operations to access files on XtreemFS; WISS, instead, exploits a special Object Sharing Service layer (also known as GOM layer) of XtreemFS for concurrent client access to file-mapped data.

Note that the applications IMA and JOBMA will be used in future evaluations when advanced features of XtreemOS become available.

All experiments were carried out on the local testbeds of the various partners. The goal of the inhouse tests is to gain first experience with the early releases of XtreemOS components before performing large-scale evaluations on the GRID5000 testbed. The problems discovered with the early releases need to be resolved before executing experiments on the Grid testbed to efficiently use the shared resources provided by the institutions contributing to Grid5000. After successful inhouse tests, the experiments can be repeated on GRID5000 on a larger scale with geographical dispersion among computational nodes. It is also recommended that the developers port the XtreemOS components to GRID5000 and perform initial test runs to provide ready-for-use system images to the users in WP4.2.

The following sections 3.2 to 3.7 present the test documentation which is based on the "IEEE Standard for Software Test Documentation", IEEE 829-1998 [8]. Accordingly, the documents in these sections cover the phases test planning, test specification, and test reporting.

Among others the *test plan* describes the scope, approach, resources, the items and the features to be tested and the testing tasks to be performed. Per XtreemOS component, one test plan is provided covering the test setup for all WP4.2 applications evaluating this component.

The test specification consists of three document types:

- The *test design specification* gives more details on the test approach and

presents the features covered by the design and its respective tests. Furthermore the associated test cases and test procedures are identified, and the feature pass/fail criteria are specified. Usually, for each application testing a certain XtreemOS component, a separate test design is provided.

- The *test case specification* provides the actual input values used as well as the anticipated outputs. Moreover, the test case describes the constraints on the test procedures.

- The *test procedure specification* explains all steps needed to operate the test system.

The test reporting consists of three document types :

- The *test log* records events, input and output during test execution. We use these documents to record events which do not require further investigation.

- The *test incident report* records all events during the test execution which need further examination.

- The *test summary report* summarizes the tests. For each XtreemOS component, we give a common summary for all applications testing it.

Note that we do not use the *test item transmittal report* proposed by the standard. Details on the releases tested along with the information on development groups and test groups are provided within the other test documents.

According to the suggestions of IEEE 829-1998, each test document has an identifier which allows for a unique reference to tests also across the subsequent deliverables of WP4.2. The following naming convention is applied for the identifiers:

- wpXX-CC-tsYY-NAME for the test plan and the test summary report

- wpXX-CC-tsYY-APPZZ-NAMENN for all other test documents

Explanation:

- XX = work package number

- CC = short XtreemOS component name (2 or 3 letters)

- YY = enumerator for test sequence. Test sequences describe the self-contained setup and activities for testing a certain component. A new test sequence may be created, e.g., in the case of new software releases or for new test setups.

- APP = short application name

- ZZ = test case number
- NN = running number for document
- NAME = type of document. Here we use the following abbreviations:
  - tp = test plan
  - tds = test design specification
  - tcs = test case specification
  - tps = test procedure specification
  - tl = test log
  - tir = test incident report
  - tsr = test summary report

Example: Node-level VO support from WP2.1 tested in test sequence 1 by application GALEB.

Document type	Identifier
test plan	wp21-vo-ts01-tp
test design specification 1	wp21-vo-ts01-galeb01-tds01
test case specification 1	wp21-vo-ts01-galeb01-tcs01
test procedure specification 1	wp21-vo-ts01-galeb01-tps01
test log 1	wp21-vo-ts01-galeb01-tl01
test incident report 1	wp21-vo-ts01-galeb01-tir01
test summary report	wp21-vo-ts01-tsr

3.2 Evaluation of Node-level VO Support

The node-level VO support component from WP 2.1 (short name: wp21-vo) provides a bridge between local accounts on nodes and identities of global grid users that are members of various VOs.

3.2.1 Test plan – VO

Test plan identifier: wp21-vo-ts01-tp

Introduction

This test plan covers the software for node-level VO support, which includes:

- verification of grid user's tokens for accessing a grid node,
- mapping global user identities to local user identities and group identities,

- translating VO-level access control policies into local OS-level access rights and capabilities,
- session management.

The purpose, architecture and use cases are described in the XtreemOS deliverable D2.1.2 [4].

Test Items

The current version of the software (as of the time of writing this test plan) includes the following parts:

- *nss_pam*, a VO-aware NSS (Name Service Switch) and a VO-aware PAM (Pluggable Authentication Module),
- *xos-openssh*, ssh daemon and client for use with *nss_pam*,
- *ams_server*, an account mapping service.

The source code is available at the XtreemOS SVN server as a tarball. The last version (as of 2007-09-24) is tagged 0.71, was released on 2007-06-27, and consists of *nss_pam* 0.031 and *xos-openssh* 0.7.1. *ams_server* must be obtained separately. All parts are accompanied by basic documentation in the form of installation instructions and README files.

The contact person for the software is Haiyan Yu (`yuhaiyan@ict.ac.cn`).

Features to be Tested

The following features will be tested:

- mapping of VO users to accounts on the VO-aware node (test design wp21-vo-ts01-galeb01-tds01),
- decisions on local access rights and capabilities (test design wp21-vo-ts01-galeb01-tds01),
- isolation between users belonging to different VOs and between those belonging to the same VO (test design wp21-vo-ts01-galeb01-tds01).

Features not to be Tested

At this stage, the following features will not be tested:

- groups and group ID mapping,
- VO roles and role mapping,
- quota specification and enforcement,

- performance,
- mapping from user to VO name,
- security.

These features will be tested on later versions of the software. In particular, security will be tested later because version 0.71 does not check certificate validity at all. It also has no way of checking whether the user really is a member of the VO because it has not yet been decided whether this information will be stored in a certificate attribute or elsewhere.

Approach

The purpose of these early tests is to evaluate the current version of software, provide feedback to developers, and check which requirements listed in XtreemOS deliverable D4.2.3 are fulfilled, which are partially fulfilled, and which remain to be done. We will thus focus on evaluating the higher-level design, features and usability of each module rather than bugs in the implementation. Correspondingly, the test plan, item pass criteria etc are not given in too much detail. The test plan can also be adapted during testing to the current state of the modules and to their usage philosophy.

To ensure repeatability and comparability of tests among partners, all tests will be done on a commonly available Linux distribution, using stable versions of the required third-party libraries where possible. The test documentation should include a description of the installation procedure and in particular detailed descriptions of any deviations that were made from the installation instructions supplied with the software and of steps that are not described adequately in the instructions. Such documentation will help the developers to improve installation instructions as well speed up the installation process for other partners.

Item Pass Criteria

Mapping of VO users will pass the test if users (identified by their certificate proxies and VO names) are correctly mapped to either existing local accounts, predefined local account names collected into an account pool, or accounts created on demand for each VO user, as specified in the configuration files of each test case. The system must grant or deny access rights for each case in a sensible way.

Users belonging to different VOs must, at this stage, be isolated to the same degree as different local users on the same machine. The same applies to two sessions started by a single grid user under different VO names. Users belonging to the same VO must be isolated like different local users belonging to the same group.

Testing Tasks

The test preparation requires the following tasks:

1. preparing the testing environment, e.g., installing required libraries,
2. obtaining and compiling the software,
3. creating user certificates and certificate proxies,
4. initial configuration of the software.

Each test then consists of adapting the configuration to the specific test, performing the test, and documenting the full procedure. The individual tests can be executed in any order, but the order of tasks during preparation and during execution of tests must be as given here.

Environmental Needs

The required resources include two computers. The installation and testing might interfere with normal usage of the nodes so it is recommended to use dedicated computers or virtual machines. There are no specific hardware requirements.

The machines must run Linux with kernel 2.6.12 or newer and the following libraries:

- OpenSSL 0.9.6 or newer,
- Berkeley DB,
- PAM,
- open SSH.

The development files of each libraries, e.g., the header files, are also required to compile the software.

Additionally, a utility for managing user certificates is helpful at this stage, such as the *grid-cert-** and *grid-proxy-init* utilities from the Globus Toolkit.

Responsibilities

This test plan and the included tests with the Galeb application are the responsibility of XLAB. The included tests with the jCAE applications are the responsibility of EADS.

Schedule

Being that the software is in its early development stage, the installation, setup and learning are complex processes, requiring one or two weeks if the staff is not familiar with the procedure. However, after the preparation is finished, the tests are simple and will be done in a few days.

All the planned tests should be finished in time for XtreemOS deliverable D4.2.4, which is due on 30. 11. 2007.

Risks and Contingencies

Apart from delays in testing caused by late delivery of new versions of the software and by unexpected installation problems, no specific risks are envisaged. Note that late delivery of new versions of the software can be a consequence of bugs discovered by these tests.

3.2.2 Test design specification

Test design specification identifier: wp21-vo-ts01-galeb01-tds01
Test plan reference: wp21-vo-ts01-tp

Features to be Tested

The following features are the subject of this test design specification:

1. mapping of VO users to accounts on the VO-aware node (test cases wp21-vo-ts01-galeb01-tcs01 and wp21-vo-ts01-galeb01-tcs03, testing requirements R89, R22, R24, R26, R28, R96),

2. decisions on access rights and isolation between users belonging to different VOs and between those belonging to the same VO (test cases wp21-vo-ts01-galeb01-tcs02 and wp21-vo-ts01-galeb01-tcs03, testing requirements R25, R85, R28, R86, R96).

To summarize, the fulfillment of the following requirements from D4.2.3 will be evaluated:

- full test: R25, R85, R89,

- partial test: R22, R24, R26, R28, R86, R96.

Approach Refinements

This test design covers very basic tests, which can be done either using the Galeb application, where the application spawns jobs to other nodes and accesses files on those nodes, but also using the command-line utility `ssh-xos`. The latter

approach is simpler to execute and will thus be used mostly, while the Galeb application will be used in the test case wp21-vo-ts01-galeb01-tcs03.

Galeb, as used in this test design, is a command-line application that is given two text files: input data (a sampled function) and a list of computational nodes. The master process copies the input data file to all accessible nodes from the list using `scp-xos`, then starts computational processes on all nodes using `ssh-xos`. Each computational process creates a results file (an algebraic expression that approximates the input data, obtained by a genetic algorithm, plus a measure of error). The master process copies all partial results back to the master node, then simply selects the best one.

Test Identification

The test cases wp21-vo-ts01-galeb01-tcs01 and wp21-vo-ts01-galeb01-tcs02 test one feature each from the list in wp21-vo-ts01-galeb01-tds01. The test case wp21-vo-ts01-galeb01-tcs03 tests the features tested by the first two cases in an actual application environment and will thus be done after the first two.

Feature Pass/Fail Criteria

Feature 1 passes the test if the user is mapped according to the distinguished name stored in the certificate and the VO name given as a command-line parameter to `ssh-xos`, as explained in to D2.1.2. No password should be required. The feature fails if:

- an unauthorized user manages to log into the node,
- an authorized user manages to log into the node, but is mapped incorrectly,
- an authorized user is denied access, or
- changing the VO/user mapping setting during runtime of Galeb application causes any problems.

Feature 2 passes if the remote user can access and execute exactly the same set of files as the local account she is mapped to. The feature fails in all other cases.

3.2.3 Test case specification – VO mapping Galeb

Test case specification identifier: wp21-vo-ts01-galeb01-tcs01

Test Items

This test case tests mapping of VO users to accounts on the VO-aware node.

Input Specifications

The input of this test case consists of the rules to be used to map VO users to accounts on the node and by the `vo-name` parameter passed to `ssh-xos`.

The following cases will be tested:

1. mapping to an existing local account,
2. mapping to an account pool,
3. mapping to a dynamically created account.

Each case requires one of the following entries in the `/etc/xos/mapping/\amappedfile`:

```
vo-name:certificate_subject    marjan          #local account
vo-name:certificate_subject    .accountpool1   #account pool
vo-name:certificate_subject    *               #dynamically created
    (cont.)account
```

Each case must be tested with the vo-name as given in the mapping file (in which case the command must succeed) and with another vo-name (which should fail).

Case 2 also requires the file `/etc/xos/mapping/accountpool1`, which should contain a list of account names. The accounts used for account pool must not exist yet in the system.

Output Specifications

The command run remotely by `ssh-xos` must run under the account specified in the `amappedfile`.

3.2.4 Test procedure specification

Test procedure specification identifier: wp21-vo-ts01-galeb01-tps01
Test design specification reference: wp21-vo-ts01-galeb01-tds01
Test case specification reference: wp21-vo-ts01-galeb01-tcs01
Test log identifier: wp21-vo-ts01-galeb01-tl01

Purpose

This procedure executes the test case wp21-vo-ts01-galeb01-tcs01.

Procedure Steps

Installation

1. First, the component must be compiled and installed according to the instructions supplied with the component in the `README` and `INSTALL` files.
2. The directory `/var/xos` must be created by root because version 0.71 does not create it during installation.

Log Logging can be enabled by manually starting the `sshd-xos` daemon with parameters `-d` (lowest level of logging), `-dd` or `-ddd` (highest level). Please note that the deamon started in this way will only serve one client session and will terminate when the session ends.

Set Up

1. The user certificate must be created, for example using `openssh` as described at http://www.flatmtn.com/computer/Linux-SSLCertificates.html. A certificate proxy must then be created, for example using the Globus tool `grid-proxy-init`, and put into the file `/.xos/proxy.pem`. Alternatively, this step may be replaced by using the file `proxy.pem` supplied in the `config` directory of the component sources.

2. The mapping file(s) must be prepared according to the test case specification. The certificate subject to be used in the file can be extracted from the certificate proxy (and not from the certificate itself!) by using the command `grid-cert-info -file .xos/proxy.pem -s`.

Start The test is run by typing the command
`ssh-xos -J vo-name hostname whoami`.

Wrap Up Version 0.71 of the component does not update the mapping scheme when the `amappedfile` changes. The only way to apply the changes is to delete the files in the `/var/xos` directory, which must therefore be done after each test.

Contingencies The anomalies should be dealt with by restarting the `sshd` daemon and by the actions described in the Wrap Up paragraph.

3.2.5 Test log

Test log identifier: wp21-vo-ts01-galeb01-tl01

Description

The test procedure wp21-vo-ts01-galeb01-tps01 was executed by Marjan Šterk, XLAB. The nodes used were `muca-0` and `muca-1`, based on 64-bit AMD Sempron processors and originally running Debian GNU/Linux 4.0 with kernel 2.6.8-11-amd64-generic. The kernel was upgraded during the execution (see wp21-vo-ts01-galeb01-tl01).

We tested the version 0.71 of the component (released on 2007-06-27), which includes `nss-pam` 0.031. `nss-pam` 0.04 was released a few days prior to the execution of the test but this version does not work with `ssh-xos` and is thus not usable. We used revision 396 of the `ams-server` the SVN (committed on 2007-06-12).

Installation and Setup

Execution Description The compilation and installation was started on 2007-09-04 following the README files supplied with the software. The following modules had to be installed using the `apt-get` command: libdb4.4, libdb4.4-dev, libpam0g, libpam0g-dev, libssl-dev, and libgdbm-dev.

Additionally, since the installation folder was `/usr/local/xtreemos/` instead of the default, the `-I/usr/local/xtreemos/include` option had to be added to the Makefile of the `ams_server`. This also required correcting the `sshd-xos` script, and all lines that have to be added to the `init.d/*` configuration files.

The first problem encountered was an error message at system startup:

```
xos_dbaux: Database '/var/xos/GpasswdDB.db' open failed.: No such
  (cont.)file or directory
databases_setup: ERROR: No such file or directory
```

Consultation with the author and following his suggestion to manually create the directory `/var/xos` solved the problem.

After unsuccessful attempts to log in as a VO user, the kernel was upgraded to 2.6.18-5-amd64 because the documentation states that 2.6.8 is not supported. However, the problem turned out to be caused by incorrect mapping file so we do not know whether upgrading the kernel changed anything.

Procedure Results After successful installation the software resides in the directory `/usr/local/xtreemos/`. The executables have the suffix `-xos` and the default port of `sshd` is 2222, so the component does not clash with the existing installation of `sshd` and other software.

Anomalous Events Apart from wrong configuration, the only problem was the need to manually create the database directory. This action solved the problem but future versions should create the directory automatically on installation.

Running the Test Procedure

Execution Description The procedure was run on 2007-09-24 following the procedure specification wp21-vo-ts01-galeb01-tps01.

Procedure Results Table 3.4 summarizes the local accout to which the VO user is mapped. The VO name in the amappedfile was always 'myvoname'. The account name can be either an existing local account ('marjan' in our example), dynamically created account from an account pool ('account1'..'account3'), or dynamically created account with the user name equal to the certificate subject. '*AD*' denotes denied access. The cases where mapping is incorrect are shown in bold.

When mapped to *CertS*, the console output of the `whoami` command is:

Table 3.4: Mapping to local accounts with incorrect mapping shown in **bold**.

entry in amappedfile		VO passed	actual
certificate subject	account to map to	to ssh-xos	mapped account
correct	marjan	myvoname	marjan
correct	marjan	myvonam	*AD* or **marjan**
correct	marjan	othervo	*AD*
incorrect	marjan	*any*	*AD* or **root**
⋆	marjan	*any*	**root** or ***AD***
correct or ⋆	.accountpool1	myvoname	account?
correct or ⋆	.accountpool1	myvonam	*AD* or **account?**
correct or ⋆	.accountpool1	myvona	*AD* or **account?**
correct or ⋆	.accountpool1	m	*AD* or **account?**
correct or ⋆	.accountpool1	othervo	*AD* or **root**
incorrect	.accountpool1	*any*	*AD*
correct or ⋆	⋆	myvoname	*CertS*
correct or ⋆	⋆	myvonam	*AD* or ***CertS***
correct or ⋆	⋆	othervo	*AD* or **root**
incorrect	⋆	*any*	*AD*
no entry		*any*	*AD* or **root**

```
libnss_xos.c:84: NSS: Mapping scheme: Raw DN
/O=Grid/OU=GlobusTest/OU=simpleCA-gsmlserver/CN=client/CN
   (cont.)=1262133077
libnss_xos.c:94: NSS: Access to database: All users
```

The output for access denied is:

```
groupmapping: found existed mapped GROUPS
No records found for the token
accountmapping2: read_conffile fail
Did you define the mapping rule?
No records found for the token
groupmapping: found existed mapped GROUPS
No records found for the token
accountmapping2: read_conffile fail
Did you define the mapping rule?
No records found for the token
Could not chdir to home directory /nonexist: No such file or
   (cont.)directory
/nonexist: No such file or directory
```

And the output when (incorrectly) mappet to root:

```
passwd.c:107: assertion 'gpwd->l_idtoken.g_mappedname != NULL &&
   (cont.)gpwd->l_idtoken
.g_mappedname[0] != '\0'' failed in function _nss_xos_getpwnam_r(
   (cont.)is this a bug?)
```

```
passwd.c:109: assertion 'gpwd->l_idtoken.g_mappeduid != 0' failed
   (cont.)in function _n
ss_xos_getpwnam_r(is this a bug?)
passwd.c:111: assertion 'gpwd->l_idtoken.g_mappedgid != 0' failed
   (cont.)in function _n
ss_xos_getpwnam_r(is this a bug?)
Could not chdir to home directory : No such file or directory
root
```

Anomalous Events

As given in Table 3.4, access was granted in many cases where it should not have been, sometimes even as root. Detailed analysis is given in the incident report wp21-vo-ts01-galeb01-tir01.

Furthermore, in case of a mapping entry stating that any user from 'myvoname' should be mapped to an existing local account, the user is instead mapped to root or denied the access. The latter represents a less critical bug that may or may not be a consequence of the first one, thus it will be researched after the first one is corrected.

Another anomaly was the inability to reuse account from the account pool. Detailed analysis is given in the incident report wp21-vo-ts01-galeb01-tir02.

3.2.6 Test incident report – VO mapping re-login

Test incident report identifier: wp21-vo-ts01-galeb01-tir01
Test log reference: wp21-vo-ts01-galeb01-tl01

Summary

After a successful login is done on behalf of the VO in the amappedfile, the same user can log in on behalf of another (incorrect) VO and is mapped either to the same local account or to root.

Incident Description

During testing on 2007-09-24, the software behaved as follows. If the user's certificate subject and VO name matched the data in amappedfile, the behaviour was always correct (access granted, account mapping as given in amapped file).

If the user then alters her VO name, she should be denied access because members of other VOs are not mentioned in the amappedfile. Instead, the user is granted access and is mapped either to the same account as if she were a member of the correct VO (if the new VO name starts with the same character as the correct one) or, even more critical, to **root** (if the new VO name starts with another character). Such behaviour persists until the server is restarted. Note that only commands can

be executed, while running `ssh-xos` without a command produces a "connection closed by remote host" response. However, the user can still get access to the console by passing `/bin/bash` as the command to be executed.

If the user tries to log in on behalf of an incorrect VO before there had been a successful login, the access is denied (as it should be).

The summary of all cases is given in Table 3.4 on page 32.

Impact

The bug represents a fatal security hole in the component. The developers have been notified about the problem and are working on it. Other tests are not affected by this incident.

3.2.7 Test incident report – VO mapping no account reuse

Test incident report identifier: wp21-vo-ts01-galeb01-tir02
Test log reference: wp21-vo-ts01-galeb01-tl01

Summary

Accounts in the account pool are not reused after restart of the `sshd-xos` daemon. They are only reused if manually deleted from the `/etc/passwd` file.

Incident Description

During testing on 2007-09-24, the software behaved as follows. On first usage a user with the name equal to the first account enumerated in the account pool is created. The account is not deleted after the session ends.

When server (`sshd-xos`) is restarted and the database deleted (since that is the only way to apply changes to VO composition), the first account is not reused. Instead, the second one is created.

After the server runs out of accounts in the account pool, user is incorrectly denied access. The access is possible again only after manual deletion of the locally created accounts.

Impact

The bug significantly affects the usability of account pools. It also requires extra attention when executing parts of test cases wp21-vo-ts01-galeb01-tcs02 and wp21-vo-ts01-galeb01-tcs03.

3.2.8 Test case specification – VO access Galeb

Test case specification identifier: wp21-vo-ts01-galeb01-tcs02

Test Items

This test case tests decisions on access rights and isolation between users. Only access to files will be tested – isolation of other resources, e.g. interprocess communication, is not a subject of this test case.

Input Specifications

The input of this test case is given by rules to be used to map VO users to accounts on the node and by the files that the tester should try to access/execute.

All three mappings given in test case wp21-vo-ts01-galeb01-tcs01 should be tested. For each case, the tester should log into the node and try reading, writing, and executing a test file owned by:

1. the local user that the tester is mapped to (also, the local user should try accessing a file created by the tester),
2. another local user (also, the local user should try accessing a file created by the tester),
3. another VO user belonging to the same VO (and using all three mappings),
4. another user belonging to another VO,
5. the same VO user (same certificate subject) but logged in under a different VO name,
6. the same VO user logged in under the same VO name but files created in a previous session.

The test file must be made accessible only by the owner by using the command `chmod 700 <filename>`. The file must be executable, be for example:

```
#!/bin/bash
echo Executed!
```

Output Specifications

The output should be consistent with the feature pass/fail criteria specified in wp21-vo-ts01-galeb01-tds01. The details are given in Table 3.5.

Obviously, two VO users are isolated if and only if they are mapped to different local uids. Likewise, a VO user and a local user are isolated if and only if the VO user is mapped to a local uid different from the local user's. This test case is thus in fact an extension of the account mapping test case wp21-vo-ts01-galeb01-tcs01. In any case the isolation is exactly as good as isolation between different local users.

Table 3.5: Access modes that should be allowed

login mode for user1@VO1	local account	account pool	*
file owner			
user1@VO1 (another session)	rwx	rwx	rwx
user1@VO2	rwx/- - -[a]	?[b]	?
user2@VO1	rwx/- - -	- - -	- - -
user2@VO2	rwx/- - -	- - -	- - -
local user	rwx/- - -	N/A[c]	N/A
local user accessing file of user1@VO1	rwx/- - -	N/A	N/A

[a] rwx if both are mapped to the same local account, - - - if not

[b] Not clear from the requirements; the test will only note the result and will not decide on the correctness

[c] Local usage of pooled and * accounts is not envisaged.

Intercase Dependencies

This test case should be preceded by wp21-vo-ts01-galeb01-tcs01. Testing the isolation only makes sense for those cases of wp21-vo-ts01-galeb01-tcs01 that work correctly.

3.2.9 Test procedure specification

Test procedure specification identifier: wp21-vo-ts01-galeb01-tps02
Test design specification reference: wp21-vo-ts01-galeb01-tds01
Test case specification reference: wp21-vo-ts01-galeb01-tcs02
Test log identifier: wp21-vo-ts01-galeb01-tl02

Purpose

This procedure executes the test case wp21-vo-ts01-galeb01-tcs02.

Procedure Steps

Installation, Log and Set Up See wp21-vo-ts01-galeb01-tps01.

Additionally, create the test files under all accounts listed in wp21-vo-ts01-galeb01-tcs02 and `chmod` it to 700.

Start For each case of a VO user accessing a file, type the following commands:

```
cat FILE
./FILE
whoami | xargs echo echo Written to by >>FILE
```

Wrap Up and Contingencies See wp21-vo-ts01-galeb01-tps01.

Table 3.6: Access modes that are allowed

login mode for user1@VO1	local account	account pool	*
file owner			
user1@VO1 (another session)	rwx	rwx	rwx
user1@VO2	rwx/- - -	- - -	- - -
user2@VO1	rwx/- - -	- - -	- - -
user2@VO2	rwx/- - -	- - -	- - -
local user	rwx/- - -	N/A	N/A
local user accessing file of user1@VO1	rwx/- - -	N/A	N/A

3.2.10 Test log

Test log identifier: wp21-vo-ts01-galeb01-tl02

Description

Test procedure wp21-vo-ts01-galeb01-tps02 was was executed by Marjan Šterk, XLAB. The nodes used were `muca-0` and `muca-1`, based on 64-bit AMD Sempron processors and running Debian GNU/Linux 4.0 with kernel 2.6.18-5-amd64.

The procedure was executed immediately after wp21-vo-ts01-galeb01-tps01, so the versions of the software were the same.

Running the Procedure

Execution Description The procedure was run on 2007-09-26 following the procedure specification wp21-vo-ts01-galeb01-tps02.

Procedure Results Table 3.6 summarizes the allowed access modes for each combination of file owner and accessor. It is consistent with the modes given in Table 3.5 on page 36.

In the case of * mapping, it was noted that user1@VO1 and user1@VO2 were mapped to local users with the same username (equal to the certificate subject) but different uids. If the uids were also the same, isolation would not be possible.

Anomalous Events

No anomalous events were encountered, other than those already documented in test log wp21-vo-ts01-galeb01-tl01.

3.2.11 Test case specification – VO running Galeb

Test case specification identifier: wp21-vo-ts01-galeb01-tcs03

Test Items

This test case wraps up the results of wp21-vo-ts01-galeb01-tcs01 and wp21-vo-ts01-galeb01-tcs02 by running the Galeb application. Additionally, it test how changing VO composition during application runtime affects the application.

Input Specifications

The input is given by rules to be used to map VO users to accounts on the node. The actual data input to Galeb is not important because we are not interested in the results. The genetic algorithm parameters given in the Galeb configuration file should be set so that each calculation on the slave takes long enough to monitor the execution, i.e. at least one minute.

Two different nodes should be used as slaves. The following mapping combinations should be used:

1. Mapping to a local account on first node and to an account pool on the second node.

2. Local account on one node and * on the other.

3. Runtime change of VO composition: change one of the mappings during slave process execution. Also, delete one of the mappings during runtime.

4. Lastly, mapping to a local account on one node and no mapping on the other.

Output Specifications

No errors should be reported by the application and the output file `result.txt` should appear in the current directory, except in the case where no mapping exists. The output file content is not important.

Ideally it should be possible to set the VO policy so that runtime changes do not affect the running application. However, this is not possible with the current implementation of Galeb-over-xos-ssh, so the last case is expected to fail.

Environmental Needs

All environmental needs stated in wp21-vo-ts01-tp plus the following.

Software Galeb application (master executable, slave executable, configuration file, input data) in the current directory.

Special Procedural Requirements

When testing R26, change the VO composition during the slave process execution (and thus before collection of results and before temporary file deletion) by changing the mapping file. Try deleting the respective mapping line as well as changing the mapping to another account.

Intercase Dependencies

This test case should be preceded by wp21-vo-ts01-galeb01-tcs01 and wp21-vo-ts01-galeb01-tcs02.

3.2.12 Test procedure specification

Test procedure specification identifier: wp21-vo-ts01-galeb01-tps03
Test design specification reference: wp21-vo-ts01-galeb01-tds01
Test case specification reference: wp21-vo-ts01-galeb01-tcs03
Test log identifier: wp21-vo-ts01-galeb01-tl03

Purpose

This procedure executes the test case wp21-vo-ts01-galeb01-tcs03.

Procedure Steps

Installation, Log and Set Up See wp21-vo-ts01-galeb01-tps01.

Start The test is run by typing the command
`./galeb-ssh config-file input-file`
The test should be repeated for each mapping combination given in Section wp21-vo-ts01-galeb01-tcs03 of the test case specification wp21-vo-ts01-galeb01-tcs03.

Proceed During execution, scan the output for account mapping, directory information and any errors. Log into the slave node and check where and when the temporary files are created and removed.

Wrap Up and Contingencies See wp21-vo-ts01-galeb01-tps01.

3.2.13 Test log

Test log identifier: wp21-vo-ts01-galeb01-tl03

Description

Test procedure wp21-vo-ts01-galeb01-tps03 was was executed by Marjan Šterk, XLAB. The nodes used were `muca-0` and `muca-1`, based on 64-bit AMD Sempron processors and running Debian GNU/Linux 4.0 with kernel 2.6.18-5-amd64.

The procedure was executed immediately after wp21-vo-ts01-galeb01-tps02, so the versions of the software were the same.

Running the Procedure

Execution Description The procedure was run on 2007-09-26 following the procedure specification wp21-vo-ts01-galeb01-tps03.

Procedure Results Both cases with correct account mapping executed successfully. In the case with incorrect mapping the corresponding slave process failed, such that only the results of the other slaves were used.

In case when the mapping is changed to another account or removed during slave process execution, the master process is unable to retrieve the created slave results file, the effect of which is the same as if the slave had failed. The reason is that the execution of slave and `scp`-ing of the results cannot be done in the same session. Also, the temporary files cannot be deleted, which can cause sensitive information to remain uncontrolled on the remote node.

Note that this should not be considered a bug in itself, since instead of `scp` other XtreemOS components will later be used to transfer the input data and results, e.g. XtreemFS.

Anomalous Events

No anomalous events were encountered, other than those already documented in test log wp21-vo-ts01-galeb01-tl01.

3.2.14 Test design specification

Test design specification identifier: wp21-vo-ts01-jcae01-tds01
Test plan reference: wp21-vo-ts01-tp

Features to be Tested

The following features are the subject of this test design specification:

1. mapping of VO users to accounts on the VO-aware node
2. decisions on access rights and isolation between users belonging to different VOs and between those belonging to the same VO

To summarize, the fulfillment of the following requirements from D4.2.3 will be evaluated:

- full test: R25, R85, R89,
- partial test: R22, R24, R26, R28, R86, R96.

Approach Refinements

This test execute Amibe on a remote node using ssh-xos and check that user mapping is done correctly.

Test Identification

The test case for this test design specification is wp21-vo-ts01-jcae01-tcs01.

Feature Pass/Fail Criteria

The test will pass if:

- ssh-xos connection works for authorized users and is forbidden for unauthorized one.
- user mapping is correct

3.2.15 Test case specification – VO jCAE

Test case specification identifier: wp21-vo-ts01-jcae01-tcs01

Test Items

This test case evaluates:

- mapping of VO users
- ssh (XtreemOS version) access to the VO
- file isolation

Input Specifications

The tests only require a geometry file for the jCAE Mesher (Amibe) and meshing parameters. The geometry file can be created with jCAE following this tutorial: http://jcae.sourceforge.net/getting_started/index.html#Generating+a+geometry

Output Specifications

We will check that

- it is possible to connect to each nodes with ssh
- the local account matches the one specified in amappedfile
- another user cannot modify the files created by Amibe.

Environmental Needs

Hardware This test requires 2 computers (virtual machines are also fine).

Software This test run on Linux. It requires Openssl, Berkeley DB, PAM and openssh. We also need Amibe (the mesher of jCAE) and libstdc++.so.5 which is required by Java.

We also need the GNUTLS `certtool` tool to extract information from the X.509 certificates provided.

Intercase Dependencies

Running this test makes no sense if wp21-vo-ts01-galeb01-tcs01 and wp21-vo-ts01-galeb01-tcs02 fail.

3.2.16 Test procedure specification

Test procedure specification identifier: wp21-vo-ts01-jcae01-tps01
Test log identifier: wp21-vo-ts01-jcae01-tl01
Test design specification reference: wp21-vo-ts01-jcae01-tds01

Purpose

This is the procedure to execute the test wp21-vo-ts01-jcae01-tcs01.

Procedure Steps

Log A part of the log will be available on the console (ssh-xos or su user output). The sshd-xos log will be found in the `var/log` directory. The file name depends on the Linux distribution, for Debian it is `/var/log/auth.log`.

Set Up

- Follow the procedure from xtreemos-nss-pam-0.03/README (part of the source bundle of xtreemos-nss-pam).
- For xos-ssh the setup instruction can be found in the XtreemOS SVN in file `WP2.1/T2.1.3/trunk/releases/v0.71/README`.

Proceed

- Get the subject name from the X.509 certificate
- Setup a mapping for this user. We will redo the test with different mapping in their turn:

```
vo:<certificate CN>  localaccount
vo:<certificate CN>  .accountpool
```

- For each mapping, we will test the following commands:

```
getent passwd <certificate CN>@vo
su - <certificate CN>@vo
ssh-xos localhost
```

- Then for each mapping we will remotely run Amibe
- During the execution of Amibe we will log to the remote node and try to modify files

3.2.17 Test log

Test log identifier: wp21-vo-ts01-jcae01-tl01

Description

- The test was run on 2007-10-25 by Jerome Robert (EADS). The 2 computers where two 32bit VMware virtual machines running Debian Sid (Linux 2.6.22). Let call them xos1 and xos2.
- We tested the 0.71 version downloaded from directory `WP2.1/T2.1.3/trunk/releases/v0.71/` of the XtreemOS SVN. We will need the README file found at the same location.
- We used a simplified bundle of jCAE containing only Amibe: http://downloads.sourceforge.net/jcae/jcae-0.14.2-Linux.tar.bz2

Installation

To build xtreemos-nss-pam we needed to install some packages:

```
apt-get install libpam0g-dev libdb4.5-dev libssl-dev libgdbm-dev
```

When compiling ssh-xos the first time we forgot to specify the `-with-pam` configure flag. It was creating an sshd which was always refusing connection.

Execution

We use the proxy.pem file available as
`WP2.1/T2.1.3/trunk/releases/v0.7/config/proxy.pem`
in the XtreemOS SVN. Then we get the certificate subject:

```
cat proxy.pem | certtool -i | grep Subject:
        Subject: O=Grid,OU=GlobusTest,OU=simpleCA-gsmlserver,CN=
            (cont.)client,CN=1262133077
```

We define the mapping in `/etc/xos/mapping/amappedfile` on xos2. We use a local mapping:

```
xtreemos:/O=Grid/OU=GlobusTest/OU=simpleCA-gsmlserver/CN=client/CN
    (cont.)=1262133077 jerome
```

On xos1, we test both local account and account pool:

```
xtreemos:*                                  .accountpool1
xtreemos:*                                  jerome
```

Then as a first test we try to log on.

```
xos2:~# su - /O=Grid/OU=GlobusTest/OU=simpleCA-gsmlserver/CN=
    (cont.)client/CN=1262133077@xtreemos
xos_dbaux_grp: Database '/var/xos/GpasswdDB.db' open failed.: No
    (cont.)such file or directory
databases_setup: ERROR: No such file or directory
groupmapping: read_conffile fail
xos_dbaux_grp: Database '/var/xos/GpasswdDB.db' open failed.: No
    (cont.)such file or directory
databases_setup: ERROR: No such file or directory
xos_dbaux_grp: Database '/var/xos/GpasswdDB.db' open failed.: No
    (cont.)such file or directory
databases_setup: ERROR: No such file or directory
Segmentation fault
```

A workaround for this problem is to create the /var/xos directory:

```
mkdir /var/xos
```

Once this directory is created, the first access to the database leads to a segmentation fault:

```
xos2:~# getent passwd /O=Grid/OU=GlobusTest/OU=simpleCA-gsmlserver
    (cont.)/CN=client/CN=1262133077@xtreemos
No records found for the token
groupmapping: read_conffile fail
No records found for the token
Segmentation fault
xos2:~# getent passwd /O=Grid/OU=GlobusTest/OU=simpleCA-gsmlserver
    (cont.)/CN=client/CN=1262133077@xtreemos
groupmapping: found existed mapped GROUPS
No records found for the token
No records found for the token
Not found the record
No records found for the token
account3:XOS-cert:1003:0:/O=Grid/OU=GlobusTest/OU=simpleCA-
    (cont.)gsmlserver/CN=client/CN=1262133077@xtreemos@OTHER:/home/
    (cont.)account3:/bin/sh
```

Once the `$HOME/.xos/proxy.pem` file has been set up on xos2 we can try to login with ssh-xos on xos1:

```
jerome@xos2:~$ ssh-xos -p 2222 -J xtreemos xos1
channel 1: open failed: administratively prohibited: open failed
Could not chdir to home directory /home/account1: No such file or
    (cont.)directory
account1@xos1:/$
```

Now we run AMIBE. As we do not have XtreemFS in this test we work in `/tmp`:

```
jerome@xos2:~$ ssh-xos -p 2222 -J xtreemos xos1 /opt/jcae-0.14.2/
    (cont.)amibe.sh /tmp/sphere.brep /tmp 0.01 0
```

Amibe runs normally as ACCOUNT1. Next the mapping on xos1 is changed:

```
xtreemos:*                                    jerome
```

With this configuration we are mapped again on ACCOUNT1, not JEROME as expected. As the `getent passwd` output suggest that the mapping is reused from the database we delete the `/var/xos` to reinitialize it.
With the mapping the connection always fail. `getent passwd` confirm that:

```
xos1:/opt# getent passwd /O=Grid/OU=GlobusTest/OU=simpleCA-
    (cont.)gsmlserver/CN=client/CN=1262133077@xtreemos
groupmapping: found existed mapped GROUPS
No records found for the token
account mapping: one-by-one for mapping to exist user
:XOS-cert:0:0:/O=Grid/OU=GlobusTest/OU=simpleCA-gsmlserver/CN=
    (cont.)client/CN=1262133077@xtreemos@OTHER::
```

As a workaround we specify a one to one mapping:

```
xtreemos:/O=Grid/OU=GlobusTest/OU=simpleCA-gsmlserver/CN=client/CN
    (cont.)=1262133077 jerome
```

And then Amibe starts under the user `jerome`.

Incident Report Identifiers

The incident reports for this test log are wp21-vo-ts01-jcae01-tir01 and wp21-vo-ts01-jcae01-tir01.

3.2.18 Test incident report – Database initialisation

Test incident report identifier: wp21-vo-ts01-jcae01-tir01

Summary

The Berkley database is not properly initialised.

Incident Description

With a fresh installation of nss-pam-xos and the following user mapped, just do:

```
xos2:~# getent passwd /O=Grid/OU=GlobusTest/OU=simpleCA-gsmlserver
    (cont.)/CN=client/CN=1262133077@xtreemos
```

The expected output would be:

```
account3:XOS-cert:1003:0:/O=Grid/OU=GlobusTest/OU=simpleCA-
    (cont.)gsmlserver/CN=client/CN=1262133077@xtreemos@OTHER:/home/
    (cont.)account3:/bin/sh
```

But the following error occured:

```
xos_dbaux_grp: Database '/var/xos/GpasswdDB.db' open failed.: No
    (cont.)such file or directory
databases_setup: ERROR: No such file or directory
```

A workaround for this problem is create the /var/xos directory:

```
mkdir /var/xos
```

Then another bug appears:

```
xos2:~# getent passwd /O=Grid/OU=GlobusTest/OU=simpleCA-gsmlserver
    (cont.)/CN=client/CN=1262133077@xtreemos
No records found for the token
groupmapping: read_conffile fail
No records found for the token
Segmentation fault
```

Issuing the command again give a correct output.

Impact

This is not critical as there are workarounds.

3.2.19 Test incident report – VO Mapping changes

Test incident report identifier: wp21-vo-ts01-jcae01-tir02

Summary

xtreemos-nss-pam does not take the change of mapping into account.

Incident Description

Consider a grid user A and two local accounts a and b. The mapping can be either one to one or account pool. Here is how to reproduce the incident:

- Map A to a and login.

- Disconnect
- Change the mapping by mapping A to b
- Login. There A should be logged as b but xtreemos-nss-pam remember the former login and A is logged as b. To go further we delete the database. This is of course not the normal way of using xos-nss-pam.
- Login again with a cleared data base, map user A to b.

Impact

The user mapping cannot be changed without resetting the database.

3.2.20 Test summary report – VO

Test summary report identifier: wp21-vo-ts01-tsr

Summary

The testing of the node-level VO support (modules *nss-pam* (version 0.031, released on 2007-06-27), *xos-openssh* (version 0.071, released on 2007-06-27), and *ams_server* (revision 396 from the SVN, committed on 2007-06-12)) was evaluated using

- simple command-line tests in test cases wp21-vo-ts01-galeb01-tcs01 and wp21-vo-ts01-galeb01-tcs02,
- the Galeb application in test case wp21-vo-ts01-galeb01-tcs02,
- the jCAE application in test case wp21-vo-ts01-jcae01-tcs01.

The first test case focused on account mapping rules, the second one on how these rules affect isolation between users (i.e. file access permissions). The last two used the functionality tested by the first two in a real application use-case.

Comprehensiveness Assessment

The testing process was in line with the approach given in Section wp21-vo-ts01-tp.

Summary of Results

While most tests executed successfully, there were four unresolved incidents, given in the incident reports wp21-vo-ts01-galeb01-tir01, wp21-vo-ts01-galeb01-tir02, wp21-vo-ts01-jcae01-tir01, and wp21-vo-ts01-jcae01-tir02. Particularly the first one represents a fatal security hole and must be corrected as soon as possible.

Evaluation

Apart from the bugs that caused the incidents identified above, the software appears from the user perspective to be well thought-out. As expected, not all requirements are satisfied in this early version:

R22: fulfilled,

R24: partly fulfilled (local root configures the usage of the node by VO users, but the VO as such is not created or managed anywhere),

R25: partly fulfilled (will be fulfilled when the incidents identified above are resolved and VO membership of users is checked),

R26: not fulfilled (changing the account mapping can cause VO user's files to become inaccessible; could be fulfilled by other components),

R28: fulfilled,

R85: partly fulfilled (testing of group-level access and access to other objects required to assess complete fulfillment),

R86: fulfilled,

R89: not fulfilled (subject of the user's certificate proxy must be entered into `ampappedfile`),

R96: fulfilled.

Of course, the system will behave as stated in the requirements only if it is properly configured, e.g. correct mappings in `amappedfile` etc.

It should be noted that the tested version of the software produces many debug messages, which are also visible to unauthorized users and could be exploited by the potential attacker.

3.3 Evaluation of Checkpoint and Restart

The checkpoint/restart module for XtreemOS is being developed by WP 2.1. This component is based on Berkeley Lab Checkpoint/Restart (BLCR) which is a kernel module that allows for checkpointing single processes and parallel applications. More information can be found here:
http://ftg.lbl.gov/CheckpointRestart/CheckpointRestart.shtml

Additional functionality needs to be added for the gridified environment envisioned by XtreemOS and will be carried out by WP2.1. One part of the work will be so that executable code and auxiliary libraries can be included as part of the checkpoint. This will enable it to be migrated to other heterogeneous machines, which make up a part of the the grid running the same OS, but still retain its original environment. Various other issues such as name caching, dealing with open

files and allowing several different ways to be launched will have to be addressed so that XtreemOS can benefit fully from the scheduling and fault tolerant advantages that such a component can provide.

The preliminary tests detailed below may not be able to delve too deeply into the usage features of the checkpoint/restart module at this point in time, since there is still ongoing development work on the component to add some core functionalities for XtreemOS. Nevertheless, it is hoped that these first results will be of use to the developers; helping to put what is required from the component in context and also helping to pinpoint some of the key functionality that is required from the wide range of applications we have chosen to help with the experimentation and evaluation process. As the component matures, the experimentation plans will be expanded to cover the new functionality, and the dialogue with the developers in WP2.1 will be maintained throughout.

3.3.1 Test plan – CR

Test plan identifier: wp21-cr-ts01-tp

Introduction

The checkpointing and restarting mechanisms from WP2.1 (short name: wp21-cr) provides a useful functionality of being able to checkpoint a whole application. This can be used to make snapshots of the system that later can be reused in case it is necessary to restore older states of the application. Furthermore, the checkpointing enables the migration of applications by checkpointing, transferring the saved state to another node, and later restarting the specific application.

Test Items

The checkpointer is based on the Berkeley Library Checkpoint/Restart module which can be found here (latest version, as of 25-09-07, is 0.6.1):

http://ftg.lbl.gov/CheckpointRestart/CheckpointRestart.shtml

The checkpointer for XtreemOS adds functionality to BLCR so that it can save the executable and libraries as part of the checkpoint. The patches to apply these extensions can be obtained from svn:
svn+ssh://scm.gforge.inria.fr/svn/xtreemos/WP2.1/T2.1.4/patchs

Features to be Tested

The following features will be tested:

- Current ability to perform checkpointing of applications using BCLR.
- Experience when performing checkpointing using new XtreemOS-specific functionality.

Features not to be Tested

Restarting checkpointed application on other nodes will not be tested at this stage since the current priority is only to establish how different applications react when being checkpointed/restarted. The next phase of experiments will investigate migrations in more detail, assuming that any application issues that cause a problem for the module have been solved.

Approach

The aim of these first experiments is to provide early feedback to the developers on any issues that partners identify with the checkpointing and restarting of their applications.

Item Pass/Fail Criteria

The software will pass the tests if the applications can be successfully checkpointed and then restarted.

Suspension Criteria and Resumption Requirements

Failure in the installation process of the checkpointer and restarting module will mean that the test can not be continued.

Testing Tasks

The following list would be considered a typical workflow for performing the current tests:

- Installation of checkpointing and restarting module.
- Installation and setup of application.
- Running application.
- Checkpointing of an application.
- Restarting of an application.

Environmental Needs

Early testing only requires a single machine. The specific environment varies between the different tests. Thus, they are specified in the corresponding test case specifications. In general, all tests require a normal Linux system and the installed checkpointing and restarting module provided by WP2.1.

Responsibilities

Tests using the SPECweb application will be handled by BSC. SAP uses two test scenarios. In the first one, SAP uses the standard test applications provided from BLCR. In the second case, the standard SAP WebAS will be used. T6 uses DBE, a multithreaded socket based application, to examine the checkpointing/restart behaviour when dealing with remote connections.

Schedule

Initials tests will be completed before 30-11-07 and the results will be included in deliverable D4.2.4.

Risks and Contingencies

No risks are foreseen in testing this component since the BLCR component is stable. The added functionality for XtreemOS may not be stable enough yet for full testing but feedback on how applications currently react to checkpointing should be useful. Different JVM versions and JVMs produced by different companies may have very different results therefore it is proposed that support for all JVMs should not be considered; only the latest versions of the most popular.

3.3.2 Test design specification

Test design specification identifier: wp21-cr-ts01-spec01-tds01
Test plan reference: wp21-cr-ts01-tp

Features to be Tested

The testing of the checkpointing and restarting of a Java application is the objective of these tests. More specifically, the tests apply to requirements R33, R35, R37 and R39.

Approach Refinements

Since these are the first tests that are being conducted on the checkpointer, the SPECweb application will not be used at this stage. In order to get early and useful feedback we will attempt to checkpoint and restart a simple multi-threaded Java application and based on the results determine the viability of running the tests on

the SPECweb application. The results obtained will provide information that will apply to all Java applications. Two separate cases will be considered:

- Testing with the current version of BLCR (0.6.1)
- Testing with the updated version of BLCR for XtreemOS

Test Identification

The test case with the current version of BLCR is wp21-cr-ts01-spec01-tcs01 while the test case for the version for XtreemOS is wp21-cr-ts01-spec01-tcs02.

Feature Pass/Fail Criteria

If the application can be checkpointed and restarted with no noticeable differences it will have passed. Anything else will be considered as a failure.

3.3.3 Test case specification – BLCR and simple Java app

Test case specification identifier: wp21-cr-ts01-spec01-tcs01

Test Items

We will test the latest version of the BLCR checkpointer module on a simple multi-threaded Java application in this case.

Input Specifications

The input will be the running Java application.

Output Specifications

The output will be simple text, detailing the length of time needed for the application to finish, that the Java application outputs to screen upon completion.

Environmental Needs

Hardware A single x86 machine is required.

Software The following software is required:

- Linux kernel 2.6.15 or higher
- JRE 1.4.2 or higher
- BLCR 0.6.1
- a simple multi-threaded Java application that runs for a pre-determined time

3.3.4 Test procedure specification

Test procedure specification identifier: wp21-cr-ts01-spec01-tps01
Test design specification reference: wp21-cr-ts01-spec01-tds01
Test case specification reference: wp21-cr-ts01-spec01-tcs01
Test log identifier: wp21-cr-ts01-spec01-tl01

Purpose

The purpose of this test is to see how the checkpointer module will handle a multi-threaded Java application.

Procedure Steps

Set Up

- Download the latest (as of 25-09-07) BCLR module v0.6.1 from http://ftg.lbl.gov/CheckpointRestart/CheckpointDownloads.shtml.
- Untar the package and enter the installation directory.
- Configure for the version of Linux currently being used with a command such as
  `./configure -with-linux=/usr/src/linux-2.6.15`
- Perform the `make` command.
- Check that the module has been created correctly; `make insmod check`
- If so, install the module; `make install` (root password required)
- Update the LD_LIBRARY_PATH if needed:
  `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib`

Start

- Check that the BLCR modules are inserted properly. `lsmod` will display blcr, blcr_vmadump and blcr_imports if they are there.
- Run `make insmod` command from the installation directory if they are not there already (requires root).
- Start the Java application using `cr_run .` The entire command will look something like this:
  `cr_run java -jar myThreadedTest.jar`

Proceed

- While the application is running determine its PID using `ps` .
- Run the BLCR checkpoint command
  `cr_checkpoint [options] PID`
  where OPTIONS should be a Linux signal such as term or kill, and the PID used is the one previously determined.
- Check that the checkpoint file was created in the current directory. It will be called 'context.PID'.
- Restart the application from the checkpoint using the command
  `cr_restart context.PID .`
- Observe that the application finishes correctly.

Shut Down The severity of the error messages will dictate whether the test needs to be suspended.

Stop The Java application was written to stop after a predetermined time.

3.3.5 Test log

Test log identifier: wp21-cr-ts01-spec01-tl01

Description

Test case wp21-cr-ts01-spec01-tcs01 was executed according to procedure wp21-cr-ts01-spec01-tps01.

Activity and Event Entries

This test was run on the 26th November 2007 by BSC.

Execution Description wp21-cr-ts01-spec01-tps01

Procedure Results

```
Results using term signal:
# cr_checkpoint --term <PID>
# cr_restart context.<PID>
Restart failed: No such file or directory
comment: the file context.<PID> was there.

First attempt using kill signal:
# cr_checkpoint --kill <PID>
```

```
# cr_restart context.<PID>
Restart failed: Permission denied
comment: the file context.<PID> had the correct owner and access
   (cont.)permissions for the user.
```

After editing the /etc/ncsd.conf file and disabling name caching for passwd/-groups the checkpoint command with the kill signal was reran and it completed successfully.

Anomalous Events

There were some failures to restart the checkpoints.

Incident Report Identifiers

The incident report for these tests is found in wp21-cr-ts01-spec01-tir01.

3.3.6 Test incident report – Failures while restarting checkpoints

Test incident report identifier: wp21-cr-ts01-spec01-tir01

Summary

There were two different types of failures when trying to restart the checkpoints.

Incident Description

The first failure encountered happened after obtaining a checkpoint using the term command. The error message when attempting to restart that checkpoint says that there is no such file or directory, even though the checkpoint appears to have been created successfully. Further investigation using dmesg shows the following errors:

```
vmadump: open('/tmp/hsperfdata\_khogan/15956', 0x2) failed: -2
vmadump: mmap failed: /tmp/hsperfdata\_khogan/15956
```

The /tmp files referred to by vmadump were not present on the system, so it appears that the term signal caused these to be removed when the system was being shutdown.

The next failure happened after obtaining a checkpoint using the kill command. The error message says that permission is denied, even though the checkpoint has the correct permissions for the current user. dmesg shows the following errors:

```
vmadump: open('/var/run/nscd/passwd', 0x0) failed: -13
vmadump: mmap failed: /var/run/nscd/passwd
```

NCSD referred to by dmesg is the name caching daemon installed on the test machine and appears to be causing a conflict. It has been confirmed that the conflict was due to NCSD by disabling the service and retrying the test, which completed successfully.

Impact

The term signal is used by Linux to signal a shutdown to an application. It allows for a graceful shutdown by the application and removes temporary files which unfortunately will prevent many applications from being restarted. Therefore, this signal should not be used in the context of a checkpoint/restart operation.

The kill signal works, but we have identified that there is an issue with name caching. This needs to be addressed in some way, either name caching functionality will have to be removed on the system or an other workaround will have to be sought.

3.3.7 Test case specification – XOS CR and simple Java app

Test case specification identifier: wp21-cr-ts01-spec01-tcs02

Test Items

We will test the BLCR checkpointer module with the added XtreemOS functionality on a simple multi-threaded Java application in this case.

Input Specifications

The input will be the running Java application.

Output Specifications

The output will be simple text, detailing the length of time needed for the application to finish, that the Java application outputs to screen upon completion.

Environmental Needs

Hardware A single x86 machine is required.

Software The following software was used:

- Linux kernel 2.6.15 or higher
- JRE 1.4.2 or above
- BLCR 0.6.1
- patch for BLCR module with XtreemOS extensions
- simple multi-threaded Java application that runs for a pre-determined time

3.3.8 Test procedure specification

Test procedure specification identifier: wp21-cr-ts01-spec01-tps02
Test design specification reference: wp21-cr-ts01-spec01-tds01
Test case specification reference: wp21-cr-ts01-spec01-tcs02
Test log identifier: wp21-cr-ts01-spec01-tl02

Purpose

The purpose of this test is to see how the BLCR checkpointer module with Xtreem-OS extensions handles a multi-threaded Java application.

Procedure Steps

Log

Set Up

- Download the latest (as of 25-09-07) BCLR module v0.6.1 from here: http://ftg.lbl.gov/CheckpointRestart/CheckpointDownloads.shtml
- Untar the package and enter the installation directory.
- Patch BLCR with the XtreemOS extensions:
  Patch available from directory svn+ssh://scm.gforge.inria.fr/svn/xtreemos/WP2.1/T2.1.4/patchs.
  Execute command: `patch -p1 <PSE_v3.patch`
- Configure for the version of Linux currently being used with a command such as `./configure -with-linux=/usr/src/linux-2.6.15`
- Perform the `make` command.
- Check that the module has been created correctly; `make insmod check`
- If so, install the module; `make install` (root password required)
- Update the LD_LIBRARY_PATH if needed;
  `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib`

Start

- Check that the BLCR modules are inserted properly. `lsmod` will display blcr, blcr_vmadump and blcr_imports if they are there.
- Run `make insmod` command from the installation directory if they are not there already (requires root).
- Start the Java application using
  tt cr_run . The entire command will look something like this: `cr_run java -jar myThreadedTest.jar`

Proceed

- While the application is running determine its pid using
  tt ps .
- Run the BLCR checkpoint command
  `cr_checkpoint [options] PID`
  where OPTIONS could be a Linux signal such as the term or kill signal, and the PID used is the one previously determined.
- Check that the checkpoint file was created in the current directory. It will be called 'context.PID'.
- Restart the application from the checkpoint using the command
  `cr_restart context.PID .`
- Observe that the application finishes correctly.

Shut Down The severity of the error messages will dictate whether the test needs to be suspended.

Stop The Java application was written to stop after a predetermined time.

3.3.9 Test log

Test log identifier: wp21-cr-ts01-spec01-tl02

Description

Test case wp21-cr-ts01-spec01-tcs02 was executed according to procedure wp21-cr-ts01-spec01-tps02.

Activity and Event Entries

This test was attempted on the 26th November 2007 by BSC.

Execution Description The steps that were followed for this test are referred to in wp21-cr-ts01-spec01-tps02. We were only able to get as far as the make command when we encountered errors.

```
/home/khogan/blcr-0.6.1\_patched/cr\_module/kbuild/cr\_io.c: In
    (cont.)function 'sys\_mkdir\_2':
/home/khogan/blcr-0.6.1\_patched/cr\_module/kbuild/cr\_io.c:758:
    (cont.)warning: implicit declaration of function 'mutex\_unlock'
/home/khogan/blcr-0.6.1\_patched/cr\_module/kbuild/cr\_io.c:758:
    (cont.)error: structure has no member named 'i\_mutex'
```

Procedure Results Unable to continue with procedure due to failure in building the module.

Environmental Information The test was being conducted on the SUSE-based Novell Linux 9 using Linux kernel 2.6.15.

Anomalous Events

The module failed to build using v3 of the XtreemOS patch.

Incident Report Identifiers

The incident report for this is found in wp21-cr-ts01-spec01-tir02

3.3.10 Test incident report – Problems installing BLCR with XtreemOS extensions

Test incident report identifier: wp21-cr-ts01-spec01-tir02

Summary

Failure to compile BLCR version 0.6.1 module after v3 of XtreemOS patch applied.

Incident Description

After applying the patch, running ./configure and then running make presents the following error:

```
/home/khogan/blcr-0.6.1\_patched/cr\_module/kbuild/cr\_io.c: In
   (cont.)function 'sys\_mkdir\_2':
/home/khogan/blcr-0.6.1\_patched/cr\_module/kbuild/cr\_io.c:758:
   (cont.)warning: implicit declaration of function 'mutex\_unlock'
/home/khogan/blcr-0.6.1\_patched/cr\_module/kbuild/cr\_io.c:758:
   (cont.)error: structure has no member named 'i\_mutex'
```

Impact

Could not continue with test. The developer has confirmed that they will update the patch so that the test can be retried.

3.3.11 Test design specification

Test design specification identifier: wp21-cr-ts01-webas01-tds01
Test plan reference: wp21-cr-ts01-tp

Features to be Tested

The tests for XtreemOS checkpointing and restarting mechanisms are focusing on the following features:

1. checkpointing of a running application. The corresponding test cases are wp21-cr-ts01-webas01-tcs01 and wp21-cr-ts01-webas01-tcs02. Furthermore, these tests address the requirements R35, R37, and R39.

2. restarting the application from the previously stored checkpoint. The corresponding test cases are wp21-cr-ts01-webas01-tcs01 and wp21-cr-ts01-webas01-tcs02. Furthermore, these tests address the requirements R35, R37, R39, and R33.

To summarize, these tests will evaluate the fulfillment of the following requirements from D4.2.3:

1. full tests: R37.

2. partial tests: R35, R39, and R33.

Approach Refinements

The initial tests in wp21-cr-ts01-webas01-tcs01 are not focused on the SAP WebAS but on two simple applications. The first application provided by the developers of the checkpointing and restarting mechanism is a simple counter on the command line. The second simple application is a standard Linux editor *gedit*. Using these two simple programs should result in a first evaluation of the underlying concept. The tests with the SAP WebAS are performed in wp21-cr-ts01-webas01-tcs02.

As described, the first scenario consists of two tests with relatively simple applications. The first application is a counter on the console which counts from 1 to 100. Our test will checkpoint the running applications. Later we will restart the same application and check whether the program continues to count from the previous number. The second simple test is to see if an opened *gedit* document, with text ('qwert') can be be restarted from checkpoint. This test serves as the first test with some GUI components and file access.

The more complex scenario of checkpointing and restarting a running SAP WebAS is executed last. Here, we can evaluate whether the mechanism can handle open data base connections, static and dynamic libraries and several threads.

Test Identification

The test case wp21-cr-ts01-webas01-tcs01 will perform the checkpointing and restarting tests using the simple counter program and the *gedit* editor. The test case wp21-cr-ts01-webas01-tcs02 uses a running SAP WebAS.

Feature Pass/Fail Criteria

The two individual features, the checkpointing and the restarting of an application cannot be tested independently as we do not have any evaluation tool for the checkpointing than the restart mechanism. Thus, from the application perspective it is only possible to evaluate the combined behavior. The tests are successful if the applications can be checkpointed and restarted with no noticeable differences. Anything else will be considered as a failure and reported back to WP2.1.

In detail, the counting example passes, when the program counts all remaining numbers. The editor example passes when it is still possible to insert some text and to save it as usual. The SAP WebAS example passes the test if it is possible to checkpoint it, to restart it and that users can still access the SAP WebAS and modify individual data.

3.3.12 Test case specification – CR with Simple Applications

Test case specification identifier: wp21-cr-ts01-webas01-tcs01

Test Items

We will test the checkpointing and restart module of WP2.1. To this end, we will first use two simple examples. The first counting example is provided by BLCR. The second example is the simple Linux editor *gedit*.

Input Specifications

The input will be on the one hand the simple counting example from BLCR. The second input will be the gedit application.

The counting program does not require any input as this application simply counts from 1 to 100. The input for the *gedit* application is ”qwert”.

Output Specifications

The output for the counting program is that the remaining numbers up to 100 are correctly shown.

After restarting the editor *gedit*, the content ”qwert” should be displayed. Furthermore, it must be possible to modify the text and save it properly.

Environmental Needs

Hardware A single x86 machine is required.

Software The following software is required:

- Linux kernel 2.6.16 or higher

- BLCR 0.6.1 with the XtreemOS modifications
- simple counting program provided by BLCR and a simple text editor *gedit*.

3.3.13 Test procedure specification

Test procedure specification identifier: wp21-cr-ts01-webas01-tps01
Test design specification reference: wp21-cr-ts01-webas01-tds01
Test case specification reference: wp21-cr-ts01-webas01-tcs01
Test log identifier: wp21-cr-ts01-webas01-tl01

Purpose

The purpose of these tests is to see how the checkpointing and restart module will handle a simple counting program as well as a simple text editor *gedit*.

Procedure Steps

Installation The test requires a proper Linux system without further modifications. The setup of the checkpointing and restarting functionalities is described under setup.

Log The output of the counting program is logged into a file using the command line. The output of the editor *gedit* is simply described.

Set Up The setup is as follows.

```
Install BLCR:
    svn+ssh://<login>@scm.gforge.inria.fr/~
    svn/xtreemos/WP2.1/T2/1.4/

 # cd blcr-<VERSION>
 # chmod 777 autogen.sh
 # ./autogen.sh
 # mkdir builddir
 # cd builddir
 #../configure
 # make
 # make install

Load BLCR modules

# /sbin/insmod
/usr/local/lib/blcr/2.6.12-1.234/blcr_imports.ko
# /sbin/insmod
/usr/local/lib/blcr/2.6.12-1.234/blcr_vmadump.ko
# /sbin/insmod
/usr/local/lib/blcr/2.6.12-1.234/blcr.ko
```

```
Check that modules have been loaded correctly

# lsmod | grep blcr

Should produce: (if not, try reloading modules again)

blcr                   67728  0
blcr_vmadump           37528  1 blcr
blcr_imports           25856  2 blcr,blcr_vmadump
```

Start For the counting example:

```
(Open new shell - Shell A)
# cd blcr-0.5.3/builddir/counting
# cr_run  ./counting

Counting demo starting with <PID>
Count = 0
Count = 1
Count = 2
Count = 3

(Open new shell - Shell B)

# cr_checkpoint -term <PID>
```

For the simple text editor *gedit*:

```
#cr_run ./gedit
```

Proceed For the counting example:

```
(In Shell A)
...
Count = 3
Terminated

#cr_restart context.<PID>

Count = 4
Count = 5
Count = 6
Count = 7
```

For the text editor example, we have four different options for a checkpointing. In general, these options are also valid for the counting example but not necessary.

```
Option 1:
#cr_checkpoint -kill <PID>
```

```
#cr_restart context.<PID>

Option 2:
#cr_checkpoint --save-all --kill <PID>
#cr_restart context.<PID>

Option 3:
#cr_checkpoint --save-exe --kill <PID>
#cr_restart context.<PID>

Option 4:
# cr_checkpoint --kill --save-private --save-shared <PID>
# cr_restart context.<PID>
```

Measure The two tests do not require special measurement setups.

Shut Down All tests can be shut down by using the process kill options provided by Linux.

Restart A restart is in both cases only meaningful from the beginning. Thus, restart in this case means performing the test again right from the beginning.

Stop The counting example stops after having counted to 100. The text editor has to be stopped manually from the corresponding menu.

Wrap Up Both tests do not require an explicit wrap up. However, it is meaningful to remove the stored checkpoints after the test in order to keep the hard disc clean.

Contingencies In case of anomalies, the applications should simply be restarted.

3.3.14 Test log

Test log identifier: wp21-cr-ts01-webas01-tl01

Description

The tests were performed by SAP. Furthermore, the requirements on the setup were rather small such that only a few MB memory was required and no special hardware at all.

Activity and Event Entries

All tests were performed on the 10th and 28th November 2007 by SAP.

Execution Description All tests were performed according to the test setup. Thus, a standard SuSE Linux Enterprise 10.3 was used. Thereafter, the checkpointing and restarting mechanism provided by WP2.1 was installed and finally, the tests executed.

Procedure Results For the counting problem we have received a valid and correct output:

```
(Open new shell - Shell A)
# cd blcr-0.5.3/builddir/counting
# cr_run   ./counting

Counting demo starting with pid 11539
Count = 0
Count = 1
Count = 2
Count = 3

(Open new shell - Shell B)

# cr_checkpoint -term 11539

(In Shell A)

...
Count = 3
Terminated

#cr_restart context.11539

Count = 4
Count = 5
Count = 6
Count = 7
```

For the text editor test, we have received an unusual result for all four options:

```
Option 1:
# cr_checkpoint --kill <PID>
# cr_restart context.<PID>
Restart failed: No such file or directory
comment: the file context.<PID> was there.

Option 2:
# cr_checkpoint --save-all --kill <PID>
# cr_restart context.<PID>
Restart failed: No such file or directory
comment: the file context.<PID> was there.

Option 3:
#  cr_checkpoint --save-exe --kill <PID>
# cr_restart context.<PID>
Restart failed: No such file or directory
```

```
comment: the file context.<PID> was there.

Option 4:
# cr_checkpoint --kill --save-private --save-shared <PID>
# cr_restart context.<PID>
Restart failed: No such file or directory
comment: the file context.<PID> was there.
```

Therefore, we conclude that the checkpointing and restarting mechanism does not work properly for the text editor *gedit*.

Environmental Information We cannot associate special environmental information. The system setup was stable and did not change during our experiments.

Anomalous Events

The text editor example failed to restart. However, we cannot conclude for the reason. This might be hidden in the generation of the checkpoint or in the restart mechanism.

Incident Report Identifiers

The incident report can be found under the identifier: wp21-cr-ts01-webas01-tir01.

3.3.15 Test incident report – Text editor does not restart

Test incident report identifier: wp21-cr-ts01-webas01-tir01

Summary

The simple text editor *gedit* could not be restarted from the checkpoint.

Incident Description

After a checkpointing of the text editor *gedit*, the editor could not be restarted correctly. The output on the command line gives the impression that the checkpointing itself was successful. However, the failed restart might result from a checkpoint that was not generated correctly. Thus, it is the task of the development groups to find the real reason.

Impact

The text editor could not be restarted from the checkpoint. Thus, the application could not be used at all after the checkpointing.

3.3.16 Test case specification – CR with WEBAS

Test case specification identifier: wp21-cr-ts01-webas01-tcs02

Test Items

This test will evaluate the checkpointing and restarting mechanism provided by WP2.1. To this end, the SAP WebAS will be used as an example for a complex and large business application.

Input Specifications

The application will not have specific input. Aim of the test is to see whether a checkpointing can be performed and the application restarted. Input are some random user clicks to find out whether the application is active.

Output Specifications

There is no specific output to the system. Just that some tabs can be chosen by users or that some entries can be filled with text. It is more a kind of test whether the system is active at all.

Environmental Needs

Hardware A single x86_64 machine with at least 4 GB of RAM and 150 GB of available disc space is required.

Software The following software is required:

- Linux kernel 2.6.16 or higher
- BLCR 0.6.1 with the XtreemOS modifications
- SAP WebAS 6.40

3.3.17 Test procedure specification

Test procedure specification identifier: wp21-cr-ts01-webas01-tps02
Test design specification reference: wp21-cr-ts01-webas01-tds01
Test case specification reference: wp21-cr-ts01-webas01-tcs02
Test log identifier: wp21-cr-ts01-webas01-tl02

Purpose

The purpose of this test is to evaluate whether the checkpointing and restarting mechanism perform correctly by using the SAP WebAS as the test application.

Procedure Steps

Installation The test requires a proper Linux system without further modifications. Furthermore, the standard SAP WebAS 6.40 has to be installed on this Linux machine. The setup of the checkpointing and restarting functionalities is described under setup.

Log The startup output of the SAP WebAS is available on the command line. If the system is up an running, the graphical user interfaces of the SAP WebAS can be used to evaluate whether the system behaves correctly.

Set Up First, the checkpointing and restarting mechanism needs to be installed. This is done as described for wp21-cr-ts01-webas01-tps01. Therefore, we do not repeat these steps here. The SAP WebAS is installed following the standard SAP procedure. As this is a rather long description and specific to SAP, we do not describe this process here.

Start The start of the WebAS is simple.

```
#su benadm
# whoami
benadm
# startsap
```

Proceed The proceed process consists of using the generated checkpoint and then restarting the application.

Measure The measurement is the check whether the system response is normal during the start up as well as after the checkpointing and restarting of the SAP WebAS.

Thus, the start up should be as follows:

```
#su benadm
# whoami
benadm
# startsap

Checking BEN Database
------------------------------
 ABAP Database is not available via R3trans

Checking BEN Database
------------------------------

Starting SAP-Collector Daemon
------------------------------
16:08:12 08.11.2007    LOG: Effective User Id is root
```

```
************************************************
* This is Saposcol Version COLL 20.93 640 - AMD/Intel
x86_64 with Linux, PL:159, Nov 23 2006
* Usage:  saposcol -l: Start OS Collector
*         saposcol -k: Stop  OS Collector
*         saposcol -d: OS Collector Dialog Mode
*         saposcol -s: OS Collector Status
* The OS Collector (PID 5980) is already running .....
************************************************
 saposcol already running
 Running /usr/sap/BEN/SYS/exe/run/startdb
Trying to start database ...
Log file: /usr/sap/BEN/benadm/startdb.log
BEN database started
/usr/sap/BEN/SYS/exe/run/startdb completed successfully

Checking BEN Database
-------------------------------
 ABAP Database is running

Starting SAP Instance DVEBMGS19
-------------------------------
 Startup-Log is written to /usr/sap/BEN/benadm/startsap_
DVEBMGS19.log
 Instance on host bfssrv05 started
```

The user should be able to login to the system as shown in Figures 3.1 and 3.2.

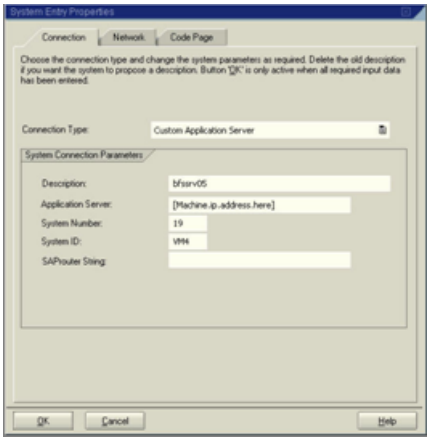


Figure 3.1: Login for the SAP WebAS.

If the SAP WebAS cannot be started correctly, the output is shown in Figure 3.3.

Shut Down The test can be shut down by using the "Exit" entry in the GUI menu. In case that the system is not responding at all, the process kill options provided by Linux can be used. However, this might cause inconsistent data.

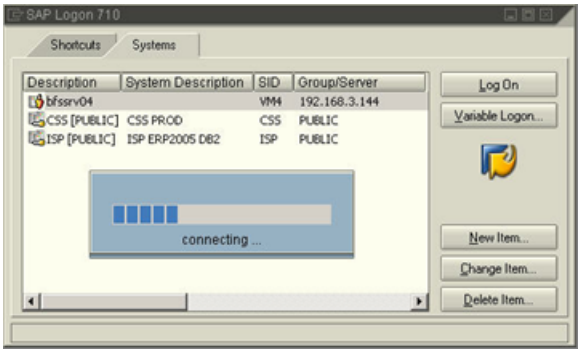


Figure 3.2: SAP Logon (used to decide to which WebAS to connect).

Figure 3.3: SAP message if the startup fails.

Restart A restart is only meaningful from the beginning. Thus, restart in this case means performing the test again right from the beginning.

Stop The WebAS can be stopped by using the ”Exit” entry in the menu. In all other cases, the WebAS can be killed from the command line.

Wrap Up The test does not require an explicit wrap up. However, it is meaningful to remove the stored checkpoints after the test in order to keep the hard disc clean.

Contingencies In case of anomalies, the applications should simply be restarted.

3.3.18 Test log

Test log identifier: wp21-cr-ts01-webas01-tl02

Description

The tests were performed by SAP using the SAP WebAS. The tests required a normal x86 machine with at least 4 GB of memory and 150 GB of hard disc space.

Activity and Event Entries

All tests were performed on the 23rd November 2007 by SAP.

Execution Description The tests were performed according to the test setup. Thus, a standard SuSE Linux Enterprise 10.3 was used. Thereafter, the checkpointing and restarting mechanism provided by WP2.1 was installed. Furthermore, the SAP WebAS was installed on the corresponding single machine. Finally, the tests were executed on the prepared machine.

Procedure Results For the SAP WebAS, we have received an invalid result:

```
# su benadm
% cr_run startsap

Checking BEN Database
_____________________

 ABAP Database is not available via R3trans

Checking BEN Database
_____________________


Starting SAP-Collector Daemon
_____________________________

ERROR: ld.so: object 'libcr.so.0' from LD_PRELOAD
cannot be preloaded: ignored.
ERROR: ld.so: object 'libpthread.so.0' from
LD_PRELOAD cannot be preloaded: ignored.
17:34:24 08.11.2007   LOG: Effective User Id is root
************************************************
* This is Saposcol Version COLL 20.93 640 -
AMD/Intel x86_64 with Linux, PL:159, Nov 23 2006
* Usage:  saposcol -l: Start OS Collector
*         saposcol -k: Stop  OS Collector
*         saposcol -d: OS Collector Dialog Mode
*         saposcol -s: OS Collector Status
* The OS Collector (PID 4650) is already running .....
************************************************
 saposcol already running
 Running /usr/sap/BEN/SYS/exe/run/startdb
Trying to start database ...
Log file: /usr/sap/BEN/benadm/startdb.log
ERROR: ld.so: object 'libcr.so.0' from LD_PRELOAD
cannot be preloaded: ignored.
ERROR: ld.so: object 'libpthread.so.0' from LD_PRELOAD
cannot be preloaded: ignored.
ERROR: ld.so: object 'libcr.so.0' from LD_PRELOAD
cannot be preloaded: ignored.
ERROR: ld.so: object 'libpthread.so.0' from LD_PRELOAD
cannot be preloaded: ignored.
ERROR: ld.so: object 'libcr.so.0' from LD_PRELOAD
cannot be preloaded: ignored.
ERROR: ld.so: object 'libpthread.so.0' from LD_PRELOAD
cannot be preloaded: ignored.
BEN database started
/usr/sap/BEN/SYS/exe/run/startdb completed successfully
```

```
Checking BEN Database
------------------------------
ABAP Database is running

Starting SAP Instance DVEBMGS19
------------------------------
Startup-Log is written to /usr/sap/BEN/benadm/startsap
DVEBMGS19.log
Instance on host bfssrv05 started
```

Obviously, some dynamic libraries could not be loaded using the BLCR approach. Therefore, the SAP WebAS cannot operate in the usual way. Thus, we did not reach the stage where we can use the checkpointing functionality itself.

Environmental Information We cannot associate special environmental information. The system setup was stable and did not change during our experiments.

Anomalous Events

The SAP WebAS application failed to start correctly. Thus, no further experiments were possible.

Incident Report Identifiers

The incident report can be found under the identifier: wp21-cr-ts01-webas01-tir02.

3.3.19 Test incident report – WEBAS does not start with CR

Test incident report identifier: wp21-cr-ts01-webas01-tir02

Summary

The SAP WebAS cannot be started normally using the checkpointing and restarting procedure. Thus, the checkpointing and restarting mechanism could not be evaluated.

Incident Description

The SAP WebAS could not be started using the provided command from the checkpointing and restarting mechanism. The corresponding output provides the information that some dynamic libraries could not be handled properly. Thus, the testing of the checkpointing and restarting mechanism itself was not performed.

Impact

The SAP WebAS could not be started correctly. This leads to the fact that the SAP WebAS cannot be used at all in combination with the currently provided checkpointing and restarting functionality.

3.3.20 Test design specification

Test design specification identifier: wp21-cr-ts01-dbe-tds01
Test plan reference: wp21-cr-ts01-tp

Features to be Tested

In order to test the checkpointing and restart we conduct two tests:

1. checkpointing a basic Java application with a single thread and no socket connections. After killing the application we will try to restart it again in another machine and context.

2. checkpointing a multi-thread application with several threads running at the same time and several connections. After killing the application, we will try to restart it again on another machine and environment (memory has been freed, connections have been closed, etc).

Approach Refinements

The initial tests in wp21-cr-ts01-dbe-tcs01 are not focused on the DBE application, that will be used in the second test, but on a more simple Java application.

This first test consists of a standalone application that does not open Input/Output connections, but stores local variables that change time.

Test Identification

The test case wp21-cr-ts01-dbe-tcs01 will perform the checkpointing and restarting tests using a simple auto-increment program. The test case wp21-cr-ts01-dbe-tcs02 uses the DBE application

Feature Pass/Fail Criteria

The tests will pass if, the application ca be successfully checkpointed and restarted.

In detail, the auto-increment application passes if it maintains the last number and can follow the normal process of auto-increment numbers. The second test passes if after the restart, the DBE continues normally even if it lost ongoing transactions.

3.3.21 Test case specification – Checkpointing a simple Java file-writing application

Test case specification identifier: wp21-cr-ts01-dbe-tcs01
The main purpose of this test is to know which operations can be checkpointed and restarted and which ones can not. For instance, the dbe application is a very intensive application in terms of files, threads and sockets. It also uses advanced Java features such as dynamic class loading. We will test progressively if these main features can be checkpointed and restarted easily or if them present problems while restarting.

Test Items

In an infinite loop this application opens two files stored in the same directory where the application is and writes a string on them. The file names are passed as an argument to the program.

The string to be written in each file is a number that increments its value in each iteration. This number is stored in a static variable in the class. Its initial value is 1.

Input Specifications

The input of the application are the names of the files we want to write to.

Output Specifications

Two files with a list of numbers form 1 to X. One number is written each second.

Environmental Needs

Hardware A single x86 machine is required.

Software The following software was used:

- Linux kernel 2.6.14-generic
- JRE 1.5.0_10 from Sun
- BLCR 0.6.1

3.3.22 Test procedure specification

Test procedure specification identifier: wp21-cr-ts01-dbe-tps01
Test log reference: wp21-cr-ts01-dbe-tl01
Test design specification reference: wp21-cr-ts01-dbe-tds01

Purpose

The purpose of the test is to see if the checkpointing and restart is able to handle simple Java applications

Procedure Steps

Set Up

- Download the latest BLCR module v0.6.1 from:
  http://ftg.lbl.gov/CheckpointRestart/CheckpointDownloads.shtml
- Configure and make the module.
- Install the module

Start

- Check that the BLCR modules are inserted properly. `lsmod` will display blcr, blcr_vmadump and blcr_imports if they are there.
- Run `make insmod` command from the installation directory if they are not there already (requires root).
- Start the Java application using `cr_run` . The entire command will look something like this:
  `cr_run java -jar fileTest.jar file1.txt file2.txt`

Proceed

- While the application is running determine its pid using `ps` .
- Run the BLCR checkpoint command
  `cr_checkpoint [options] PID`
- Kill the application using SIGKILL instead of SIGTERM, that should say to the virtual machine that the application exited correctly.
- Restart the application from the checkpoint using the command
  `cr_restart context.PID .`
- Observe that the application finishes correctly.

Measure The test will succeed if after killing the application, it can restart from the checkpointed status. The applications is writing an auto-incremented number each second. If we checkpoint at second 4, and kill the application at second 10, we expect that the application restart showing 5 in the screen an not starting from the beginning of continuing from 10.

Shut Down We will kill the application with the -9 parameter

Restart We will execute the restart procedure

Stop The Java application never stops

Contingencies There where not contingencies.

3.3.23 Test log

Test log identifier: wp21-cr-ts01-dbe-tl01

Description

Tests where preformed by T6. No more hardware than a computer was needed

Activity and Event Entries

All tests were performed on the 8 November 2007 by T6.

Execution Description see wp21-cr-ts01-dbe-tps01

Procedure Results The test was executed successfully. The program was restarted after a KILL signal, following the execution from the status saved in the checkpoint file.

This is the output of the application while checkpointing: writing 1 writing 2 writing 3 writing 4 (we checkpoint here) writing 5 writing 6 (we kill here) killed

After restarting, we can see the output writing 5 writing 6 writing 7 writing 8 ...

Output files involved in the test contain the correct list of numbers form 1 to n.

Environmental Information The test was executed in a laptop computer running Linux (kernel version 2.6.22-14)

Anomalous Events

There were not anomalous events. The program restarted and the executed continued from the checkpointing.

Incident Report Identifiers

There were not incidents, the application could restart as expected from tho checkpointed status.

3.3.24 Test case specification – Checkpointing a multithreaded Java application which uses sockets

Test case specification identifier: wp21-cr-ts01-dbe-tcs02

Test Items

We will test the DBE application. It is a multithreaded application that uses both, client and server sockets. Server sockets remain open and listening while the application is running. This could represent a problem when restarting, cause the kernel is controlling the open resources.

Input Specifications

The usual DBE start command, that includes the properties files where third party libraries are located

Output Specifications

The output is an infinite loop running application that waits for remote connections and processes them.

Environmental Needs

Hardware A single x86 machine is required.

Software The following software was used:

- Linux kernel 2.6.14-generic
- JRE 1.5.0_10 from Sun
- BLCR 0.6.1

3.3.25 Test procedure specification

Test procedure specification identifier: wp21-cr-ts01-dbe-tps02 Checkpointing multi-threaded Java application with sockets
Test log reference: wp21-cr-ts01-dbe-tl02
Test design specification reference: wp21-cr-ts01-dbe-tds01

Purpose

The purpose of this is to see how the checkpointer module will handle a multithreaded Java application with server and client sockets open.

Procedure Steps

We start the application in the standard way and, after few seconds we checkpoint it. We wait some more seconds and we kill it. After a sort period of time, enough to allow the kernel to remove references to sockets that have been closed or destroyed, we try to restart the application from the checkpointed status.

Log The multithreaded application starts and most of the threads continue running from the checkpointing status. Some of them fail (controlled exceptions) because they cannot read/write in their assigned sockets.

The application could restart fine if, after this error, the server socket is open again. Unfortunately, this kind of error in a server is not expected, and the exception is simply reported, and the socket is not reopened of rebinded again.

Set Up

- Download the latest BLCR module v0.6.1 from:
  http://ftg.lbl.gov/CheckpointRestart/CheckpointDownloads.shtml
- Configure and make the module.
- Install the module

Start

- Check that the BLCR modules are inserted properly. `lsmod` will display blcr, blcr_vmadump and blcr_imports if they are there.
- Run `make insmod` command from the installation directory if they are not there already (requires root).
- Start the Java application using `cr_run` . The entire command will look something like this:
  `cr_run bin/run.sh`

Proceed

- While the application is running determine its pid using `ps` .
- Run the BLCR checkpoint command
  `cr_checkpoint [options] PID`
  where OPTIONS could be a Linux signal such as the term or kill signal, and the PID used is the one previously determined.
- Check that the checkpoint file was created in the current directory. It will have be called 'context.PID'.

- Restart the application from the checkpoint using the command `cr_restart context.PID .`
- Observe that the application finishes correctly.

Measure

Shut Down The severity of the error messages will dictate whether the test needs to be suspended.

Restart We restart the application from the checkpointed status using the standard process, without a preparation task.

Stop The Java application will stop manually by pressing CTRL and C form the shell.

Wrap Up We can never reproduce the same environment as when the application was running, because it contains references to kernel identifiers that have been deleted by the kernel. Those identifiers now means nothing for the kernel and no resource is associated to them. Read and write operations will fail.

Contingencies Knowing this issue we can write applications that deal with it and allow the Java application to recover from such failures after a checkpointing. As explained before, one of these actions can be to reopen and rebind server sockets when an IOException is thrown. Client sockets (threads talking with another server) need not be re-open, because it is probably that the remote server.

3.3.26 Test log

Test log identifier: wp21-cr-ts01-dbe-tl02

Description

Tests where performed by T6. No more hardware than a linux computer was needed.

Activity and Event Entries

All tests were performed on the 8 November 2007 by T6.

Execution Description see wp21-cr-ts01-dbe-tps02

Procedure Results The test failed. Even when the application was able to restart, communication sockets where not valid anymore, and the communication with the clients was lost.

Anomalous Events

The server socket cannot be opened when the application is restarted.

Incident Report Identifiers

wp21-cr-ts01-dbe-tir02

3.3.27 Test incident report – Problem with open sockets

Test incident report identifier: wp21-cr-ts01-dbe-tir02

Summary

The application can restart correctly, but a set of errors related to server sockets are produced.

Incident Description

When the application restarts, the server socket cannot be opened (probably an identifier deleted by the kernel).

The error message is related to Java classes, and it indicates that the socket identifier is not available anymore.

Impact

The main application would have be changed in order to cope with this issue. The failure in open server sockets would have to be controlled by the application, re-opening it in case the binding fails.

3.3.28 Test summary report – CR

Test summary report identifier: wp21-cr-ts01-tsr

Summary

Testing was performed using an early version of the kernel module modified for XtreemOS (blcr-0.6.pre0_snapshot_2007_07_12), the latest version of the original module (BLCR 0.6.1) and a XtreemOS patched version of the same.

- checkpointing and restart of native processes was evaluated in wp21-cr-ts01-webas01-tcs01,

- wp21-cr-ts01-spec01-tcs01 and wp21-cr-ts01-spec01-tcs02 tested the modules ability to checkpoint/restart Java applications using the original module and the one modified for XtreemOS respectively,
- the modules ability to checkpoint/restart a complex real-world application was tested using SAP WebAS in wp21-cr-ts01-webas01-tcs02
- the modules ability to handle a multi-threaded Java application which relies on sockets was tested in wp21-cr-ts01-dbe-tcs02

Summary of Results

At this early stage of the development, it is no surprise to find that there were issues of robustness and problems trying to execute some of the tests. For this reason, complex scenarios were not envisioned at this stage of the evaluation process and only basic use case scenarios were attempted. Even still, problems were encountered with those; although it may not be fair to read too much into these early findings, since some of the required functionality will almost certainly need features from the other work packages to be fulfilled completely.

Evaluation

As is to be expected, not all requirements are fulfilled in this early version:

R32: could not be tested yet,

R33: not fulfilled, see wp21-cr-ts01-spec01-tcs01, wp21-cr-ts01-webas01-tcs01, and wp21-cr-ts01-webas01-tcs02

R34: could not be tested yet

R35: partially fulfilled

R36: could not be tested yet

R37: fulfilled

R38: could not be tested yet

R39: not fulfilled, problems with IPC reported in wp22-fm-ts01-galeb01-tir02, problems with NSCD reported in wp21-cr-ts01-spec01-tl01 and problems using sockets reported in wp21-cr-ts01-dbe-tl02. Furthermore, dynamic libraries are not supported during the startup of an application, see wp21-cr-ts01-webas01-tl02.

The main issues encountered show that using the Linux term signal when checkpointing an application often will prevent the application from restarting correctly. This is because the term signal automatically cleans any temporary files that

the process was using. The kill signal does not do any clean up and will allow the applications to restart correctly however, in the case of migration, any temporary files or environment settings will need to be accessible from the new node. When the distributed file system is incorporated with the checkpointing module it should be able to support this easily. Restoring sockets from a checkpoint has also been identified as a potential problem.

3.4 Evaluation of Federation Management (LinuxSSI)

3.4.1 Test plan – FM

Test plan identifier: wp22-fm-ts01-tp

Introduction

This test plan covers the software for LinuxSSI implementation. SSI functionalities include:

- scalable SSI mechanisms
- checkpoint/restart
- reconfiguration mechanisms
- high performance disk input/output operations in a cluster
- customizable scheduler

Test Items

The software to be tested is KERRIGHED. Source codes and documentation are available on the KERRIGHED web site: http://www.kerrighed.org.

Recent version of KERRIGHED are:

- version 2.1.1, released on 2007-08-22,
- version 2.1.0, released on 2007-07-05,
- development versions in the SVN archive without formal release dates.

Even if the experimentations have been started with version 2.1.0, this report will be only about version 2.1.1 and about the revisions 3073 and 3075 of the Kerrighed SVN archive. Note that version 2.1.1 has been released thanks to one bug reported by our tests on 2.1.0.

Features to be Tested

The following features will be tested:

- process migration
- checkpointing
- scheduling
- SMP support
- customizable scheduler

Features not to be Tested

Kerrighed is not yet stable enough for performance testing. So at this stage, the following features will not be tested:

- scalability
- I/O performance
- reconfiguration

Approach

The purpose of the early test is to provide feedback to developers about the implemented features. Before testing full complex application we will run small trivial programs to ensure that the system is stable enough. If such simple applications do not work, they will be a good start point for developer to debug, much easier than having to debug the kernel running large applications.

Item Pass Criteria

Kerrighed will pass a test if:

- running application under the condition defined by the test (migration enabled, number of processes, scheduling ...) does not make the kernel crash.
- an application run on Kerrighed, under the condition of the test, behave as if it was launched on a common system. It means that before running an application over Kerrighed we will run it without Kerrighed to ensure that discovered bugs are from Kerrighed, not from the application.

Testing Tasks

The test requires the following tasks:

- obtaining and building Kerrighed
- deploying Kerrighed on a cluster
- building and installing the application
- run the application with Kerrighed disabled
- enable Kerrighed
- run the application on Kerrighed

Environmental Needs

Kerrighed is designed to be run on a cluster but we will not do scalability and performance tests but only validity and stability tests. So two PCs running Linux and connected with a dedicated network will be enough. The Linux distribution must contains the required libraries to run the test applications. There is no requirements on the kernel version as the installation procedure of Kerrighed will provide it's own kernel. As Kerrighed is a kernel level software the test bed should provide a way to connect to node which is independent of the operative system. A KVM or a physical access are fine, but with ssh ones will not be able to get control back on the nodes after a kernel crash.

Responsibilities

This test plan and the included tests with Elfipole are the responsibility of EADS. The tests with Galeb are the responsibility of XLAB.

Schedule

Assuming that the software is in its early development stage, the installation, setup and learning to use is complex, requiring one or two weeks if the staff is not familiar with the procedure. Tests may be long to do as manipulation and configuration error could cause kernel panic and require to restart the whole cluster. All the planned tests should be finished in time for XtreemOS deliverable D4.2.4.

Risks and Contingencies

Apart from delays in testing caused by late delivery of new versions of the software and by unexpected installation problems, no specific risks are envisaged. Note that late delivery of new versions of the software can be a consequence of bugs discovered by these tests.

3.4.2 Test design specification

Test design specification identifier: wp22-fm-ts01-elfi01-tds01
Test plan reference: wp22-fm-ts01-tp

Features to be Tested

We will test the process migration and the scheduling. For the scheduling we will only test that, under certain circumstances, it trigger a process migration. We will not test the scheduling efficiency.

Approach Refinements

The test is to be done on the Elfipole application. But to be able to provide more accurate information in case of incident, we will check the migration with incremental complexity programs:

1. Test the migration of a program without any IO or IPC (cpuburn) to ensure that Kerrighed is properly installed and configured.
2. Run a simple MPI program with multiple processes on one node without migration to get a reference output. The comparison with this output will be the pass/fail criteria.
3. Run a very simple MPI program to test migration of IPC. Compare the output with previous run.
4. Run Elfipole without migration to get a reference output.
5. Run Elfipole with migration and compare with previous output.

All tests will be done with and without SMP. For tests without SMP we will use the latest stable version of Kerrighed (version 2.1.1). As there is currently no stable version of Kerrighed supporting SMP we will use the development version (subversion trunk, revision 3073).

Test Identification

The related test case are wp22-fm-ts01-elfi01-tcs01 (no SMP) and wp22-fm-ts01-elfi01-tcs02 (SMP).

Feature Pass/Fail Criteria

The migration will be considered as a migrated process is not corrupt and is transparent for the application. It should no crash and the application output should not be different than the output created when no process are migrated.

The scheduling will be considered as working if when the load of one node during 10 second is above 2. This is an arbitrary coarse criteria but as said above the gaol of this test is not to check the scheduling efficiency.

3.4.3 Test case specification – FM migration of IPC

Test case specification identifier: wp22-fm-ts01-elfi01-tcs01

Test Items

We will test the process migration and the scheduling of Kerrighed. For the scheduling we will only test that, under certain circumstances, it triggers a process migration. We will not test the scheduling efficiency.

We will test 2 types of IPC migration:

- socket (through mpich p4)
- shared memory (through mpich shmem)

The Elfipole user's guide and MPICH documentation will be required to run this test.

Input Specifications

To test the process migration we will run Elfipole on two case extracted from its auto-tests database. Their identifier are:

- coneSphereCoated_plan
- coneSpherePC

Output Specifications

The output will be the log of the execution on the console. We will compare the log with migration disabled and enabled.

Environmental Needs

Hardware

The test require a x86 compatible cluster.

Software

- This test require a configured Kerrighed cluster with at least 2 nodes. Kerrighed can be found on the Kerrighed web site: http://www.kerrighed.org. The version used for the test is 2.1.1.
- This test require Elfipole which is an internal EADS software not publicly available. The version of Elfipole used for this test is the CVS snapshot 20070816.
- Elfipole also require `mpich` which is available here: http://www-unix.mcs.anl.gov/mpi/mpich1/.

3.4.4 Test procedure specification

Test procedure specification identifier: wp22-fm-ts01-elfi01-tps01
Test design specification reference: wp22-fm-ts01-elfi01-tds01

Purpose

This procedure executes the test case wp22-fm-ts01-elfi01-tcs01.

Procedure Steps

Installation

- Set up the kerrighed cluster but don't start it yet.
- Set up a distributed file system for the cluster
- Install cpuburn on each nodes
- Install mpich shared memory and common (p4) versions
- Build the mpich `pi.c` example (simply do `mpicc pi.c`) for shmem and p4 versions of mpich
- Build and install Elfipole for shmem and p4 versions of mpich

Log

Logging is done to the standard output in free format.

Proceed

- Start SSI mode with `krgadm cluster start` krgadm cluster start command
- Enable migration with `krgcapset -d +CAN_MIGRATE`
- Start 4 burnMMX processes and check if some of them migrate to other nodes
- Reboot the cluster to be back to non SSI mode
- Start the pi p4 example:

```
mpirun -np 2 ./a.out 1000000000
```

- Start the pi shmem example:

```
mpirun.mpich-shmem -np 2 ./a.out 1000000000
```

- Start SSI mode with `krgadm cluster start` krgadm cluster start command
- Start the pi p4 example and compare with the output of the non SSI test
- Start the pi shmem example and compare with the output of non SSI test
- Reboot the cluster to be back to non SSI mode
- Start the p4 version of Elfipole:

```
mpirun -np 2 ./AS\_ELFIP\_FMM -p01 coneSphereCoatedAbs\_planX
    (cont.).p01
```

- Start the shmem version Elfipole:

```
mpirun.mpich-shmem -np 2 ./AS\_ELFIP\_FMM -p01
    (cont.)coneSphereCoatedAbs\_planX.p01
```

- Start SSI mode with `krgadm cluster start` krgadm cluster start command
- Start the p4 version of Elfipole and compare the output with the non SSI test
- Start the shmem version Elfipole and compare the output with the non SSI test

3.4.5 Test log

Test log identifier: wp22-fm-ts01-elfi01-tl01

Installation and setup

The procedure followed to installed Kerrighed is available at this address:
http://www.kerrighed.org/wiki/index.php/Installing_Kerrighed_2.1.0
The installation was done a Debian Sid and ended on 2007-09-14.

Running the test

Migration of cpuburn

- Install cpuburn

```
apt-get install cpuburn
```

- Start Kerrighed

```
sudo krgadm cluster start
No node specified... we're going to start all available nodes
we're going to start 0
we're going to start 1
```

- Enable Kerrighed process migration and start 4 cpuburn process:

```
krgcapset -d +CAN_MIGRATE
burnMMX & burnMMX & burnMMX & burnMMX &
```

- Running `dmesg` shows that 2 processes among 4 have migrated:

```
send_kerrighed_signal: 100667 (Migration Mgr) -> 107113 (
    (cont.)burnMMX)
send_kerrighed_signal: 100667 (Migration Mgr) -> 107114 (
    (cont.)burnMMX)
```

  Running `top` and display *last used cpu* (with f, j shortcut) confirm that processes have migrated.

Migration of mpich example (p4 version)

- Install mpich

```
sudo apt-get install mpich-bin libmpich1.0-dev
```

- Build the cpi.c example:

```
mpicc.mpich /usr/share/doc/libmpich1.0-dev/examples/pi/cpi.c
```

- Run the pi example itself without migration:

```
mpirun.mpich ./a.out -np 4 500000000
```

- The output is:

```
The computed value of the integral is 3.141592653590194
The exact value of the integral is    3.141592653589793
```

- Start Kerrighed

```
sudo krgadm cluster start
No node specified... we're going to start all available nodes
we're going to start 0
we're going to start 1
```

- Enable process migration

```
krgcapset -d +CAN_MIGRATE
```

- Run the pi example itself

```
mpirun.mpich ./a.out -np 4 500000000
```

- A process is migrated:

```
send_kerrighed_signal: 35128 (Migration Mgr) -> 35355 (a.out)
send_kerrighed_signal: 35128 (Migration Mgr) -> 35324 (a.out)
```

- Then we get a kernel error on the destination node:

```
Call Trace:
 [<c0373806>] sys_socketcall+0x326/0x3f2
 [<c0103cb0>] sysenter_past_esp+0x5d/0x81
 =======================
Code: ac 05 00 00 85 d2 0f 45 c2 8b 80 a4 00 00 00 c7 44 24
   (cont.)08 50 77 92 d0 c7 04 24 c7 f0 92 d0 89 44 24 04 e8 69
   (cont.)61 81 ef 8b 54 24 5c <8b> 02 ba d0 00 00 00 e8 38 76
   (cont.)84 ef 89 c5 b8 f4 ff ff ff 85 ed
EIP: [<d09034fe>] kcb_faf_getsockopt+0x54/0x18d [kerrighed]
   (cont.)SS:ESP 0068:ce403f1c
```

- And the process end with an error message:

```
p0_35324: (268.647121) net_send: could not write to fd=4,
   (cont.)errno = 32
```

Migration of mpich example (shmem version)

- Install mpich

```
sudo apt-get install mpich-shmem-bin libmpich-shmem1.0-dev
```

- Build the cpi.c example:

```
mpicc.mpich-shmem /usr/share/doc/libmpich1.0-dev/examples/pi/
   (cont.)cpi.c
```

- Run the pi example itself without migration:

```
mpirun.mpich-shmem ./a.out -np 4 500000000
```

- The output is:

```
The computed value of the integral is 3.141592653589730
The exact value of the integral is    3.141592653589793
```

- Start kerrighed

```
sudo krgadm cluster start
No node specified... we're going to start all available nodes
we're going to start 0
we're going to start 1
```

- Enable process migration

```
krgcapset -d +CAN_MIGRATE
```

- Run the pi example itself

```
mpirun.mpich-shmem ./a.out -np 4 500000000
```

- A process is migrated:

```
send_kerrighed_signal: 35125 (Migration Mgr) -> 35432 (a.out)
send_kerrighed_signal: 35125 (Migration Mgr) -> 35433 (a.out)
```

- We see in `dmesg` that an error occurred:

```
BUG: at /usr/src/kerrighed-2.1.1/modules/epm/g_signal.c:46
    (cont.)export_signal_struct()
 [<d08ff7d7>] export_signal_struct+0x189/0x196 [kerrighed]
 [<d0909ebb>] network_ghost_write+0x0/0xc1 [kerrighed]
 [<d08faf99>] export_task+0x5f1/0x800 [kerrighed]
 [<d08fe068>] export_kerrighed_task+0x17b/0x1b4 [kerrighed]
 [<d08fd6b6>] export_process+0xe6/0x197 [kerrighed]
 [<d0926362>] match_debug+0x17/0x76 [kerrighed]
 [<d08fb495>] send_task+0x2dd/0x4a0 [kerrighed]
 [<d08fba8e>] kcb_task_migrate+0x1bd/0x2e9 [kerrighed]
 [<c0122cbc>] dequeue_signal+0x79/0x8f
 [<c0123275>] get_signal_to_deliver+0x17d/0x4a0
 [<c0103417>] do_notify_resume+0x99/0x67e
 [<d09207e8>] tipc_handler+0x0/0x11a [kerrighed]
 [<d08f748b>] kddm_print_debug+0x27/0x159 [kerrighed]
 [<c0102f5f>] __switch_to+0x15a/0x163
 [<c03ff9df>] __sched_text_start+0x4af/0x51a
 [<c03ef315>] process_signal_queue+0x52/0x65
 [<c011d7c0>] tasklet_action+0x32/0x52
 [<c0103db2>] work_notifysig+0x13/0x19
 =======================
```

- The main process end with the following error:

```
Child process died unexpectedly from signal 7
/usr/lib/mpich-shmem/bin/mpirun.ch_shmem: line 91: 35431
    (cont.)Aborted /home/robert/./a.out "500000000"
```

Migration of Elfipole (p4 version)

- Run Elfipole without SSI

```
mpirun -np 2 ./AS_ELFIP_FMM -p01 coneSphereCoatedAbs_planX.
    (cont.)p01
```

```
...

nombre d'onde milieu 0 : (6.442805e+00, 6.442805e+00)
nombre d'onde milieu 1 : (3.143767e+00, 0.000000e+00)
nombre d'onde du vide : (3.143767e+00, 0.000000e+00)
  Size of PE grid : 2 x 1
  Size of PE block : 256
  Size of disk block : 512
 using coneSphereCoatedAbs_planX.p01

====( kpbEM: 0 in [0..0] )====

 ASSEMBLY OF THE IMPEDANCE MATRIX
build interaction Matrix : MAIN[3576] x MAIN[3576] (LOWER
    (cont.)SYMMETRIC, DOUBLE COMPLEX, MATSPIDO )
MatAssembly_SPIDO :   0% ......... 10% ......... 20%
    (cont.)......... 30% ......... 40% ......... 50% .........
    (cont.)60% ......... 70% ......... 80% ......... 90%
    (cont.)......... 100% Done.
MatPrintNorm : CC Impedance Matrix [3576 x 3576]
    (cont.)8.7734052552577545e+04
 ASSEMBLY OF THE RIGHT HAND SIDE (order 0)
build illumination Vector : MAIN[3576] x EMISSION[26] (DOUBLE
    (cont.) COMPLEX, VECSPIDO )
VecPrintNorm : RHS [3576 x 26] 1.1013923349818592e+01
 FACTORIZATION OF THE IMPEDANCE MATRIX
MatPrintNorm : Factorized Matrix [3576 x 3576]
    (cont.)8.3264484684284866e+02
 COMPUTATION OF THE SOLUTION
DIRECT METHOD
VecPrintNorm : Solution [3576 x 26] 8.5269108692446083e-02
CHAMPS LOINTAINS
MatAssembly_SPIDO :   0% ......... 10% ......... 20%
    (cont.)......... 30% ......... 40% ......... 50% .........
    (cont.)60% ......... 70% ......... 80% ......... 90%
    (cont.)......... 100% Done.
*** NO CURRENTS ***
Timer Assemblage        = 147.1
Timer Solveur           = 79.8
Timer Post-traitements = 4.2
Chrono pour Linux
```

- Run Elfipole with SSI

```
krgcapset -d +CAN_MIGRATE
mpirun -np 2 ./AS_ELFIP_FMM -p01 coneSphereCoatedAbs_planX.
    (cont.)p01
```

- An error occured at the MPI level. No kernel error log is available:

```
...

nombre d'onde milieu 0 : (6.442805e+00, 6.442805e+00)
nombre d'onde milieu 1 : (3.143767e+00, 0.000000e+00)
nombre d'onde du vide : (3.143767e+00, 0.000000e+00)
  Size of PE grid : 2 x 1
  Size of PE block : 256
  Size of disk block : 512
 using coneSphereCoatedAbs_planX.p01

====( kpbEM: 0 in [0..0] )====

 ASSEMBLY OF THE IMPEDANCE MATRIX
build interaction Matrix : MAIN[3576] x MAIN[3576] (LOWER
   (cont.)SYMMETRIC, DOUBLE COMPLEX, MATSPIDO )
MatAssembly_SPIDO :   0% ......... 10% ......... 20% ......
   (cont.)p0_35280: (15.584414) net_send: could not write to fd
   (cont.)=5, errno = 12
p0_35280:  p4_error: net_send write: -1
    p4_error: latest msg from perror: Cannot allocate memory
p0_35280: (17.809280) net_send: could not write to fd=4,
   (cont.)errno = 32
```

Migration of Elfipole (shmem version)

- Run Elfipole without SSI

```
mpirun.mpich-shmem -np 2 ./AS_ELFIP_FMM -p01
   (cont.)coneSphereCoatedAbs_planX.p01
```

- As expected the output is the same than for p4 version.

- Run Elfipole with SSI

```
krgcapset -d +CAN_MIGRATE
mpirun.mpich-shmem -np 2 ./AS_ELFIP_FMM -p01
   (cont.)coneSphereCoatedAbs_planX.p01
```

- An error occured at the MPI level. No kernel error log is available.

```
...
nombre d'onde milieu 0 : (6.442805e+00, 6.442805e+00)
nombre d'onde milieu 1 : (3.143767e+00, 0.000000e+00)
nombre d'onde du vide : (3.143767e+00, 0.000000e+00)
  Size of PE grid : 2 x 1
  Size of PE block : 256
  Size of disk block : 512
 using coneSphereCoatedAbs_planX.p01

====( kpbEM: 0 in [0..0] )====
```

```
 ASSEMBLY OF THE IMPEDANCE MATRIX
build interaction Matrix : MAIN[3576] x MAIN[3576] (LOWER
    (cont.)SYMMETRIC, DOUBLE COMPLEX, MATSPIDO )
MatAssembly_SPIDO :   0% ......... 10% ......... 20% ......
    (cont.)Child process died unexpectedly from signal 7
/usr/lib/mpich-shmem/bin/mpirun.ch_shmem: line 91: 35574 Bus
    (cont.)error                    /home/robert/./AS_ELFIP_FMM-shmem
    (cont.)"-p01" "coneSphereCoatedAbs_planX.p01"
```

Anomalous Events

- It is suggested to deploy kerrighed using nfsboot. In our installation an unidentified problem was causing rpc.statd not to start on the client nodes. So we decided to switch to systemimager to deploy kerrighed.
- The default configuration of the cluster was to netboot on eth1 (no way to select in the BIOS). It allows discovering a bug in TIPC module of kerrighed. This bug was reported to the kerrighed team:

  https://listes.irisa.fr/wws/arc/kerrighed.users/2007-08/msg00032.html
- Some limitations on -c 2 option of the `krgadm cluster start` command where identified:

  https://listes.irisa.fr/wws/arc/kerrighed.users/2007-08/msg00022.html
- The `make install` was not working in kerrighed 2.1.0. It was reported to kerrighed team and fixed in release 2.1.1.

  https://listes.irisa.fr/wws/arc/kerrighed.users/2007-08/msg00011.html
- Migration of processes using socket or shmem IPC does not work. See more in the incident report (wp22-fm-ts01-elfi01-tir01).

Incident Report Identifiers

The IPC migration incident is described in wp22-fm-ts01-elfi01-tir01.

3.4.6 Test incident report – Unexpected behavior of MPI processes

Test incident report identifier: wp22-fm-ts01-elfi01-tir01

Summary

Processes using mpich have unexpected behavior.

Incident Description

test	result
cpi.c with p4	fail with kernel error log
cpi.c with shmem	fail with kernel error log
elfipole with p4	fail
elfipole with shmem	fail

Elfipole test did not produced any kernel error but:

- Elfipole works when migration is disabled
- socket and shmem IPC are also causing problems with the cpi.c mpich example

It is deduced that even for the Elfipole the problems comes from Kerrighed.

Impact

Elfipole processes cannot be migrated by Kerrighed.

3.4.7 Test case specification – FM SMP

Test case specification identifier: wp22-fm-ts01-elfi01-tcs02

Test Items

This is the same test than wp22-fm-ts01-elfi01-tcs01, except than we use SMP nodes, so please, refer to this test case specification.

Environmental Needs

Hardware This test require a x86 compatible cluster with SMP nodes.

Software

- This test require a configured Kerrighed cluster with at least 2 nodes. Kerrighed can be found on the Kerrighed web site: http://www.kerrighed.org. As stable versions doesn't support SMP we use a development one. The subversion trunk, revision 3073 will be used.
- This test require Elfipole which is an internal EADS software not publicly available. The version of Elfipole used for this test is the CVS snapshot 20070816.
- Elfipole also require `mpich` which is available here: http://www-unix.mcs.anl.gov/mpi/mpich1/.

3.4.8 Test procedure specification

Test procedure specification identifier: wp22-fm-ts01-elfi01-tps02
Test design specification reference: wp22-fm-ts01-elfi01-tds01

Purpose

This procedure executes the test case wp22-fm-ts01-elfi01-tcs02. The procedure is almost the same than in wp22-fm-ts01-elfi01-tps01, so we will only describe differences. For installation, log and proceed, refer to wp22-fm-ts01-elfi01-tps01.

Procedure Steps

Building kerrighed development version

- Download the KERRIGHED sources:

```
svn -$\,$-username <username> co https://scm.gforge.inria.fr/
    (cont.)svn/kerrighed/trunk kerrighed
```

- Install the autotools. Automake 1.9 is required even if the documentation says that version above 1.9 works, then run autogen.sh
- Edit the kernel configuration:

```
cd kernel
make menuconfig
```

  Enable SMP support and only what you need (hard disk and network drivers).

- Build kerrighed

```
make kernel
make
```

- To install it run this commands as root:

```
make kernel-install
make install
```

3.4.9 Test incident report – Kernel panic in Kerrighed rev3073

Test incident report identifier: wp22-fm-ts01-elfi01-tir02

Summary

With revision 3073 of Kerrighed, inserting the kerrighed module into the kernel caused a kernel panic. This bug as been reported to Kerrighed project as #4200.

Incident Description

When the kerrighed module is inserted (either by /etc/init.d/kerrighed or manually) the following kernel error occurs:

```
Call Trace:
 send_msg+0x51/0x61
 tipc_disc_send_msg+0x15/0x17
 tipc_disc_create+0xb3/0xbd
 tipc_enable_bearer+0x398/0x451
...
Code: Bad EIP value.
EIP: [<00000000>] 0x0 SS:ESP 0068:ceefbcb4
```

Impact

Kerrighed revision 3073 cannot be used so we cannot test the SMP feature.

3.4.10 Test design specification

Test design specification identifier: wp22-fm-ts01-galeb01-tds01
Test plan reference: wp22-fm-ts01-tp

Features to be Tested

The following features are the subject of this test design specification:

1. migration of individual processes (test case wp22-fm-ts01-galeb01-tcs01, testing requirements R6),
2. checkpointing of applications (test case wp22-fm-ts01-galeb01-tcs01, testing requirements R47, R32, R39, R33).

Additionally, running the tests on a 64-bit architecture will test R16. The fulfillment of the following requirements from D4.2.3 will thus be evaluated:

- full test: R6,
- partial test: R47, R32, R39,

Approach Refinements

Since the tests covered by wp22-fm-ts01-elfi01-tds01 include multiple applications of various complexity, tests covered by this test design will only use the Galeb application. Galeb, as used in this test design, is a command-line application that is a text file containing the input data (a sampled function). The master process starts N slave processes, which then start the computation. Each slave process sends the results (an algebraic expression that approximates the input data, obtained by a genetic algorithm, plus a measure of error) to the master using SysV message queues. The master process simply selects the best one and outputs it.

Test Identification

Test case wp22-fm-ts01-galeb01-tcs01 (checkpointing and migration) will be performed.

Feature Pass/Fail Criteria

Feature 2 will pass if a process is successfully checkpointed (with the checkpoint stored to a file) and restarted, as well as being successfully migrated to another SSI node (without explicitly storing the checkpoint into a file). The migration must not break the on-going IPC.

3.4.11 Test case specification – FM checkpoint-migrate process

Test case specification identifier: wp22-fm-ts01-galeb01-tcs01

Test Items

This test case tests checkpointing, restart and migration of individual processes.

Input Specifications

The input of this test case consists of the specification of the way the application is started and the scenario of checkpointing, restart and migration operations that is to be executed. The actual data input to Galeb is not important because we are not interested in the results. The genetic algorithm parameters given in the Galeb configuration file should be set so that each calculation on the slave takes long enough to monitor the execution, i.e. at least one minute.

The application will be started in the following ways:

1. a single, stand-alone process,
2. a master process with a single slave,
3. a master process with one slave per SSI node,
4. a master process with 50 slaves per SSI node.

The following scenarios will be executed for each of the above configurations:

1. run the application without checkpointing or migration,
2. run the application and migrate one slave/master/multiple slaves/all processes (including master); try with 1 to 10 migrations,
3. migrate master when slaves are finished but before master reads their results from the message queue,

4. checkpoint the application while the slave processes are calculating, restart, compare results,

5. checkpoint the application once all slaves have finished,

6. combined migration/checkpointing scenario: for each migration scenario MS and each checkpointing scenario CS, do MS followed by CS as well as vice versa.

Note that the application should be modified as needed to allow the operator to trigger the migration or checkpointing at the desired stage of the execution.

Output Specifications

No errors should be reported by the application and the output file `output.txt` should appear in the current directory. For example, a successful run of the parallel calculation with two slave processes looks like this:

```
worker1:~/galeb-xos# master-smp/galeb-master-smp 2 in data/
   (cont.)xkvadratPlusEna.txt\
          popsize 500 ngen 100 time 20
MASTER: 211:started
MASTER: 118:created msg queue
MASTER: 232:starting slaves
 36288: 241:starting slave
 36289: 241:starting slave
MASTER: 256:all slaves started, waiting for them to finish
 36289: 243:finished, result is 200.000000
 36289: 133:sending result
 36289: 245:results sent to master, slave exiting
 36288: 243:finished, result is 6.840950
 36288: 133:sending result
 36288: 245:results sent to master, slave exiting
MASTER: 271:all slaves finished
Done, 2 of 2 slaves succeeded. Parsing results...
MASTER: 158:received 200.000000
MASTER: 168:last received is best so far
MASTER: 158:received 6.840950
MASTER: 168:last received is best so far
MASTER: 171:receiving done
MASTER: 280:all done, cleaning up
Parsed results of the 2 successful slaves.
Program executed successfully!
MASTER: 287:exiting
MASTER: 128:removed msg queue
worker1:~/galeb-xos# cat output.txt
6.840950
(exp(-0.2045))+((x)*(x))
```

Note that each log line contains the process identifier and the line number.

Environmental Needs

All environmental needs stated in wp22-fm-ts01-tp plus the following.

Software Galeb application (master executable, slave executable, input data, configuration file) in the current directory.

3.4.12 Test procedure specification

Test procedure specification identifier: wp22-fm-ts01-galeb01-tps01
Test design specification reference: wp22-fm-ts01-galeb01-tds01
Test case specification reference: wp22-fm-ts01-galeb01-tcs01
Test log identifier: wp22-fm-ts01-galeb01-tl01

Purpose

This procedure executes the test case wp22-fm-ts01-galeb01-tcs01.

Procedure Steps

Log The Kerrighed signals are logged automatically in the system log, which can be seen using the `dmesg` command.

Compiling Galeb master with the DEBUG symbol defined causes it to print the status information to the console.

Set Up The Galeb application must be installed. The input files must reside on a shared filesystem in order to be read by all nodes. If the scenario requires an action exactly at a certain stage (e.g., during IPC), modify the application to wait for user input somewhere during that stage.

Before starting, run the following commands:

```
krgcapset -k $$ -d +CAN_MIGRATE
krgcapset -k $$ -d +DISTANT_FORK
krgcapset -k $$ -d +CHECKPOINTABLE
krgcapset -k $$ -s
```

The output of the last command shows the inheritable effective capabilities of the current process (i.e. the shell):

```
Inheritable Effective Capabilities: 027
        CHANGE_KERRIGHED_CAP, CAN_MIGRATE, DISTANT_FORK,
            (cont.)CHECKPOINTABLE
```

The processes started from this shell will thus be migratable and checkpointable.

Start The stand-alone version is started using the following command:

```
./galeb-slave popsize <pop size> \
   ngen <num of generations> in <input file>
```

The popsize and ngen parameters should be set appropriately so that the calculation takes at least 1 minute.

The cluster flavor of the parallel Galeb is started by starting the master process:

```
./galeb-master-smp <number of slaves> \
   <parameters for slave process (see above)>
```

Proceed According to the selected scenario (see Section wp22-fm-ts01-galeb01-tcs01 for the list), trigger migration and/or checkpointing at an appropriate moment.

Contingencies The anomalies should be dealt with by killing all remaining Galeb processes in order not to interfere with further testing. This will leave all open message queues in the system, which should be deleted using the `ipcrm` command. If any kernel errors are reported, the cluster should be rebooted to make sure that the error will not affect further testing.

3.4.13 Test log

Test log identifier: wp22-fm-ts01-galeb01-tl01

Description

The test procedure wp22-fm-ts01-galeb01-tps01 was executed by Marjan Šterk, XLAB. The tests were done on a three-node cluster connected through a dedicated Ethernet switch. The head node ran on Debian GNU/Linux with kernel 2.6.18 and was only used as a DHCP/IPX/NFS server to boot the other two, diskless nodes. LinuxSSI revision 3075 was checked out from the Kerrighed SVN and installed by Marko Novak, XLAB.

Installation and Setup

Execution Description The LinuxSSI branch of Kerrighed, revision 3075, was checked out from the Kerrighed SVN server on 2007-10-19. The installation for the two diskless nodes was then performed following the instructions on http://www.kerrighed.org/wiki/index.php/Installing_Kerrighed_2.1.0#Installing_Kerrighed_from_the_tarball.

Procedure Results The installation of LinuxSSI was successful. After issuing the command `krg-adm cluster start` the two nodes are joined to logically form a single system.

Running Galeb without migration or checkpointing

Execution Description The tests were done on 2007-10-29. The application was started in the four ways enumerated in wp22-fm-ts01-galeb01-tcs01 following the procedure described in wp22-fm-ts01-galeb01-tps01.

Procedure Results Successful in all cases.

Migration scenarios

Execution Description The tests were done on 2007-10-29. The application was started in the four ways enumerated in wp22-fm-ts01-galeb01-tcs01 following the procedure described in wp22-fm-ts01-galeb01-tps01. The number of migrations was gradually increased to 10, unless the application failed consistently with a lower number of migrations.

Procedure Results If no process was migrated more than once, no problems were encountered. Furthermore, the master process could be migrated any number of times without problems, regardless of whether the migration was triggered while the slave processes were calculating or after they had finished and the master only had to collect their results.

However, migrating the calculating (slave) process itself turned out to be very problematic. If the calculation was run as a stand-alone application, it consistently crashed during the second migration. If the calculation was run in parallel, migrating slaves twice always caused at least one of the migrated slaves to fail. Some of the migrated slaves and all others finished successfully.

Anomalous events As described above, migrating a calculating process twice consistently crashed the process. Because the error occurs even when the process is run stand-alone, we suppose that it has nothing to do with parallel execution or IPC. Detailed analysis is given in the incident report wp22-fm-ts01-galeb01-tir01.

Checkpointing scenarios

Execution Description The tests were done on 2007-10-30. The application was started in the four ways enumerated in wp22-fm-ts01-galeb01-tcs01 following the procedure described in wp22-fm-ts01-galeb01-tps01.

Procedure Results Checkpointing and restarting a stand-alone slave process was always successful. The results of the restarted process were the same as of the original one.

Restarting a checkpointed parallel application never worked correctly. If the checkpoint was made while the slaves were still calculating, the master process assumed on restart that all slave processes have failed. If the checkpoint was made

after the slaves finished the calculation and before the master started reading their results from the message queue, the reading failed.

Anomalous events As described above, a parallel application was never restarted successfully. For detailed analysis please see the incident report wp22-fm-ts01-galeb01-tir02.

We also noted that the restarted process always carries the same PID as the original one, so restart is not possible if there is another process with the same PID. This is unlikely to happen during a single session because LinuxSSI does not reuse PIDs of the terminated processes (we did not test what happens when the system runs ouf of PID-space). However, if the cluster is rebooted, another process can take the PID of a checkpoint, which makes restart impossible until the next reboot (we were able to cause such behaviour). The reuse of PIDs on reboot is a minor problem that should be solved in later versions.

The combined migration/checkpointing/migration scenario

Execution Description The tests were done on 2007-10-30. A stand-alone application was started, migrated, and checkpointed, following the procedure described in wp22-fm-ts01-galeb01-tps01. A restart from the checkpoint was then triggered.

Procedure Results The restart of the stand-alone migrated process always failed. The more complicated configurations and scenarios were thus not tested.

Anomalous events As described above, restarting a stand-alone process that had been migrated before checkpointing always failed. Detailed analysis is given in the incident report wp22-fm-ts01-galeb01-tir03.

3.4.14 Test incident report – FM Galeb second migration

Test incident report identifier: wp22-fm-ts01-galeb01-tir01

Summary

As notified in wp22-fm-ts01-galeb01-tl01, the calculating process of the Galeb application fails on second migration, regardless of whether it is run stand-alone or as a slave process.

Incident Description

If the process is run stand-alone in the usual way, a segmentation fault happens on the second migration. Note that in our two-node test cluster the second migration always meant migrating the process back onto the node where it was originally started.

Because we suspected that the error may be correlated to memory allocation, the application was modified to `sleep()` every now and then during calculation. The migrations were then made only during sleeping, when no memory allocation could possibly be going on. With such modified approach up to three migrations were always successful, however, the failure occured anytime from 4th to 13th migration. The error with this approach was either "Segmentation fault" or an error log like the following:

```
*** glibc detected *** slave/galeb-slave: malloc(): memory
    (cont.)corruption (fast): 0x080aa0c8 ***
======= Backtrace: =========
/lib/libc.so.6[0xb7d7607b]
/lib/libc.so.6(__libc_malloc+0x90)[0xb7d770a0]
/usr/lib/libstdc++.so.6(_Znwj+0x27)[0xb7f3d777]
slave/galeb-slave[0x805a6f0]
slave/galeb-slave[0x805a68b]
slave/galeb-slave[0x805a68b]
slave/galeb-slave[0x805a68b]
slave/galeb-slave[0x805a68b]
slave/galeb-slave[0x805a788]
slave/galeb-slave[0x805a68b]
slave/galeb-slave[0x805a68b]
slave/galeb-slave[0x805a936]
slave/galeb-slave[0x805b0c4]
slave/galeb-slave[0x8072dd0]
slave/galeb-slave[0x80502a9]
slave/galeb-slave[0x804bc04]
/lib/libc.so.6(__libc_start_main+0xe0)[0xb7d21050]
slave/galeb-slave(__gxx_personality_v0+0x9d)[0x804ab01]
======= Memory map: ========
08048000-08088000 r-xp 00000000 00:11 1952108     /root/galeb-
    (cont.)xos/slave/galeb-slave
08088000-08089000 rwxp 00040000 00:11 1952108     /root/galeb-
    (cont.)xos/slave/galeb-slave
08089000-080aa000 rwxp 08089000 00:00 0
080aa000-080cb000 rwxp 080aa000 00:00 0
b7c00000-b7c21000 rwxp b7c00000 00:00 0
b7c21000-b7d00000 ---p b7c21000 00:00 0
b7d0a000-b7d0b000 rwxp b7d0a000 00:00 0
b7d0b000-b7e4d000 r-xp 00000000 00:11 1751066     /lib/libc
    (cont.)-2.6.1.so
b7e4d000-b7e4e000 r-xp 00142000 00:11 1751066     /lib/libc
    (cont.)-2.6.1.so
b7e4e000-b7e50000 rwxp 00143000 00:11 1751066     /lib/libc
    (cont.)-2.6.1.so
b7e50000-b7e54000 rwxp b7e50000 00:00 0
b7e54000-b7e5e000 r-xp 00000000 00:11 1750637     /lib/libgcc_s
    (cont.).so.1
b7e5e000-b7e5f000 rwxp 00009000 00:11 1750637     /lib/libgcc_s
    (cont.).so.1
b7e5f000-b7e83000 r-xp 00000000 00:11 1751072     /lib/libm
    (cont.)-2.6.1.so
b7e83000-b7e85000 rwxp 00023000 00:11 1751072     /lib/libm
```

```
    (cont.)-2.6.1.so
b7e85000-b7f65000 r-xp 00000000 00:11 1751499     /usr/lib/
    (cont.)libstdc++.so.6.0.9
b7f65000-b7f68000 r-xp 000e0000 00:11 1751499     /usr/lib/
    (cont.)libstdc++.so.6.0.9
b7f68000-b7f6a000 rwxp 000e3000 00:11 1751499     /usr/lib/
    (cont.)libstdc++.so.6.0.9
b7f6a000-b7f70000 rwxp b7f6a000 00:00 0
b7f72000-b7f76000 rwxp b7f72000 00:00 0
b7f76000-b7f92000 r-xp 00000000 00:11 1751058     /lib/ld
    (cont.)-2.6.1.so
b7f92000-b7f94000 rwxp 0001b000 00:11 1751058     /lib/ld
    (cont.)-2.6.1.so
bfb33000-bfb48000 rw-p bfb33000 00:00 0           [stack]
ffffe000-fffff000 r-xp 00000000 00:00 0           [vdso]
Aborted
```

As expected, the error occurs also if the calculating process that is being migrated is a part of a parallel computation. For example, starting with two slaves and migrating the first one twice resulted in the following log:

```
worker1:~/galeb-xos# master-smp/galeb-master-smp 2 in data/
   (cont.)xkvadratPlusEna.txt\
          popsize 500 ngen 100 time 20
MASTER: 211:started
MASTER: 118:created msg queue
MASTER: 232:starting slaves
 36292: 241:starting slave
 36293: 241:starting slave
MASTER: 256:all slaves started, waiting for them to finish

(slaves calculate)
(we migrate 36292 from worker1 to worker2)
(both continue to calculate)
(we migrate 36292 back)

*** glibc detected *** master-smp/galeb-master-smp: malloc():\
memory corruption (fast): 0x0826f8f0 ***
======= Backtrace: =========
/lib/libc.so.6[0xb7ce607b]
/lib/libc.so.6(__libc_malloc+0x90)[0xb7ce70a0]
/usr/lib/libstdc++.so.6(_Znwj+0x27)[0xb7ead777]
master-smp/galeb-master-smp[0x80625d5]
master-smp/galeb-master-smp[0x8050960]
master-smp/galeb-master-smp[0x807cbcd]
master-smp/galeb-master-smp[0x807a039]
master-smp/galeb-master-smp[0x8073e49]
master-smp/galeb-master-smp[0x8074593]
master-smp/galeb-master-smp[0x80516f9]
master-smp/galeb-master-smp[0x804d4ee]
master-smp/galeb-master-smp[0x804bb29]
/lib/libc.so.6(__libc_start_main+0xe0)[0xb7c91050]
master-smp/galeb-master-smp(__gxx_personality_v0+0x91)[0x804aeb1]
======= Memory map: ========
```

```
08048000-0808a000 r-xp 00000000 00:11 1952099    /root/galeb-xos/
    (cont.)master-smp/galeb-master-smp
0808a000-0808b000 rwxp 00042000 00:11 1952099    /root/galeb-xos/
    (cont.)master-smp/galeb-master-smp
0808b000-081b3000 rwxp 0808b000 00:00 0
081b3000-08277000 rwxp 081b3000 00:00 0
b7b00000-b7b21000 rwxp b7b00000 00:00 0
b7b21000-b7c00000 ---p b7b21000 00:00 0
b7c7a000-b7c7b000 rwxp b7c7a000 00:00 0
b7c7b000-b7dbd000 r-xp 00000000 00:11 1751066    /lib/libc-2.6.1.
    (cont.)so
b7dbd000-b7dbe000 r-xp 00142000 00:11 1751066    /lib/libc-2.6.1.
    (cont.)so
b7dbe000-b7dc0000 rwxp 00143000 00:11 1751066    /lib/libc-2.6.1.
    (cont.)so
b7dc0000-b7dc4000 rwxp b7dc0000 00:00 0
b7dc4000-b7dce000 r-xp 00000000 00:11 1750637    /lib/libgcc_s.so
    (cont.).1
b7dce000-b7dcf000 rwxp 00009000 00:11 1750637    /lib/libgcc_s.so
    (cont.).1
b7dcf000-b7df3000 r-xp 00000000 00:11 1751072    /lib/libm-2.6.1.
    (cont.)so
b7df3000-b7df5000 rwxp 00023000 00:11 1751072    /lib/libm-2.6.1.
    (cont.)so
b7df5000-b7ed5000 r-xp 00000000 00:11 1751499    /usr/lib/libstdc
    (cont.)++.so.6.0.9
b7ed5000-b7ed8000 r-xp 000e0000 00:11 1751499    /usr/lib/libstdc
    (cont.)++.so.6.0.9
b7ed8000-b7eda000 rwxp 000e3000 00:11 1751499    /usr/lib/libstdc
    (cont.)++.so.6.0.9
b7eda000-b7ee0000 rwxp b7eda000 00:00 0
b7ee3000-b7ee6000 rwxp b7ee3000 00:00 0
b7ee6000-b7f02000 r-xp 00000000 00:11 1751058    /lib/ld-2.6.1.so
b7f02000-b7f04000 rwxp 0001b000 00:11 1751058    /lib/ld-2.6.1.so
bfab8000-bfacd000 rw-p bfab8000 00:00 0          [stack]
ffffe000-fffff000 r-xp 00000000 00:00 0          [vdso]
Warning: slave 36292 did not finish successfully.
 36293: 243:finished, result is 7.227602
 36293: 133:sending result
 36293: 245:results sent to master, slave exiting
MASTER: 271:all slaves finished

Done, 1 of 2 slaves succeeded. Parsing results...
MASTER: 158:received 7.227602
MASTER: 168:last received is best so far
MASTER: 171:receiving done
MASTER: 280:all done, cleaning up
Parsed results of the 1 successful slaves.
Program executed successfully!
MASTER: 287:exiting
MASTER: 128:removed msg queue
```

The final result of the application is thus simply the result returned by the slave that was not migrated:

```
worker1:~/galeb-xos# cat output.txt
7.227602
(0.3967)+((0.3967)+((0.3967)+((x)*(x))))
```

Impact

The described error is a critical bug in LinuxSSI and as such should be fixed quickly. It prevents testing more complicated migration and combined migration/checkpointing scenarios. It also prevents testing the scheduler because migration is a basic functionality required by schedulers.

3.4.15 Test incident report – FM Galeb parallel application restart

Test incident report identifier: wp22-fm-ts01-galeb01-tir02

Summary

As notified in wp22-fm-ts01-galeb01-tl01, restarting a checkpointed parallel application was never successful.

Incident Description

The behaviour of the restarted application depended on the stage at which the checkpoint was done. If the Galeb application is checkpointed while the slave processes are calculating and the master is waiting for them in a `waitpid()` system call, the output is as follows:

```
worker1:~/galeb-xos# master-smp/galeb-master-smp 2 in data/
   (cont.)xkvadratPlusEna.txt\
            popsize 500 ngen 100 time 20
MASTER: 211:started
MASTER: 118:created msg queue
MASTER: 232:starting slaves
 36273: 241:starting slave
 36274: 241:starting slave
MASTER: 256:all slaves started, waiting for them to finish

(checkpoint)
(terminate the original application with Ctrl-C)
(see if any IPC was left)

worker1:/var/chkpt# ipcs
...
------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x00000000 32768      root       600        0            0

(restart)
worker1:~/galeb-xos# restart 36272 1
```

```
Restart application 36272 (v1) ...
Warning: slave 36273 did not finish successfully.
Warning: slave 36274 did not finish successfully.
MASTER: 271:all slaves finished
Done, 0 of 2 slaves succeeded. Parsing results...
worker1:~/galeb-xos# MASTER: 171:receiving done
MASTER: 280:all done, cleaning up
Parsed results of the 0 successful slaves.
Program executed successfully!
MASTER: 287:exiting
MASTER: 128:removed msg queue

(both slaves still calculate)
 36274: 243:finished, result is 200.000000
 36274: 133:sending result
 36274: 245:results sent to master, slave exiting
 36273: 243:finished, result is 200.000000
 36273: 133:sending result
 36273: 245:results sent to master, slave exiting
(the slaves don't check whether sending was successful)

worker1:~/galeb-xos# ipcs
...
------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
```

The problem is that on restart of the master, the `waitpid()` system call returns immediately, before the slave processes are even created. The master thus exits, thinking that all slaves have failed. The slaves then finish the calculation but the results are lost because there is no master to collect them.

Another problem is that if we wait for the application to finish after the checkpoint is done instead of terminating it, the message queue is removed by the application and should be restored from the checkpoint, which does not happen. This does not affect the outcome because the master thinks all slaves failed anyway. The only difference is that when the master exits, it cannot delete the message queue:

```
...
MASTER: 287:exiting
Error 22 removing message queue
```

If the checkpoint is done during an artificially inserted pause after the slaves have finished but before the collection of the results and the original application is terminated after checkpointing, the messages from slaves remain in the queue. The application (i.e. the master process, as the slaves are already done) can be restarted and successfully reads results and finishes:

```
worker1:~/galeb-xos# master-smp/galeb-master-smp 2 in data/
   (cont.)xkvadratPlusEna.txt\
            popsize 500 ngen 100 time 20
MASTER: 211:started
MASTER: 118:created msg queue
MASTER: 232:starting slaves
```

```
 36295: 241:starting slave
 36296: 241:starting slave
MASTER: 256:all slaves started, waiting for them to finish
 36296: 243:finished, result is 200.000000
 36296: 133:sending result
 36296: 245:results sent to master, slave exiting
 36295: 243:finished, result is 0.165203
 36295: 133:sending result
 36295: 245:results sent to master, slave exiting
MASTER: 271:all slaves finished

(checkpoint)
(terminate with Ctrl-C)

worker1:~/galeb-xos# ipcs
...
------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x00000000 98304      root       600        72           2

worker1:~/galeb-xos# restart 36294 1
Restart application 36294 (v1) ...
worker1:~/galeb-xos#
worker1:~/galeb-xos# Done, 2 of 2 slaves succeeded. Parsing
   (cont.)results...
MASTER: 158:received 200.000000
MASTER: 168:last received is best so far
MASTER: 158:received 0.165203
MASTER: 168:last received is best so far
MASTER: 171:receiving done
MASTER: 280:all done, cleaning up
Parsed results of the 2 successful slaves.
Program executed successfully!
MASTER: 287:exiting
MASTER: 128:removed msg queue
```

However, if the message queue does not exist anymore (because the original application closed it or because the system was rebooted), the results of the slaves are lost again.

Impact

The described error represents an important shortcoming of the implemented checkpointer. The checkpointing and restarting mechanisms should be improved to successfully cope with parallel applications that use System V IPC. The error also prevents performing more complicated combined checkpointing/migration tests.

3.4.16 Test incident report – FM Galeb checkpoint after migration

Test incident report identifier: wp22-fm-ts01-galeb01-tir03

Summary

As notified in wp22-fm-ts01-galeb01-tl01, a stand-alone calculating process failed to restart from a checkpoint that was made after migration.

Incident Description

The calculating process was started, migrated and checkpointed:

```
worker1:~/galeb-xos# slave/galeb-slave in data/xkvadratPlusEna.txt
   (cont.)\
            popsize 500 ngen 1000 time 30

(worker2:/var/chkpt# migrate 36261 1)
(worker2:/var/chkpt# checkpoint 36261)
(Checkpointing application in which process 36261 is involved)
(Identifier: 36261)
(Version: 1)
(Description: No description)
(Date: Tue Oct 30 17:22:02 2007)

(the original program ends, outputting the results:)
200
(x)*(x)
0;*;[1,2];
1;x;int;
2;x;int;
```

The restart was then triggered, which produced the following errors:

```
worker2:/var/chkpt# restart 36261 1
Restart application 36261 (v1) ...

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: Oops: 0000 [#1]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: CPU:    0

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: EIP:    0060:[<f89253c0>]    Not tainted VLI

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: EIP is at faf_file_import+0xcf/0x167 [kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: ds: 007b   es: 007b   ss: 0068

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: eax: dfe42e30   ebx: fffffffe   ecx: dffff140
   (cont.)edx: c13fbd80

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: EFLAGS: 00010286   (2.6.20-krgLinuxSSI #1)
```

```
Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: Process krgrpc(93:2867 (pid: 101764, ti=dfd2e000
   (cont.)task=dfc8c740 task.ti=dfd2e000)

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: esi: dfdec1c0   edi: f7d383bc   ebp: c192e1b0
   (cont.)esp: dfd2fcd4

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: Call Trace:

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: Stack: f89500bb f89581da 00000003 ffffffff
   (cont.)ffffffff ffffffff 00000000 f8958301

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel:        c1ba5d08 dfd2ff00 0000002e f897e5ac
   (cont.)f7d383bc 00000000 c192e1b0 f892836a

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel:  [<f892836a>] import_one_open_file+0xd7/0x37c [
   (cont.)kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel:        dfe42e30 f8958aa7 00000002 00000000
   (cont.)ffffffff ffffffff 00000009 f8958d15

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel:  [<f89286af>] import_open_files+0xa0/0x15a [
   (cont.)kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel:  [<f8928a18>] import_files_struct+0x2af/0x370 [
   (cont.)kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel:  [<f891a812>] import_task+0x51b/0x889 [kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel:  [<f894eb26>] match_debug+0x18/0x78 [kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel:  [<f891d1e0>] import_process+0xdf/0x1f7 [kerrighed
   (cont.)]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel:  [<c0161845>] mntput_no_expire+0x11/0x47

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel:  [<f891dd7b>] restart_process+0x2b7/0x814 [
   (cont.)kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
```

```
worker2 kernel: [<f89081cf>] handle_init_restart+0x4db/0x52a [
  (cont.)kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: [<f8943edc>] rpc_desc_send_alloc+0x12/0x2f [
  (cont.)kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: [<f89455d6>] rpc_handler_kthread_int+0x50/0x7c [
  (cont.)kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: [<f894eb26>] match_debug+0x18/0x78 [kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: [<f8908aeb>] handle_do_restart+0x101/0x1d9 [
  (cont.)kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: [<f894582d>] thread_pool_run+0x0/0x418 [kerrighed
  (cont.)]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: [<f89459f6>] thread_pool_run+0x1c9/0x418 [
  (cont.)kerrighed]

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: [<c012ba26>] kthread+0x8a/0xaf

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: [<c012b99c>] kthread+0x0/0xaf

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: [<c0104847>] kernel_thread_helper+0x7/0x10

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: Code: 44 24 24 83 38 03 75 0b 89 f2 89 e8 e8 4f 05
  (cont.) 00 00 eb 09 89 f2 89 e8 e8 a3 f8 ff ff 89 c3 89 f0 e8 18
  (cont.)92 82 c7 8b 44 24 40 89 18 <8b> 4b 18 8b 73 78 f7 c1 00 00
  (cont.)10 00 74 05 8b 7b 74 eb 16 89 c8

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: ========================

Message from syslogd@worker2 at Tue Oct 30 17:22:28 2007 ...
worker2 kernel: EIP: [<f89253c0>] faf_file_import+0xcf/0x167 [
  (cont.)kerrighed] SS:ESP 0068:dfd2fcd4
```

The `restart` process stayed in the system and could not be killed:

```
worker1:~/galeb-xos# pkill restart
worker1:~/galeb-xos# ps -F -u root|grep restart
root     36268 36245  0   878   712   0 17:27 pts/1    00:00:00
  (cont.)grep restart
```

```
root      101795  101780  0 389    392    1 17:22 pts/0    00:00:01
   (cont.)restart 36261 1
```

Restarting any process was impossible until rebooting the cluster. The test was repeated with a simple `cpu_eater` process and the error was the same.

Impact

The described error is a critical bug in LinuxSSI and as such should be fixed quickly. It prevents testing more complicated combined migration/checkpointing scenarios. The error may be correlated to the one described in wp22-fm-ts01-galeb01-tir01.

3.4.17 Test summary report – FM

Test summary report identifier: wp22-fm-ts01-tsr

Summary

Kerrighed was evaluated using:

- simples programs (cpuburn and mpich program sample) in wp22-fm-ts01-elfi01-tcs01 and Kerrighed 2.1.1.
- elfipole in wp22-fm-ts01-elfi01-tcs01 and Kerrighed 2.1.1.
- elfipole in wp22-fm-ts01-elfi01-tcs02 and Kerrighed trunk revision
- galeb in wp22-fm-ts01-elfi01-tcs02 and Kerrighed LinuxSSI branch revision

Variances

Two features where not tested:

- SMP support (R17)
- customizable scheduler

Both are not available in the last stable version of Kerrighed (2.1.1), so development version where tried. Those versions were not stable enough to do usable tests. Test on 64bit version of Kerrighed were also looked after but Kerrighed developers says it was not yet ready.

Summary of Results

Tests shows that processes migration is not yet robust enough to do more complex tests. Anyway some SSI features are working:

- Unified /proc (It is possible to monitor and kill processes from any nodes of the cluster).
- Migration of simple processes

Evaluation

Tested requirements are partly fulfilled. We present the list of XtreemOS-related requirements and their fulfillment status:

R6: partly fulfilled (will be fulfilled once the bugs described in the incident reports are fixed)

R32, R47: partly fulfilled. Automatic failure detection is not yet implemented, the error described in wp22-fm-ts01-galeb01-tir01 shows there is a bug in migration.

R39: partly fulfilled. wp22-fm-ts01-galeb01-tcs01 and wp22-fm-ts01-elfi01-tcs01 show that IPC is not restored properly, also, test applications uses a single thread per process and no network communication, while checkpointing while a file is open was not tested.

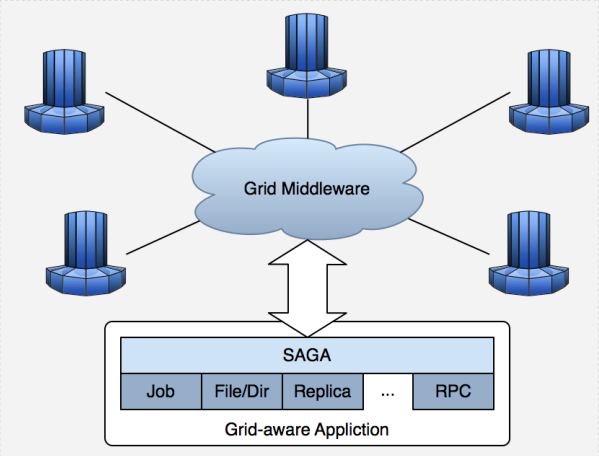
Summary of Activities

Setting a Kerrighed test bed up could be a long task. Setting an environment for easy deployment (i.e. nfsboot) or configure a working linux kernel could be time consuming. Even if Kerrighed is not ready for large test case we will probably use Grid5000 for next tests to save time on test bed maintenance. A testbed installation of Kerrighed on Grid5000 needs to fulfill the following requirements:

- KVM like access (using Kaconsole)
- easy deployment (using Kadeploy)
- SMP and 64bit CPUs

3.5 Evaluation of XtreemOS API : SAGA

SAGA stands for “a Simple API for Grid Application”. It is an application level programming abstraction that builds on the top of XtreemOS or of grid middleware such as Globus or OMII-UK.



Two SAGA implementations – in C++ and Java – have been planned as well as wrappings in C, fortran, Python and Perl. The only technical requirement is the availability of the library boost in a version higher than 1.33.

The 0.6 and 0.7 versions of the C++ implementation ("SAGA-A") have been respectively released in June and November 2007. SAGA is the latest Open Source adoption of the GWD-R.90 (SAGA Core API) standard. It provides a Grid programming API for the six functional packages defined by GWD-R90 1.0:

- Replica Management
- Job Management
- Namespaces
- File Management
- Remote Procedure Call
- Streams

3.5.1 XtreemOS SAGA Implementation

In XtreemOS project, workpackage **WP 3.1** is responsible for the porting of SAGA. The first release of an XtreemOS component is planned for February 2008.

Though the XtreemOS component is not yet available, one can already have a fair idea of the completeness of the API and the fulfillment of the requirements by reading the published specification of SAGA and by experimenting with it running only on a standard Linux Box.

3.5.2 Test plan – Evaluation of the SAGA API

Test plan identifier: wp31-api-ts01-tp

Introduction

This test plan evaluates the completeness of the C++ implementation of SAGA API developed by WP3.1. The SAGA functionalities include:

- Submission, monitoring, deletion of jobs.
- Creation, deletion, copying and renaming of files.
- Remote procedure Calls.
- Stream management.
- Replica management

Test Items

We use with the 0.6 version of the SAGA-A C++ implementation built on the 1.34 boost library. This version is installed on a linux x86_64 machine running debian etch distribution.

- Boost 1.34 C++ library was downloaded from the web site:
  http://www.boost.org/.
- SAGA-A 0.6 C++ implementation library was directly checked out from the SVN repository (revision 768):
  https://svn.cct.lsu.edu/repos/saga/branches/sage-0.6/trunk
  The 10-day old SAGA-A 0.7 version released on November 12^{th} was also downloaded but abandoned because the examples did not compile correctly and revealed a mismatch of header files. The same problem occurred for our test application. The SAGA development team recommended us to take the previous release instead.

Features to be Tested

In this document, the following features will be tested:

- Submission, monitoring, deletion of jobs.
- Creation, deletion, copying and renaming of files.

The SAGA filesystem API entirely covers the POSIX API and makes it usable to several grid platforms. A user may then prefer to use the SAGA API instead of regular POSIX file API, in order to obtain a portable application over these environment

Note that the C++ implementation of SAGA specification is still ongoing and the tested release (0.6) partially supports it.

Features not to be Tested

The following features are not tested as they are not supported by the 0.6 release:

- Remote procedure Calls.
- Stream management.
- Replica management

As mentioned in the SAGA specification, these features are part of the complete implementation of SAGA which should be available on XtreemOS.

Approach

The purpose of the early test is to provide feedback to developers about the implemented features and about the fulfillment of requirements.

Tests are performed by installing a version of SAGA that runs locally on one testbed machine. That way, we could test calls to the SAGA C++ API but all the spawned job were executing locally.

Item Pass Criteria

SAGA will pass a test if:

- running application under the condition defined by the test design specification does not crash the application.
- an application calling SAGA, under the condition of the test, gives the same results as if jobs or files were managed directly from a shell prompt.

Testing Tasks

The testing tasks include:

- building and installing SAGA
- running simple programs that submit, monitor, and delete jobs running the ZEPHYR application.
- running simple programs managing files (initialization of a dir tree, creation, writing, reading, deletion and renaming of files).

The testing tasks does not yet include:

- running and benchmarking SAGA asynchronous tasks support.
- performing remote procedure calls.
- sharing streams among SAGA processes

- evaluating the Advert Service

As mentioned in the SAGA specification, these features are part of the complete implementation of SAGA which should be available on XtreemOS. These latter features are planned to be tested in succeeding WP4.2 deliverables.

Environmental Needs

SAGA and Boost have no special hardware requirements. They could be executed on a regular desktop two-processor PC with 2 GB RAM, and 2 GHz CPU clock speed. They are designed to be independent of any specific Linux distribution. On the testbed, we run Linux Kernel 2.6.18-4-amd64 with SMP and 64 bit support. SAGA and Boost have been compiled and installed with the pre-installed third-party software and libraries :

- gmake 3.81
- gcc 4.1.2
- autoconf 2.61
- automake 1.9.6

Responsibilities

This test plan and the included tests are the responsibility of EDF.

Schedule

Tests have been performed from November 15^{th} to December 14^{th} 2007.

Risks and Contingencies

Delays in testing caused by unexpected installation and execution problems are envisaged.

3.5.3 Test design specification

Test design specification identifier: wp31-api-ts01-zephyr01-tds01

Features to be Tested

In this release, we test some of the available features of the SAGA C++ API running on a regular Linux system. In particular, the support of

- Job Management : spawn, monitor, cancel or terminate a job
- File management : create, copy, rename, delete files in different directories.

Approach Refinements

After compiling and installing SAGA (first test case), we test the SAGA implementation and API in two ways:

- a test application that runs, monitors, and cancels a ZEPHYR application.
- a test program that performs file operations (creation of a directory tree, creation, copy, renaming or deletion of files)

Test Identification

Two test cases relate to this design specification :

- wp31-api-ts01-zephyr01-tcs01 dealing with the installation of SAGA and focusing on job management issues.
- wp31-api-ts01-zephyr01-tcs01 dealing with files handling operations.

Feature Pass/Fail Criteria

Tests are considered as passed when the test program behavior and output correspond to the SAGA specification.

3.5.4 Test case specification – Job Management

Test case specification identifier: wp31-api-ts01-zephyr01-tcs01

Test Items

Using the SAGA API, this test will run the EDF Application Zephyr on the target machine, regularly monitoring its state, checking whether it is still running or not till its nominal or in error termination. It will also test the cancellation of an ongoing application.

Input Specifications

Input is generated by the test program itself. No additional input is required.

Input needed by the application Zephyr are read from a “fort.55” file. This file must be present in the same directory from where the test program is run.

Output Specifications

The test program prints its output on the standard output stream as reproduced further in this document.

The application Zephyr dumps its text output into a “fort.66” file.

Environmental Needs

Software The test program spawns, monitors and cancels an actual EDF application : Zephyr.

The hardware and software testing environment is the one described at section wp31-api-ts01-tp

3.5.5 Test procedure specification

Test procedure specification identifier: wp31-api-ts01-zephyr01-tps01

Purpose

This procedure executes the test case wp31-api-ts01-zephyr01-tcs01.

Procedure Steps

Set Up The generic test procedure setup includes:

- Installation and compilation of SAGA.
- Compiling and running the test program

Installation and compilation of SAGA on Debian Linux etch distribution including the following steps:

1. Download of Boost 1.34.1
2. Compilation of Boost and manual linking of a shared library.
3. Checking out the 0.6 release of SAGA.
4. Compilation of SAGA.

Start In order to start the test one needs to perform the following steps:

1. Download, compile and install the 1.34.1 boost C++ library :

```
Download boost_1_34_1.tar.gz from website http://www.boost.
    (cont.)org
cd /home/kortas/BUILD
tar xvfz /home/kortas/XTREEMOS/WP4.2/SAGA/boost_1_34_1.tar.
    (cont.)gz
cd boost_1_34_1/
./configure --prefix=/home/kortas/SOFTS/boost_34_1
make
make install
```

2. Download, compile and install the 0.6 SAGA-A C++ implementation

```
cd /home/kortas/BUILD
svn checkout svn co https://svn.cct.lsu.edu/repos/saga/
   (cont.)branches/saga-0.6/trunk
mv trunk saga-c++-0.6-src
export LD_LIBRARY_PATH=/home/kortas/SOFTS/boost_34_1/lib:
   (cont.)$LD_LIBRARY_PATH
./configure --with-boost=/home/kortas/SOFTS/boost_34_1 --
   (cont.)prefix=/home/kortas/SOFTS/saga-06
make
```

As stated in the release notes, but not on the website, the gcc version used must be 3.4.8 or higher otherwise the code will not compile because of inconsistent C++ headers.

3. Compile the following test program inspired by the one in /home/kortas/BUILD/saga-c++-0.6/examples/packages/job/job_run.cpp and running a Zephyr application.

```
//  Copyright (c) 2005-2007 Andre Merzky (andre@merzky.net)
//
//  Distributed under the Boost Software License, Version
   (cont.)1.0. (See accompanying
//  file LICENSE_1_0.txt or copy at http://www.boost.org/
   (cont.)LICENSE_1_0.txt)

#include <iostream>
#include <saga.hpp>

std::string state_to_string (saga::job::state state)
{
  switch ( state )
  {
    case saga::job::New       : return ("New       ");
    case saga::job::Running   : return ("Running   ");
    case saga::job::Suspended : return ("Suspended");
    case saga::job::Done      : return ("Done      ");
    case saga::job::Failed    : return ("Failed    ");
    case saga::job::Canceled  : return ("Canceled  ");
    default                   :
        std::cout << "Unknown state: " << state << std::endl;
        return ("Unknown   ");
  }
}



//////////////
int main (int argc, char* argv[])
{
    namespace attribs = saga::attributes;

    saga::job_description jd_valid;
```

```
    jd_valid.set_attribute (attribs::
       (cont.)job_description_executable,   "/home/kortas/EX/
       (cont.)ZEPHYR/SRC/zephyr");

    saga::job_service s1("any://localhost");

    saga::job j1 = s1.create_job(jd_valid);
    saga::job j2 = s1.create_job(jd_valid);

    std::cout << "NOT RUN! Job state is: " << state_to_string
       (cont.)(j1.get_state()) << std::endl;

    j1.run();
    std::cout << "JobID: " << j1.get_job_id() << std::endl;
    std::cout << "Just RUN! Job state is: " <<
       (cont.)state_to_string(j1.get_state()) << std::endl;

    j1.suspend();
    std::cout << "SUSPENDED! Job state is: " <<
       (cont.)state_to_string(j1.get_state()) << std::endl;

    j1.resume();
    std::cout << "RESUMED! Job state is: " << state_to_string
       (cont.)(j1.get_state()) << std::endl;


    std::cout << "NOT DONE! Job state is: " <<
       (cont.)state_to_string(j1.get_state()) << std::endl;

    j1.wait();

    std::cout << "AWAITED! Job state is: " << state_to_string
       (cont.)(j1.get_state()) << std::endl;

    sleep(30);
    std::cout << "should be DONE by now! Job state is: " <<
       (cont.)state_to_string(j1.get_state()) << std::endl;

    //j1.cancel();



    j2.run();
    std::cout << "JobID: " << j2.get_job_id() << std::endl;
    std::cout << "Just RUN! Job state is: " <<
       (cont.)state_to_string(j2.get_state()) << std::endl;

    j2.cancel();
    std::cout << "CANCELED! Job state is: " <<
       (cont.)state_to_string(j2.get_state()) << std::endl;

    return 0;
}
```

using the following makefile

```
#  Copyright (c) 2005-2006 Andre Merzky (andre@merzky.net)
SAGA_ROOT=/home/kortas/SOFTS/BUILD/saga-c++-0.6-src

SAGA_SRC            = main.cpp
SAGA_BIN            = main

SAGA_ADD_BIN_OBJ = $(SAGA_SRC:%.cpp=%.o)

include $(SAGA_ROOT)/make/saga.application.mk
```

4. Finally run the test program

```
export LD_LIBRARY_PATH=/home/kortas/SOFTS/saga-06/lib:\
    /home/kortas/SOFTS/boost_34_1/lib:$LD_LIBRARY_PATH:
export SAGA_LOCATION=/home/kortas/SOFTS/saga-06/
./main
```

Contingencies

- After compiling Boost, we also need to manually add an additional link, otherwise SAGA would not link properly with boost searching the library libboost_serialization-gcc41-mt-d-1_34_1.so that did not exist.
- In order to run properly the example bash_correction in /home/kortas/SOFTS/BUILD/saga-c++-0.6-src/example, we need to compile the package logical_file in /home/kortas/SOFTS/BUILD/0.6/examples/packages/logical_file: and copy by hand the libsaga_package_logicalfile.so into the /home/kortas/SOFTS/saga-06/lib directory.

3.5.6 Test log

Test log identifier: wp31-api-ts01-zephyr01-tl01

Description

The tests have been done in the hardware and software environment described in wp31-api-ts01-tp

Activity and Event Entries

Execution Description The tests described here have been conducted by Samuel Kortas (EDF). The execution procedure is described in specification wp31-api-ts01-zephyr01-tds01.

Procedure Results Installation and set up of the boost and saga libraries did not reveal any other contingencies than the ones presented in wp31-api-ts01-zephyr01-tps01. Summary: The installation has been completed successfully.

Execution of the test program

Calling the test program several times, it produced the following output :

```
NOT RUN! Job state is: New
JobID: [any://localhost]-[rendvous-fn1:9205]
Just RUN! Job state is: Running
SUSPENDED! Job state is: Suspended
RESUMED! Job state is: Running
NOT DONE! Job state is: Running
AWAITED! Job state is: Running
should be DONE by now! Job state is: Running
JobID: [any://localhost]-[rendvous-fn1:9385]
Just RUN! Job state is: Running
CANCELED! Job state is: Canceled
```

or this one alternatively

```
NOT RUN! Job state is: New
JobID: [any://localhost]-[rendvous-fn1:26007]
Just RUN! Job state is: Running
SUSPENDED! Job state is: Suspended
RESUMED! Job state is: Running
NOT DONE! Job state is: Running
terminate called after throwing an instance of 'boost::process::
   (cont.)system_error'
  what(): boost::process::child::wait: waitpid(2) failed: No
    (cont.)child processes
Abandon
```

We checked that the job submitted was actually running on the machine by browsing the results produced by Zephyr stored into a "fort.66" produced in the same directory as the test program.

Anomalous Events

- The "exception error" encountered in the alternative result obtained could be explained by the termination of the executed job before probing its status. The job disappeared, its status is therefore not readable any more and throwing the exception.
  Still this behavior does not agree with the Saga specification stating that the status of the terminated job should have been "Done".

- In the "regular" output, the job status also remains "running" after a call to the wait() method. In fact in agreement with the Saga specification, we understand that the status should either be "canceled", "done" or "failed".

Summary: Questions on these two points have been forwarded to the WP 3.1 team. But, for the moment we consider **the test as failed**. Still, though the tests could not be run through the XtreemOS execution environment (SAGA API has not been ported on XtreemOS yet), one could easily prepare, start, monitor, end or cancel a job, which is very encouraging.

3.5.7 Test incident report – Job status reporting problem

Test incident report identifier: wp31-api-ts01-zephyr01-tir01

Summary

SAGA API reporting of job status gave unexpected result.

Incident Description

Using SAGA API on a Linux system, the status of an ended job either can not be probed without throwing an exception, either gives a wrong answer "Running" whereas it should be "Done", "Canceled" or "failed" according to the SAGA specification.

Impact

Could not continue with test.

3.5.8 Test case specification – File Management

Test case specification identifier: wp31-api-ts01-zephyr01-tcs02

Test Items

We will test the behavior of the SAGA filesystem API by creating, renaming, copying and deleting files of different sizes in a set of directories on the target machine using the saga API.

Input Specifications

Input is generated by the test program itself. No additional input is required.

Output Specifications

The test program prints its output on the standard output stream reproduced in the following of this document.

Environmental Needs

Software A simple program has been used as a test program. The hardware and software testing environment is the one described in wp31-api-ts01-tp

3.5.9 Test procedure specification

Test procedure specification identifier: wp31-api-ts01-zephyr01-tps02

Purpose

This procedure executes the test case wp31-api-ts01-zephyr01-tcs02.

Procedure Steps

Set Up The generic test procedure setup includes:

- Installation and compilation of SAGA.
- Compiling and running the test program

Installation and compilation of SAGA on Debian Linux etch distribution including the following steps:

1. Download of Boost 1.34.1
2. Compilation of Boost and manual linking of a shared library.
3. Checking out the 0.6 release of SAGA.
4. Compilation of SAGA.

Start In order to start the test one needs to perform the following steps:

1. Installation and compilation of SAGA on Debian Linux etch distribution : these steps are described in the previous test plan wp31-api-ts01-zephyr01-tps01.
2. Compile the following simple test program that performs file operation:

```
#include <iostream>
#include <saga.hpp>

saga::file  write_file(saga::directory dir,char *name, int
   (cont.)size) {
  char*
 something="**********...************;

  saga::file file;

  file=dir.open(name,saga::file::Write);
```

```
  for (int s;size>0;size-=1024) {
    s=1024;
    if (size<1024) {
      s=size;
    }
     file.write(something,1);
  }
  file.close();

  return file;
}



//////////////////////////////////
int main (int argc, char* argv[])
{

  printf ("\n\n\n cleaning Working directory....\n");
  system("rm -rf /tmp/SAGA");

   // creation of SAGA dir
  printf ("\n\n\n creating directory tree....\n");
  saga::directory dir("any://localhost/tmp/SAGA/",saga::file
     (cont.)::ReadWrite|saga::file::Create|saga::file::
     (cont.)CreateParents);
  saga::directory dir1k("any://localhost/tmp/SAGA/sup1k",
            saga::file::ReadWrite|saga::file::Create|saga::
               (cont.)file::CreateParents);
  saga::directory dir1M("any://localhost/tmp/SAGA/sup1k/sup1M
     (cont.)",
            saga::file::ReadWrite|saga::file::Create|saga::
               (cont.)file::CreateParents);
  saga::directory dir5M("any://localhost/tmp/SAGA/sup1k/sup1M
     (cont.)/sup5M",
            saga::file::ReadWrite|saga::file::Create|saga::
               (cont.)file::CreateParents);

  system("ls -Rl /tmp/SAGA");

  // create file of different size
  printf ("\n\n\n creating files....\n");
  saga::file file=write_file(dir,"es.txt",5);
  write_file(dir1k,"es1k.txt",1024);
  write_file(dir1M,"es1M.txt",1024000);
  write_file(dir5M,"es5M.txt",1024000*5);
  write_file(dir5M,"es100M.txt",1024000*100);
  system("ls -Rl /tmp/SAGA");



  //copy it
  printf ("\n\n\n copying file SAGA/es.txt to SAGA/as.txt
     (cont.)....\n");
```

```
    file.copy("/tmp/SAGA/as.txt");
    system("ls -Rl /tmp/SAGA");

    // rename it
    printf ("\n\n\n renaming file SAGA/es.txt in SAGA/es2.txt
       (cont.)....\n");
    file.move("/tmp/SAGA/es2.txt");
    system("ls -Rl /tmp/SAGA");

    // erase copy
    printf ("\n\n\n deleting file SAGA/as.txt ....\n");
    saga::file file2("/tmp/SAGA/as.txt");
    file2.remove();
    system("ls -Rl /tmp/SAGA");

    return 0;
}
```

using the following makefile

```
#  Copyright (c) 2005-2006 Andre Merzky (andre@merzky.net)
SAGA_ROOT=/home/kortas/SOFTS/BUILD/saga-c++-0.6-src

SAGA_SRC           = main.cpp
SAGA_BIN           = main

SAGA_ADD_BIN_OBJ = $(SAGA_SRC:%.cpp=%.o)

include $(SAGA_ROOT)/make/saga.application.mk
```

3. Finally run the test program

```
export LD_LIBRARY_PATH=/home/kortas/SOFTS/saga-06/lib:\
   /home/kortas/SOFTS/boost_34_1/lib:$LD_LIBRARY_PATH:
export SAGA_LOCATION=/home/kortas/SOFTS/saga-06/
./main
```

As a comparison element, here is the equivalent shell script.

```
echo cleaning Working directory....
\rm -rf /tmp/SAGA

echo
echo
echo creating directory tree....
mkdir -p /tmp/SAGA/sup1k/sup1M/sup5M
ls -l -R /tmp/SAGA

# creation source file
python -c "print '*'*1024*100" > /tmp/source

# creation 10, 1k, 1M and 5M files
```

```
echo
echo
echo creating files...

dd count=1 bs=1024 if=/tmp/source of=/tmp/SAGA/es.txt
dd count=1 bs=1k if=/tmp/source of=/tmp/SAGA/sup1k/es1k.txt
dd count=1 bs=1M if=/tmp/source of=/tmp/SAGA/sup1k/sup1M/es1M.
   (cont.)txt
dd count=1 bs=5M if=/tmp/source of=/tmp/SAGA/sup1k/sup1M/sup5M/
   (cont.)es5M.txt

ls -l -R /tmp/SAGA

echo
echo
echo copyingăfile SAGA/es.txt to SAGA/as.txt....
cp /tmp/SAGA/es.txt /tmp/SAGA/as.txt
ls -l -R /tmp/SAGA

echo
echo
echo renamingăfile SAGA/es.txt in SAGA/es2.txt....
mv /tmp/SAGA/es.txt /tmp/SAGA/es2.txt
ls -l -R /tmp/SAGA

echo
echo
echo deletingăfile SAGA/as.txt
rm /tmp/as.txt

ls -l -R /tmp/SAGA
```

3.5.10 Test log

Test log identifier: wp31-api-ts01-zephyr01-tl02

Description

This test has been done in the hardware and software environment described in wp31-api-ts01-tp

It consists in using SAGA API to handle files (creation, renaming, copy, or deletion) in a directory tree.

Activity and Event Entries

The tests described here have been conducted by Samuel Kortas (EDF). The execution procedure is described in specification wp31-api-ts01-zephyr01-tps02.

Execution of the test program

Calling the test program several times, after a “touch es.txt”, the following correct output is always produced:

```
 cleaning Working directory....
 creating directory tree....
/tmp/SAGA:
total 4
drwxr-xr-x 3 kortas edf 4096 2007-12-13 10:23 sup1k

/tmp/SAGA/sup1k:
total 4
drwxr-xr-x 3 kortas edf 4096 2007-12-13 10:23 sup1M

/tmp/SAGA/sup1k/sup1M:
total 4
drwxr-xr-x 2 kortas edf 4096 2007-12-13 10:23 sup5M

/tmp/SAGA/sup1k/sup1M/sup5M:
total 0

 creating files....
/tmp/SAGA:
total 8
-rw-r--r-- 1 kortas edf    1 2007-12-13 10:23 es.txt
drwxr-xr-x 3 kortas edf 4096 2007-12-13 10:23 sup1k

/tmp/SAGA/sup1k:
total 8
-rw-r--r-- 1 kortas edf    1 2007-12-13 10:23 es1k.txt
drwxr-xr-x 3 kortas edf 4096 2007-12-13 10:23 sup1M

/tmp/SAGA/sup1k/sup1M:
total 8
-rw-r--r-- 1 kortas edf 1000 2007-12-13 10:23 es1M.txt
drwxr-xr-x 2 kortas edf 4096 2007-12-13 10:23 sup5M

/tmp/SAGA/sup1k/sup1M/sup5M:
total 112
-rw-r--r-- 1 kortas edf 100000 2007-12-13 10:23 es100M.txt
-rw-r--r-- 1 kortas edf   5000 2007-12-13 10:23 es5M.txt

 copying file SAGA/es.txt to SAGA/as.txt....
/tmp/SAGA:
total 12
-rw-r--r-- 1 kortas edf    1 2007-12-13 10:23 as.txt
-rw-r--r-- 1 kortas edf    1 2007-12-13 10:23 es.txt
drwxr-xr-x 3 kortas edf 4096 2007-12-13 10:23 sup1k

/tmp/SAGA/sup1k:
total 8
-rw-r--r-- 1 kortas edf    1 2007-12-13 10:23 es1k.txt
drwxr-xr-x 3 kortas edf 4096 2007-12-13 10:23 sup1M
```

```
/tmp/SAGA/sup1k/sup1M:
total 8
-rw-r--r-- 1 kortas edf 1000 2007-12-13 10:23 es1M.txt
drwxr-xr-x 2 kortas edf 4096 2007-12-13 10:23 sup5M

/tmp/SAGA/sup1k/sup1M/sup5M:
total 112
-rw-r--r-- 1 kortas edf 100000 2007-12-13 10:23 es100M.txt
-rw-r--r-- 1 kortas edf   5000 2007-12-13 10:23 es5M.txt

 renaming file SAGA/es.txt in SAGA/es2.txt....
/tmp/SAGA:
total 12
-rw-r--r-- 1 kortas edf    1 2007-12-13 10:23 as.txt
-rw-r--r-- 1 kortas edf    1 2007-12-13 10:23 es2.txt
drwxr-xr-x 3 kortas edf 4096 2007-12-13 10:23 sup1k

/tmp/SAGA/sup1k:
total 8
-rw-r--r-- 1 kortas edf    1 2007-12-13 10:23 es1k.txt
drwxr-xr-x 3 kortas edf 4096 2007-12-13 10:23 sup1M

/tmp/SAGA/sup1k/sup1M:
total 8
-rw-r--r-- 1 kortas edf 1000 2007-12-13 10:23 es1M.txt
drwxr-xr-x 2 kortas edf 4096 2007-12-13 10:23 sup5M

/tmp/SAGA/sup1k/sup1M/sup5M:
total 112
-rw-r--r-- 1 kortas edf 100000 2007-12-13 10:23 es100M.txt
-rw-r--r-- 1 kortas edf   5000 2007-12-13 10:23 es5M.txt

 deleting file SAGA/as.txt ....
/tmp/SAGA:
total 8
-rw-r--r-- 1 kortas edf    1 2007-12-13 10:23 es2.txt
drwxr-xr-x 3 kortas edf 4096 2007-12-13 10:23 sup1k

/tmp/SAGA/sup1k:
total 8
-rw-r--r-- 1 kortas edf    1 2007-12-13 10:23 es1k.txt
drwxr-xr-x 3 kortas edf 4096 2007-12-13 10:23 sup1M

/tmp/SAGA/sup1k/sup1M:
total 8
-rw-r--r-- 1 kortas edf 1000 2007-12-13 10:23 es1M.txt
drwxr-xr-x 2 kortas edf 4096 2007-12-13 10:23 sup5M

/tmp/SAGA/sup1k/sup1M/sup5M:
total 112
-rw-r--r-- 1 kortas edf 100000 2007-12-13 10:23 es100M.txt
-rw-r--r-- 1 kortas edf   5000 2007-12-13 10:23 es5M.txt
```

```
real    0m11.977s
user    0m6.108s
sys  0m5.256s
```

We checked that the file produced (as.txt) was the copy of the initial file (es.txt). Results obtained are the same as if the equivalent shell script wp31-api-ts01-zephyr01-tps02 is run.

The observed time is always around 12 seconds to run the C++ test application on this system.

Summary: We consider the test as passed. Using Saga, one could easily create, rename, delete a file.

3.5.11 Test summary report – Evaluation of the SAGA API

Test summary report identifier: wp31-api-ts01-tsr

Although the SAGA API component has not been released yet for XtreemOS, this first set of tests gives a first impression of the job and file management interfaces expected.

Summary

The 0.6 release of the SAGA-A C++ API has been evaluated

- with a test application managing jobs running the EDF application Zephyr.
- with a simple program managing files. It is expected that the EDF applications ZEPHYR, MODERATO or SIMEON can be suitably adapted to the SAGA file management interface in order to address files on XtreemOS.

At this time, the tests could only cover job and file management and just have an informative value because the SAGA for XtreemOS has not been released yet. However, the installation of SAGA and the execution of the test programs have allowed us to gain first insights into the concepts.

During the installation of SAGA and Boost using autotools, some minor operations have to be done manually (see wp31-api-ts01-zephyr01-tps01):

- renaming of a boost shared library
- copy of saga library in the library path not mentioned in documentation

The need for these manual operations is not well-identified yet and may be due to the environment of the test machine. This issue will be further investigated when testing the next release of SAGA.

A problem was also forwarded to the SAGA team concerning a wrong status reported for an ended job. This referred to an already identified bug in SAGA.

Comprehensiveness Assessment

The testing process was in line with the approach given in wp31-api-ts01-tp.

Evaluation

Below we present a list of XtreemOS-related requirements and their fulfillment status. We also list requirements that are not implemented yet, and therefore had not been tested.

- R48 not fulfilled, since not implemented yet, but SAGA also supports GridFTP. Note that GAT support is currently implemented in the 0.7 version of SAGA and that the job management SAGA API reflects the DRMAA API very closely.
- R50 not fulfilled, since not implemented yet. Still the SAGA specification makes several references to POSIX. The full POSIX compliance still needs to be evaluated.
- R51 not fulfilled, since not implemented yet, but a C++ API is already available for the execution on a single Linux machine.
- R52 not fulfilled, since not implemented yet.

In the next release of this document, more extended tests will be performed hopefully on the integrated version of XtreemOS.

3.6 Evaluation of XtreemFS

3.6.1 Test plan – XtreemFS

Test plan identifier: wp34-xfs-ts01-tp

Introduction

This test plan covers the distributed file system XtreemFS developed by WP3.4. XtreemFS functionalities include:

- high-performance distributed file system for federated installations across multiple organizations
- fully POSIX-compliant with extensions
- suitable for Wide Area Networks (WANs) with high latencies between sites
- suitable for environments with complex failure cases like network partitioning and similarities between slow and dead nodes

- support for replication and partitioning of metadata servers, and replication and striping at file/object level
- integration into Virtual Organizations
- self-monitoring and autonomous optimization of file distribution, layout and access
- transparent object sharing service

Test Items

The software to be tested is XtreemFS. Publications are available on the XtreemFS website: www.xtreemfs.org.

The XtreemFS version tested is the internal release 1 from 2007-07-12. Source and documentation are available from the internal XtreemOS SVN.

The MaxDB distribution and its documentation is available under https://www.sdn.sap.com/irj/sdn/maxdb.

Information about Wissenheim can be found under: http://www.wissenheim.de.

Information about GRID superscalar can be found at http://www.bsc.es/gridsuperscalar. Information about fastDNAml can be found at http://geta.life.uiuc.edu/~gary/programs/fastDNAml.html.

Features to be Tested

The following features will be tested:

- POSIX compliance (open, read, write, close, ls, rm, touch, mv, cp, mkdir, cd, rmdir)
- stability
- scalability
- runtime of application benchmarks
- integrity of stored data
- preliminary test on I/O performance

Note that internal release 1 is an alpha version which is not yet optimized for performance. Therefore, it is expected that I/O performance tests provide results which will improve in later versions of XtreemFS.

Features not to be Tested

The following features will not be tested as they are not supported by the internal release:

- striping policies
- resilience to failure cases like network partitioning etc.
- support for replication and partitioning of metadata servers
- integration into Virtual Organizations

Approach

The purpose of the early test is to provide feedback to developers about the implemented features and about the fulfillment of requirements.

Tests are performed by installing XtreemFS on testbeds of local machines and executing GSDNA (provided by BSC), the database maxDB (provided by SAP) and WISS (provided by UDUS). Initial performance tests are executed using the Bonnie benchmark utility. It was decided to choose maxDB as reference application since in typical multi-tier business solutions the great majority of file operations is transactional relying on a centric database. In practical business scenarios, end-user applications access this database via the middleware WEBAS. In the experiments with XtreemFS, it was decided to stress the file system via the maxDB benchmark mut32 from SAP which also allows to trigger multi-user access with parallel read and write operations. Whereas the SAP experiments replay a database-centric scenario, the experiments with GSDNA and WISS are file-based with WISS accessing the file system via the GOM layer.

Item Pass Criteria

XtreemFS will pass a test if:

- running application under the condition defined by the test design specification does not crash the application, nor the file system nor any other system.
- an application run on top of XtreemFS, under the condition of the test, behaves as if it was launched on a common system using NFS.
- no integrity violation of stored data is reported

Testing Tasks

The testing tasks include:

- building and installing XtreemFS

- running basic tests including standard commandline file operations
- building and installing file system benchmarks
- evaluating XtreemFS by means of file system benchmarks
- building and installing the applications (with application benchmarks if applicable)
- evaluate XtreemFS by means of applications and application-specific benchmarks

Environmental Needs

XtreemFS has no special hardware requirements. It could be executed on commodity PCs with e.g. 1 GB RAM, and 2 GHz CPU clock speed. The total disk storage required largely depends on the amount of application and user data. It is possible to install XtreemFS on a single node, however, to make use of file storage in distributed OSDs one should consider a testbed with at least two PCs. XtreemFS is designed to be independent of any specific Linux distribution but requires preinstalled third-party software and libraries including:

- gmake 3.8.1
- gcc 4.1.2
- Java Development Kit 1.6
- Apache Ant 1.6.5
- FUSE 2.6
- libxml2-dev 2.6.26
- openssl-dev 0.9.8

Responsibilities

This test plan and the included tests with maxDB are the responsibility of SAP. The tests with GSDNA and WISS are the responsibility of BSC or UDUS respectively.

Schedule

Since XtreemFS is in an early development status, the setup of a running system still requires many manual installations also including the above-mentioned third party software and libraries. Test durations may be unpredictable due to the early release used and also depend on the local test configuration with the various application references. All the planned tests should be finished in time for XtreemOS deliverable D4.2.4.

Risks and Contingencies

Apart from delays in testing caused by unexpected installation and execution problems, no specific risks are envisaged.

3.6.2 Test design specification

Test design specification identifier: wp34-xfs-ts01-maxdb01-tds01
Test plan reference: wp34-xfs-ts01-tp

Features to be Tested

The following features and requirements are the subject of this test design specification:

- creating directories using **mkdir** Unix command: R50, R76
- copying files using **cp** Unix command (the existing MaxDB data files are copied from local file system to XtreemFS volume): R50
- opening/reading/writing/closing existing files (MaxDB data files) using POSIX/UNIX system calls: R1, R50 , R79

Approach Refinements

This test includes a run of the **mut32** benchmark that stresses a MaxDB installation. Using some tuning parameters that reduce the MaxDB main memory cache size it is possible to make MaxDB issue a considerable amount of I/O to the data files (called MaxDB volumes) during the **mut32** test. Besides a comparison of XtreemFS performance with the performance of other (local/distributed) file systems on OLTP workload, this test allows for an integrity test: In MaxDB, each data page stored in a file has a CRC checksum, such that any file system data corruption is reported immediately. For the preparation of a MaxDB run on XtreemFS one needs to copy the existing volume files created during the MaxDB installation to XtreemFS and symbolically link the MaxDB volume directory to the XtreemFS directory with the copied files.

In order to benchmark other file activities important for practical scenarios we run the Bonnie benchmark that sequentially writes a file (similar to writing a database log file), sequentially reads the written file and performs reads of small random chunk of data. The latter access pattern is present in database file workloads.

Test Identification

This test design specification is covered by test case wp34-xfs-ts01-maxdb01-tcs01 and wp34-xfs-ts01-maxdb01-tcs02.

Feature Pass/Fail Criteria

This test is successful if all of the following holds:

- A directory for MaxDB volume files on XtreemFS has been created successfully.
- The MaxDB files are copied successfully to the XtreemFS directory.
- The MaxDB with volume files on XtreemFS has started successfully.
- The **mut32** benchmark on the running MaxDB has finished without any errors.
- The MaxDB instance has shutdown without any errors after running the **mut32** benchmark.
- The Bonnie benchmark on XtreemFS produces no errors.

3.6.3 Test case specification – Running MaxDB

Test case specification identifier: wp34-xfs-ts01-maxdb01-tcs01

Test Items

This test will transfer the MaxDB volume files to XtreemFS, start the MaxDB instance, run the **mut32** benchmark and shutdown the database.

Input Specifications

The only input needed for the test are the data base volume files, created by the MaxDB installation procedure.

Output Specifications

A text log file with executed SQL commands and (possibly) error messages is created by the **mut32** benchmark script. Additionally, "*.bad" files are created by MaxDB in the run directory if a data page is corrupted.

Environmental Needs

Software The software required for this test:

- 32-bit Linux version of MaxDB 7.6.02.10
- a MaxDB certified Linux distribution like SUSE SLES 10
- glibc library >= 2.3.3, gtk library 2.x, X server (the root user must be able to access a display)
- glibc must support TLS (thread local storage)

3.6.4 Test procedure specification

Test procedure specification identifier: wp34-xfs-ts01-maxdb01-tps01
Test design specification reference: wp34-xfs-ts01-maxdb01-tds01
Test log identifier: wp34-xfs-ts01-maxdb01-tl01

Purpose

This procedure executes the test case wp34-xfs-ts01-maxdb01-tcs01.

Procedure Steps

Set Up The generic test procedure setup includes:

- Installation and compilation of XtreemFS
- Installation of MaxDB
- Installation of **mut32** benchmark script

We have installed XtreemFS on two types of systems:

- SUSE Linux SLES10: executes XtreemFS client
- A remastered KNOPPIX 5.1 LiveCD (Debian-based): executes XtreemFS server code (MRC/DS/OSD services)

Installation and compilation of XtreemFS on SUSE Linux SLES10 included the following steps:

1. Installation of subversion client from SLES10 SDK CDs
2. Checking out the XtreemFS (internal release 1 branch) from its subversion repository
3. Installed and compiled FUSE 2.7 from sources (SLES10 provided only FUSE 2.5)
4. Installed Java 1.6 SDK from `www.sun.com`
5. Downloaded and compiled apache-ant-1.7.0 from `ant.apache.org` (because the one provided by SLES10 ant 1.6.5 did not work)
6. Compiled and installed gnu make 3.8.1 (SLES10 provides only gnu make 3.8.0, which does not work with internal release 1 of XtreemFS)
7. Compilation of XtreemFS

Installation and compilation of XtreemFS on the KNOPPIX included the following steps:

1. Installation of subversion client (subversion package)
2. Checking out the XtreemFS (internal release 1 branch) from it's subversion repository
3. Installation of Java JRE (sun-java5-* packages)
4. Installation of FUSE development files (libfuse-dev package)
5. Downloaded and compiled apache-ant-1.7.0 from ant.apache.org
6. Compilation of XtreemFS

Installation of MaxDB on SLES10:

1. Download the community edition of MaxDB from https://www.sdn.sap.com/irj/sdn/maxdb
2. Follow the MaxDB installation instruction at https://www.sdn.sap.com/irj/sdn/maxdb (Linux version)

The installation of the proprietary **mut32** script on SLES10 includes:

1. Creating a database user (mut32)
2. Adding the database user to the MaxDB database group sdba
3. Unpacking the **mut32** tar archive into /home/mut32/mut32
4. Creating a configuration file for **mut32** that includes database name, database user and database passwords

Start In order to start the test one needs to perform the following steps:

1. Add the following variables to shell/bash startup scripts for the database user mut32:

```
export PATH=$PATH:/opt/sdb/programs/bin/:/home/mut32/mut32
    (cont.)/:./
export SID=MAXDB1
export INSTROOT=/opt/sdb/MAXDB1
```

2. Log on as mut32 user and offload the MAXDB1 database:

```
su -- mut32
cd /home/mut32/mut32
dbmcli -n localhost -d $SID -u dbadmin,password db_offline
```

3. Tune MaxDB decreasing the in-memory cache of MaxDB to 2000 pages (about 16 MB), the maximum number of user tasks to 50, and switching the direct file access off (avoids OS page caching) by creating a change_parameters file and executing the following commands:

```
mut32@client:~/mut32> cat change_parameters
param_startsession
param_put CACHE_SIZE 2000
param_put MAXUSERTASKS 50
param_put USE_OPEN_DIRECT NO
param_checkall
param_commitsession
exit
mut32@client:~/mut32> dbmcli -n localhost -d $SID -u dbadmin,
   (cont.)password -i change_parameters
```

The direct file access in MaxDB must be switched off, since the FUSE client library used by XtreemFS does not support it. Otherwise MaxDB fails accessing the volume data files:

```
mut32@client:~/mut32> dbmcli -n localhost -d MAXDB1 -u
   (cont.)dbadmin,password db_online

ERR
-24988,ERR_SQL: SQL error
-902,I/O error
3,Database state: OFFLINE
6,Internal errorcode, Errorcode 9050 "disk_not_accessible"
20017,RestartFilesystem failed with 'I/O error'
```

4. Tune MaxDB to avoid log file overflows:

```
dbmcli -n localhost -d $SID -u dbadmin,password util_execute
   (cont.)set log auto overwrite on
```

5. As root, stop the database instance executing `x_server stop`

After doing the preparation steps above one can proceed with the main test procedure:

1. Start XtreemFS DS server at KNOPPIX node0:

```
rm -rf /mnt/disk/ds*
xtreemfs_start ds -p 32638 -i -d node0 -c /tmp/ds.cfg -s /mnt
   (cont.)/disk/ds -l /mnt/disk/ds.log
```

2. Start XtreemFS MRC server at KNOPPIX node0:

```
rm -rf /mnt/disk/mrc*
xtreemfs_start mrc -p 32636 -i -d node0 -c /tmp/mrc.cfg -s /
   (cont.)mnt/disk/mrc -l /mnt/disk/mrc.log
```

3. Start XtreemFS OSD servers at KNOPPIX node1-node4:

```
rm -rf /mnt/disk/osd*
xtreemfs_start osd -p 32637 -i -d node0 -c /tmp/osd.cfg -s /
    (cont.)mnt/disk/osd -l /mnt/disk/osd.log
```

4. Create XtreemFS volume at the client node (use RAID0 striping with 4 OSDs and 16 KB stripe size):

```
mkvol --striping-policy=RAID0,16,4 http://node0:32636/
    (cont.)XFS_16_4
```

5. Mount the created volume at the client node (use `direct_io` mount option to emulate direct file access, since the direct access flag for the `open` system call is not supported by the XtreemFS/FUSE client code):

```
mkdir /mnt/XFS_16_4
xtreemfs -o volume_url=http://node0:32636/XFS_16_4,direct_io
    (cont.)/mnt/XFS_16_4
```

6. Transfer MaxDB data files to XtreemFS on the client:

```
mkdir /mnt/XFS_16_4/MaxDB
cp /var/opt/sdb/data/MAXDB1/data/* /mnt/XFS_16_4/MaxDB/
mv /var/opt/sdb/data/MAXDB1/data /var/opt/sdb/data/MAXDB1/
    (cont.)data.backup
ln -s /mnt/XFS_16_4/MaxDB /var/opt/sdb/data/MAXDB1/data
```

7. Log on as root on the client and start MaxDB: `x_server start`

8. Log on as `mut32` user on the client and start the database:

```
su -- mut32
dbmcli -n localhost -d $SID -u dbadmin,password db_online
```

9. As `mut32` user on the client start the measurement script that will run **mut32** benchmark:

```
#!/bin/bash
export LOGFILENAME=mut32_xtreemfs.log
date >> $LOGFILENAME
perl mut.pl -R /opt/sdb/MAXDB1 -nomutsave -noadddelvol -
    (cont.)nodbana 1 >> $LOGFILENAME 2>> $LOGFILENAME
date >> $LOGFILENAME
```

10. As root, shut down MaxDB instance on the client: `x_server stop`

Measure For obtaining the first performance characteristics, we compared the runtime of the **mut32** benchmark running on XtreemFS (internal release 1) with its running time on the local file system. These timings were done with the `date` command and available in the generated `mut32_xtreemfs.log`.

Wrap Up If due to some reason XtreemFS corrupts the MaxDB database instance, one should reinstall MaxDB:

1. Try to shutdown database executing as root: `x_server stop`
2. Delete all MaxDB files executing as root: `rm -rf /opt/sdb`
3. Reinstall MaxDB and configure it repeating the steps above

Contingencies During the compilation of XtreemFS on the KNOPPIX we have encountered the following messages:

```
    [javac] /UNIONFS/root/xtreemfs/java/src/org/xtreemfs/osd/
       (cont.)StorageMedia.java:262: unmappable
    character for encoding ASCII
    [javac]      *  @author Jes??s Malo
    [javac]                    ^
    [javac] /UNIONFS/root/xtreemfs/java/src/org/xtreemfs/osd/
       (cont.)StorageMedia.java:262: unmappable
    character for encoding ASCII
    [javac]      *  @author Jes??s Malo
    [javac]                    ^
    [javac] 48 errors

BUILD FAILED
/UNIONFS/root/xtreemfs/java/nbproject/build-impl.xml:245: The
   (cont.)following error
occurred while executing this line:
/UNIONFS/root/xtreemfs/java/nbproject/build-impl.xml:126: Compile
   (cont.)failed; see
the compiler error output for details.

Total time: 8 seconds
make: *** [java] Error 1
```

In order to solve this problem one should add `encoding="UTF-8"` parameter into the `<javac ...>` line in the `java/nbproject/build-impl.xml` file:

```
<javac srcdir="@{srcdir}" ... > -> <javac encoding="UTF-8" srcdir
   (cont.)="@{srcdir}" ... >
```

It was observed that the internal release of XtreemFS fails if the number of physical OSD nodes does not match the striping width of the volume (the default width is one):

```
mkvol http://node0/Default
mkdir /mnt/XFS_Default
xtreemfs /mnt/XFS_Default -o volume_url=http://node0:32636/Default
   (cont.), direct_io
cd /mnt/XFS_Default
echo TEST > TEST
TEST: Input/output error
```

This bug has been fixed in the later revisions of XtreemFS (e.g in the `trunk` version available via subversion at
https://scm.gforge.inria.fr/svn/xtreemos/WP3.4/trunk)

3.6.5 Test log

Test log identifier: wp34-xfs-ts01-maxdb01-tl01

Description

Our tests have been done with the XtreemFS internal release 1 available in XtreemOS svn repository at https://scm.gforge.inria.fr/svn/xtreemos/WP3.4/branches/internal_release_1. The svn revision being tested here has number 536 with the last modification 2007-07-12 12:08:22 -0400 (Thu, 12 Jul 2007).

Our test bed included:

- 1 node running MRC/DS servers. CPU: Dual Pentium 4 3.2 Ghz, RAM: 1 GB, hard disk(s): 1x WDC WD800JD-23JNC0, OS: KNOPPIX 5.1 Live-CD Linux
- 4 nodes running OSD servers. CPU: Dual Pentium 4 3.2 Ghz, RAM: 2 GB, hard disk(s): 1x WDC WD800JD-23JNC0, OS: KNOPPIX 5.1 Live-CD Linux
- 1 node running mut32/MaxDB/XtreemFS client: CPU: Pentium M 1.6 Ghz, RAM: 1 GB, hard disk(s): 1x HTS548040M9AT00, OS: SUSE Linux SLES10

All nodes were connected with a 1GBit Ethernet switch.

Activity and Event Entries

Execution Description The tests described here have been conducted by Roman Dementiev (SAP). The execution procedure is described in specification wp34-xfs-ts01-maxdb01-tds01.

Procedure Results During the mut32/MaxDB tests we have observed the following messages:

- Starting MaxDB (success), ignore error messages in first three lines:

```
ERR 11779  NISERVER Error during startup:'dlopen failed:/opt/
    (cont.)sdb/programs/lib/libsap'
WNG 12457  XSERVER  NI Init: Automatic niserver start failed,
    (cont.) rc = 1
WNG 12453  NISSLSRV NISSL Init: dlopen failed:/opt/sdb/
    (cont.)programs/lib/libsap
```

```
    12902 XSERVER started, 'X32/LINUX 7.6.02    Build
      (cont.)010-121-148-818'
```

- Starting the database (success):

```
OK
```

- Output of the **mut32** benchmark script (success):

```
Fri Aug 24 17:29:23 CEST 2007
    12916 XSERVER Found other running x_server with version
      (cont.) 'X32/LINUX 7.6.02    Build 010-121-148-818'
    12902 XSERVER started, 'already...'
mut without mutsave.pl - 'set log auto overwrite on' to avoid
   (cont.) LOG FULL problem ...
mut drop old and create new users ... done
mut Initialize tables ...
DROP TABLE TAB1
['drop' entries for other tables are cut in this listing due
   (cont.)to space constrains]
Commit
CREATE TABLE TAB1
['create' entries for other tables are cut in this listing
   (cont.)due to space constrains]
Commit
GRANT ... ON MAXRAW
GRANT ... ON TAB1
['grant' entries for other tables are cut in this listing due
   (cont.) to space constrains]
Commit
ALTER TABLE TAB1 DROP PRIMARY KEY
ALTER TABLE TAB1 ADD PRIMARY KEY (K0,K1)
ALTER TABLE TAB1 FOREIGN KEY ...
CREATE VIEW TABV1
GRANT ... ON TABV1
Commit
CREATE INDEX TAB1_K1 ON TAB1(K1)
CREATE INDEX TAB1_I3 ON TAB1(I3)
['grant' entries for other tables are cut in this listing due
   (cont.) to space constrains]
Commit
INSERT MAXRAW
Commit
INSERT TAB4
GRANT ... ON TAB4
INSERT TAB_READONLY
Commit
DROP/CREATE/INSERT TABJ_1
[entries for TABJ_2 to TABJ_64 are cut in this listing due to
   (cont.) space constrains]
DROP VIEW BVIEW_K_P
CREATE VIEW BVIEW_K_P
GRANT SELECT ON BVIEW_K_P TO PUBLIC
```

```
[drop/create/grant entries for other views are cut in this
   (cont.)listing due to space constrains]
UPDATE STATISTICS *
UPDATE STATISTICS COLUMN (*) FOR TAB1
Commit
get TOP 10 Allocators
—— ALLOCATORSTATISTICS ——
TOP 1: allocator = 'SystemHeap', used_bytes = '330821632',
   (cont.)maxused_bytes = '330821632'
TOP 2: allocator = 'StackSpace', used_bytes = '206557184',
   (cont.)maxused_bytes = '206557184'
TOP 3: allocator = 'SystemPageCache', used_bytes =
   (cont.)'122494976', maxused_bytes = '122494976'
TOP 4: allocator = 'RTEMem_BlockAllocator', used_bytes =
   (cont.)'121896960', maxused_bytes = '121896960'
TOP 5: allocator = 'LVCMem_BlockAllocator', used_bytes =
   (cont.)'10489856', maxused_bytes = '10489856'
TOP 6: allocator = 'RTEMem_Allocator', used_bytes =
   (cont.)'3021056', maxused_bytes = '3021056'
TOP 7: allocator = 'CommandCacheAllocator', used_bytes =
   (cont.)'1146696', maxused_bytes = '1146696'
TOP 8: allocator = 'RTEMem_RteAllocator', used_bytes =
   (cont.)'357624', maxused_bytes = '357624'
TOP 9: allocator = 'CatalogCache', used_bytes = '324088',
   (cont.)maxused_bytes = '324088'
TOP 10: allocator = 'TransContext T154', used_bytes =
   (cont.)'176800', maxused_bytes = '176800'
—— END of ALLOCATORSTATISTICS ——
In the background MUT has started - to support the MUT
Please see corresponding '*.prt'-files.
Now MUT will try to stop all background tasks.
mut was started with arguments '/opt/sdb/MAXDB1'
mut ran '19799520' minutes.
End of test at Fri Aug 24 18:00:57 2007
Fri Aug 24 18:00:57 CEST 2007
```

- Shutting down MaxDB (success):

```
    12905   XSERVER   stopped
```

Summary: The tests have been completed successfully, no MaxDB page corruptions were found. The running time of the **mut32** benchmark on XtreemFS with 4 OSD nodes was about 32 minutes. During the tests it was observed that XtreemFS client occupied an amount of main memory that was enough to hold the whole MaxDB data set. Running the test on local file system took about 25 minutes with direct access (uses only the 16MB MaxDB cache), and about 4 minutes if the local file system cache is used additionally (the whole MaxDB data is then hold in the 1GB RAM). Additional tests have shown that doubling/halving the stripe width together with the number of XtreemFS OSD nodes only changed the running time insignificantly (also with a single OSD).

3.6.6 Test case specification – Running Bonnie benchmark

Test case specification identifier: wp34-xfs-ts01-maxdb01-tcs02

Test Items

This test will run the Bonnie benchmark on XtreemFS.

Input Specifications

The test data is generated by the Bonnie benchmark.

Output Specifications

The Bonnie benchmark outputs the bandwidths and I/O per seconds achieved for the file operations to the console.

Environmental Needs

Hardware No special hardware is required.

Software The Bonnie benchmark requires a UNIX/POSIX compatible environment.

3.6.7 Test procedure specification

Test procedure specification identifier: wp34-xfs-ts01-maxdb01-tps02
Test design specification reference: wp34-xfs-ts01-maxdb01-tds01
Test log identifier: wp34-xfs-ts01-maxdb01-tl02

Purpose

This procedure executes the test case wp34-xfs-ts01-maxdb01-tcs02.

Procedure Steps

Set Up The generic test procedure setup includes:

- Installation and compilation of XtreemFS (see test procedure specification wp34-xfs-ts01-maxdb01-tps01)
- Installation of the Bonnie benchmark

The installation of the Bonnie benchmark on SLES10 includes:

1. Checkout the latest Bonnie version from the subversion repository:

```
svn checkout http://bonnie-64.googlecode.com/svn/trunk/ /root
    (cont.)/bonnie
```

2. Compile the Bonnie benchmark:

```
cd /root/bonnie
make
export PATH=$PATH:/root/bonnie
```

Start

1. Start XtreemFS as described in the test procedure specification wp34-xfs-ts01-maxdb01-tps01.

2. To test XtreemFS on the client, run the following command:
   `./Bonnie -d /mnt/XFS_16_4`

3. To test the local file system, run the following command:
   `./Bonnie -d /root`

Measure The output of the Bonnie benchmark will be analyzed. We look at the write bandwidth using block access.

3.6.8 Test log

Test log identifier: wp34-xfs-ts01-maxdb01-tl02

Description

The hardware and software settings used in this test are described in the test log wp34-xfs-ts01-maxdb01-tl01.

Activity and Event Entries

Execution Description The tests described here have been conducted by Roman Dementiev (SAP). The execution procedure is described in specification wp34-xfs-ts01-maxdb01-tps02.

Procedure Results During the Bonnie tests we have observed the following messages:

- Running Bonnie on XtreemFS (success):

```
File '/mnt/XFS_16_4/Bonnie.21089', size: 104857600
Writing with putc()...done
Rewriting...done
Writing intelligently...done
Reading with getc()...done
Reading intelligently...done
Seeker 1...Seeker 3...Seeker 2...start 'em...done...done...
    (cont.)done...
                ------Sequential Output------ --Sequential
                    (cont.)Input-- --Random--
                -Per Char- --Block--- -Rewrite-- -Per Char- --
                    (cont.)Block--- --Seeks---
Machine      GB M/sec %CPU M/sec %CPU M/sec %CPU M/sec %CPU M/
    (cont.)sec %CPU   /sec %CPU
             0   0.2  1.1   0.5  0.0   0.6  0.1   0.4  2.2
                 (cont.)1.5  0.1   130  0.3
```

- Running Bonnie on local file system (Bonnie benefits from the file system cache, since its default input size is 200 MB – success):

```
File './Bonnie.12695', size: 104857600
Writing with putc()...done
Rewriting...done
Writing intelligently...done
Reading with getc()...done
Reading intelligently...done
Seeker 1...Seeker 2...Seeker 3...start 'em...done...done...
    (cont.)done...
                ------Sequential Output------ --Sequential
                    (cont.)Input-- --Random--
                -Per Char- --Block--- -Rewrite-- -Per Char- --
                    (cont.)Block--- --Seeks---
Machine      GB M/sec %CPU M/sec %CPU M/sec %CPU M/sec %CPU M/
    (cont.)sec %CPU   /sec %CPU
             0  26.2 100.0 502.4 100.5 470.8 99.8  27.1 100.0
                 (cont.)1326.0 100.8 36075 97.4
```

- Running Bonnie on local file system with a 2GB data set (uses `-s 2048` to limit the use of the file system cache – success):

```
File './Bonnie.12707', size: 2147483648
Writing with putc()...done
Rewriting...done
Writing intelligently...done
Reading with getc()...done
Reading intelligently...done
Seeker 1...Seeker 2...Seeker 3...start 'em...done...done...
    (cont.)done...
                ------Sequential Output------ --Sequential
                    (cont.)Input-- --Random--
                -Per Char- --Block--- -Rewrite-- -Per Char- --
                    (cont.)Block--- --Seeks---
```

```
Machine      GB M/sec %CPU M/sec %CPU M/sec %CPU M/sec %CPU M/
   (cont.)sec %CPU   /sec %CPU
             2  26.9 98.0  33.2 12.6  13.7  5.6  22.3 87.3
                (cont.)33.0  6.8    130  0.8
```

- Running Bonnie on XtreemFS with a 2GB data set (uses `-s 2048` to limit the use of the file system cache – failed):

```
File '/mnt/XFS_16_4/Bonnie.4402', size: 2147483648
Writing with putc()...Bonnie: drastic I/O error (fclose after
   (cont.) putc): Numerical result out of range
```

Summary: Not all tests have been completed successfully. The write speed of XtreemFS (500 KB/s) was significantly slower than of the local file system (33 MB/s). The performance of XtreemFS did not change after doubling/halving the number of OSDs/striping width. Matthias Hess has recommended to recompile XtreemFS with setting `'NO_DEBUG=y'` in XtreemFS configuration. This did not improve the performance. The Bonnie test has failed on XtreemFS volume with the input of 2GBytes after about 10 hours of processing. This failure is reproducible.

Incident Report Identifiers

wp34-xfs-ts01-maxdb01-tir01 describes the problem with the Bonnie test running with a large 2GB input.

3.6.9 Test incident report – Bonnie benchmark with a 2 GByte input failed

Test incident report identifier: wp34-xfs-ts01-maxdb01-tir01

Summary

Tests with the Bonnie benchmark with a 2 GByte input running on the internal release 1 of XtreemFS have failed. Test procedure specification: wp34-xfs-ts01-maxdb01-tps02. Test case specification: wp34-xfs-ts01-maxdb01-tcs02. Test log: wp34-xfs-ts01-maxdb01-tl02.

Incident Description

During the evaluation of XtreemFS (internal release 1) we were not able to run the Bonnie benchmark on large files of 2 GBytes. The benchmark failed attempting to close a large file using the `fclose` system call, which returned an error. The execution of the benchmark till the error took about 10 hours. Another attempt with a complete restart of XtreemFS led to the same error.

Impact

This error has two impacts we consider at the moment:

- Large XtreemFS files can not be processed using the standard C I/O library.
- It is not possible to measure the random access time requirement R79, since for a plausible test, the input data must be large enough to exceed the client-side RAM cache.

3.6.10 Test design specification

Test design specification identifier: wp34-xfs-ts01-gsdna01-tds01
Test plan reference: wp34-xfs-ts01-tp

Features to be Tested

This test performs the following basic file system tests on a mounted XtreemFS volume:

1. Creating files
2. Creating directories
3. Accessing files and directories
4. Executing shell scripts
5. Executing binaries
6. Reading and writing files in a directory from several hosts
7. Deleting files and directories
8. Creating and deleting soft links

These tests partially verify requirements R50, R75, R76, R74, R77, and R85.

Approach Refinements

These tests reproduce the steps that a user would take in order to prepare and execute a GRID superscalar application under an XtreemFS volume. In this case, the application is GSfastDNAml. Each test generates a series of files and directories under inside the volume. The results are validated by comparing the contents of the generated files and the directory listings to those obtained by running the same steps under a non XtreemFS directory.

The first test case verifies file and directory creation. The second test case also performs file and directory creation and verifies accessing files, accessing directories, executing shell scripts, executing binaries, creating soft links and deleting soft

links. The third test case performs equivalent operations to those of the previous tests plus reading files and writing files on the same directory from several hosts simultaneously.

Test Identification

This test design specification is covered by test cases wp34-xfs-ts01-gsdna01-tcs01, wp34-xfs-ts01-gsdna01-tcs02 and wp34-xfs-ts01-gsdna01-tcs03.

Feature Pass/Fail Criteria

Total test compliance may be determined by running all test cases in order and comparing the files generated to those that would be generated under a non XtreemFS directory.

3.6.11 Test case specification – Basic file and directory creation

Test case specification identifier: wp34-xfs-ts01-gsdna01-tcs01

Test Items

This test case tests basic file and directory creation inside an already mounted XtreemFS volume.

Input Specifications

This test case has two inputs: a directory inside an already mounted XtreemFS volume and a source tarball of the GSfastDNAml application.

Output Specifications

The contents of the source tarball should have been created inside the output directory under the ownership of the current user. The files and directories must match those stored in the tarball in size, content and permissions. Correct operation can be checked by replicating the steps on a non XtreemFS directory and comparing the files (size, contents and protection).

Environmental Needs

Software This test case only requires standard utilities from UNIX (a shell, ls and tar) and a standard diff utility.

3.6.12 Test case specification – Building GSfastDNAml application

Test case specification identifier: wp34-xfs-ts01-gsdna01-tcs02

Test Items

This test case consists in building the GSfastDNAml application inside an XfreemFS volume already populated with its source code. It checks file and directory creation, file and directory access, executing shell scripts, executing binaries, creating soft links and deleting soft links.

Input Specifications

This test case has as input a directory inside an XfreemFS volume already populated with the GSfastDNAml source code.

Output Specifications

After running the compilation process, a set of binary programs and scripts should have been created inside the volume.

Environmental Needs

Software This test case requires the following software:

- Standard UNIX utilities
- A POSIX compatible make tool
- A C and C++ compiler
- An installation of GRID superscalar for clusters
- GNU automake
- GNU autoconf

All environment variables related to those tools must be set.

Intercase Dependencies

This test case depends on the results from test case wp34-xfs-ts01-gsdna01-tcs01.

3.6.13 Test case specification – Running GSfastDNAml application

Test case specification identifier: wp34-xfs-ts01-gsdna01-tcs03

Test Items

This test consists in running the GSfastDNAml application inside an XfreemFS volume with several hosts. This application spawns several processes to various hosts, which read and write several files under the same directory. The purpose of this test is to check file and directory creation, file and directory access, shell script execution, binary execution, creating and deleting soft links and concurrent file creation and access under the same directory.

Input Specifications

This test case has as input a directory inside an XfreemFS volume already populated with the GSfastDNAml application and a series of input files for the application.

Output Specifications

The application generates an output file containing the results of the analysis. An overview of the results is written to the terminal. Result correctness is checked by running the sequential version of the application outside of XtreemFS with the same input data and comparing the results files of both applications with the diff utility.

Environmental Needs

Hardware This test case requires having one or more hosts (possibly virtual) with the target architecture of the application and enough memory. The application architecture is the one of the compiler used in the test case specification wp34-xfs-ts01-gsdna01-tcs02, that by default is the one of the host where the test case has been run on. Memory requirements depend on the GSfastDNAml input file but normally do not exceed 64MB per CPU.

Software This test case requires the following software:

- An installation of GRID superscalar for clusters
- An installed and configured GSfastDNAml binary

All environment variables related to those tools must be set.

Intercase Dependencies

This test case depends on the results from test case wp34-xfs-ts01-gsdna01-tcs02.

3.6.14 Test procedure specification

Test procedure specification identifier: wp34-xfs-ts01-gsdna01-tps01
Test log identifier: wp34-xfs-ts01-gsdna01-tl01
Test design specification reference: wp34-xfs-ts01-gsdna01-tds01

Purpose

This is the test procedure to execute wp34-xfs-ts01-gsdna01-tcs01.

Procedure Steps

Log A log of the operations performed by the different XtreemFS server components may be generated during the execution of the tests. The file location is determined by how the XtreemFS server components have been started.

Set Up To set up the environment, the XtreemFS services must have been installed and started in one or more server hosts and contain an already created volume. This volume must be mounted in the host where the test is to be performed. Those procedures are explained in detail in the XtreemFS User Manual.

Proceed

- Make the current directory point to the target directory

```
cd <volume mount point>
```

- Extract the tarball

```
tar -xpf <source code tarball>
```

- Check that no errors were produced
- Extract the tarball into a non XtreemFS directory

```
(cd <reference directory> && tar -xpf <source code tarball>)
```

- Compare the contents of both directory trees

```
diff -urN <reference directory> .
```

- Compare file and directory permissions of both directory trees

```
ls -lR > /tmp/ls-pwd
ls -lR <reference directory> > /tmp/ls-ref
diff -u /tmp/ls-ref /tmp/ls-pwd
rm -f /tmp/ls-pwd /tmp/ls-ref
```

3.6.15 Test procedure specification

Test procedure specification identifier: wp34-xfs-ts01-gsdna01-tps02
Test log identifier: wp34-xfs-ts01-gsdna01-tl02

Test design specification reference: wp34-xfs-ts01-gsdna01-tds01

Purpose

This is the test procedure to execute wp34-xfs-ts01-gsdna01-tcs02.

Procedure Steps

Log A log of the operations performed by the different XtreemFS server components may be generated during the execution of the tests. The file location is determined by how the XtreemFS server components have been started.

Set Up To set up the environment, the XtreemFS services must have been installed and started in one or more server hosts and contain an already created volume. This volume must be mounted in the host where the test is to be performed. Those procedures are explained in detail in the XtreemFS User Manual.

Furthermore, all steps described in wp34-xfs-ts01-gsdna01-tps01 must be performed in order to install the source code in the volume.

Proceed

- Enter the master code directory

```
cd <volume mount point>/master
```

- Prepare the source code for compilation

```
cp Makefile.am Makefile.am.was
gsbuild copy master GSfastDNAml
cp Makefile.am.was Makefile.am
```

- Run the 'autogen.sh' script

```
./autogen.sh
```

- Build the master binary

```
make
```

- Enter the worker directory

```
cd <volume mount point>/worker
```

- Prepare the source code for compilation

```
cp Makefile.am Makefile.am.was
gsbuild copy worker GSfastDNAml
cp Makefile.am.was Makefile.am
```

- Run the 'autogen.sh' script

```
./autogen.sh
```

- Build the worker binary

```
make
```

- Configure the master part. This may require hand tuning by editing the 'project.gsdeploy' file.

```
cd <volume mount point>/master
config_master.sh interactive --master-dir=<volume mount point
    (cont.)>/master --worker-dir=<volume mount point>/worker --
    (cont.)app-name=GSfastDNAml
```

- Configure the worker part

```
cd <volume mount point>/worker
config_worker.sh interactive --master-dir=<volume mount point
    (cont.)>/master --worker-dir=<volume mount point>/worker --
    (cont.)app-name=GSfastDNAml
```

3.6.16 Test procedure specification

Test procedure specification identifier: wp34-xfs-ts01-gsdna01-tps03
Test log identifier: wp34-xfs-ts01-gsdna01-tl03

Test design specification reference: wp34-xfs-ts01-gsdna01-tds01

Purpose

This is the test procedure to execute wp34-xfs-ts01-gsdna01-tcs03.

Procedure Steps

Log A log of the operations performed by the different XtreemFS server components may be generated during the execution of the tests. The file location is determined by how the XtreemFS server components have been started. A log of the status of the application is generated in the current working directory with a name starting with '.gsmon'.

Set Up To set up the environment, the XtreemFS services must have been installed and started in one or more server hosts and contain an already created volume. This volume must be mounted in all hosts that will participate in the test. Those procedures are explained in detail on the XtreemFS User Manual.

All hosts participating in the test must have an SSH server and client installed. The user account on these hosts must have the same name and UID. Furthermore, the accounts must already have SSH public keys shared in order to access from one to the other without requiring passwords.

All steps described in the test procedures wp34-xfs-ts01-gsdna01-tps01 and wp34-xfs-ts01-gsdna01-tps02 must be performed in order to install the application in the volume.

Proceed

- Enter the master directory

```
cd <volume mount point>/master
```

- Select a sample input file and run the application

```
./GSfastDNAml <sample phy file>
```

3.6.17 Test log

Test log identifier: wp34-xfs-ts01-gsdna01-tl01

Description

This test was performed on an Athlon64 3000+ Linux host (uml-master) running Debian sid with kernel 2.6.22 (linux-image-2.6.22-2-amd64). The host run all the XtreemFS services and several virtual machines. The virtual machines (uml-host1, uml-host2, uml-host3 and uml-host4) were User Mode Linux (UML) instances running with a Debian sid User Mode Linux 2.6.22 (user-mode-linux_2.6.22-1um-1) kernel under a Debian sid distribution. All XtreemFS services were run directly on the host (uml-host) while the virtual machines functioned exclusively as clients.

The tested XtreemFS release is internal_release_1.

Activity and Event Entries

Execution Description The wp34-xfs-ts01-gsdna01-tps01 procedure was started on the UML instance (uml-host1). First the XtreemFS volume was mounted.

```
jm@uml-host1:~/XtreemFS$ /aplic/xtreemfs/current/AL/src/XtreemFS -
    (cont.)o volume_url=http://uml-master:32636/volume001,direct_io
    (cont.)mnt/volume001
```

The volume mounted successfully. Then we entered the newly mounted volume.

```
jm@uml-host1:~/XtreemFS$ cd mnt/volume001
```

The operation was successful. Then the tarball was extracted.

```
jm@uml-host1:~/xtreemfs/mnt/volume001$ tar -xpjf ~/files/fastdna.
    (cont.)tbz
tar: master/.deps: Cannot change mode to rwxr-xr-x: Function not
    (cont.)implemented
tar: master/Results: Cannot change mode to rwxr-xr-x: Function not
    (cont.) implemented
tar: master/testdata/others: Cannot change mode to rwxr-xr-x:
    (cont.)Function not implemented
tar: master/testdata: Cannot change mode to rwxr-xr-x: Function
    (cont.)not implemented
tar: master: Cannot change mode to rwxr-xr-x: Function not
    (cont.)implemented
tar: worker/.deps: Cannot change mode to rwxr-xr-x: Function not
    (cont.)implemented
tar: worker: Cannot change mode to rwxr-xr-x: Function not
    (cont.)implemented
tar: Error exit delayed from previous errors
```

The extraction was successful but generated warnings related to permission changes on directories being unimplemented. Then we extracted the same tarball into a directory outside the XtreemFS volume and compared the contents of both directories. In this case, our reference directory was located on an NFS export.

```
jm@uml-host1:~/xtreemfs/mnt/volume001$ (cd ~/xtreemfs/reference &&
    (cont.) tar -xpjf ~/files/fastdna.tbz)
jm@uml-host1:~/xtreemfs/mnt/volume001$ diff -urN ~/xtreemfs/
    (cont.)reference .
jm@uml-host1:~/xtreemfs/mnt/volume001$ ls -lR > /tmp/ls-pwd
jm@uml-host1:~/xtreemfs/mnt/volume001$ ls -lR ~/xtreemfs/reference
    (cont.) > /tmp/ls-ref
jm@uml-host1:~/xtreemfs/mnt/volume001$ diff -du /tmp/ls-ref /tmp/
    (cont.)ls-pwd
--- /tmp/ls-ref 2007-11-10 20:51:27.421800000 +0100
+++ /tmp/ls-pwd 2007-11-10 20:50:59.938199000 +0100
@@ -1,170 +1,170 @@
-/home/jm/xtreemfs/reference:
-total 8
-drwxr-xr-x 5 jm jm 4096 Sep 14  2006 master
-drwxr-xr-x 3 jm jm 4096 Sep  7  2006 worker
+.:
+total 0
+drwxrwxrwx 1 jm jm 0 Nov 10 20:42 master
+drwxrwxrwx 1 jm jm 0 Nov 10 20:43 worker

[suppressed lines]

-/home/jm/xtreemfs/reference/master/Results:
-total 84
--rw-r--r-- 1 jm jm  3576 Jul 24  2006 align.phy.cpt
```

```
—rw-r—r— 1 jm jm  3271 Jul 24  2006 align.phy.trf
—rw-r—r— 1 jm jm  2326 Jul 24  2006 fastDNAml.50.5.4fair26n51t.
    (cont.)align.err
—rw-r—r— 1 jm jm 66402 Jul 24  2006 fastDNAml.50.5.4fair26n51t.
    (cont.)align.out
+./master/Results:
+total 0
+—rwxrwxrwx 1 jm jm  3576 Nov 10 20:42 align.phy.cpt
+—rwxrwxrwx 1 jm jm  3271 Nov 10 20:42 align.phy.trf
+—rwxrwxrwx 1 jm jm  2326 Nov 10 20:42 fastDNAml.50.5.4fair26n51t.
    (cont.)align.err
+—rwxrwxrwx 1 jm jm 66402 Nov 10 20:42 fastDNAml.50.5.4fair26n51t.
    (cont.)align.out

[suppressed lines]
```

The file contents comparison was successful, but the directory contents comparison presented important differences.

Procedure Results The tar command exited with errors due to permission changes on directories being unimplemented. Although that operation on a directory generated an error, the diff performed on the directories shows that the functionality is neither implemented for directories nor files, yet it does not produce an error when run on files.

The directory diff also shows that directory sizes, hard link counts and modification times are not fully implemented.

Incident Report Identifiers

Anomalous results are reported in wp34-xfs-ts01-gsdna01-tir01.

3.6.18 Test log

Test log identifier: wp34-xfs-ts01-gsdna01-tl02

Description

This test was performed on an Athlon64 3000+ Linux host (uml-master) running Debian sid with kernel 2.6.22 (linux-image-2.6.22-2-amd64). The host run all the XtreemFS services and several virtual machines. The virtual machines (uml-host1, uml-host2, uml-host3 and uml-host4) were User Mode Linux (UML) instances running with a Debian sid User Mode Linux 2.6.22 (user-mode-linux_2.6.22-1um-1) kernel under a Debian sid distribution. All XtreemFS services were run directly on the host (uml-host) while the virtual machines functioned exclusively as clients.

The tested XtreemFS release is internal_release_1.

Activity and Event Entries

Execution Description The wp34-xfs-ts01-gsdna01-tps02 procedure was started on the uml instance (uml-host1). First the XtreemFS volume was mounted.

```
jm@uml-host1:~/xtreemfs$ /aplic/xtreemfs/current/AL/src/xtreemfs -
    (cont.)o volume_url=http://uml-master:32636/volume001,direct_io
    (cont.)mnt/volume001
```

The volume mounted successfully. Then we entered the master code directory.

```
jm@uml-host1:~/xtreemfs$ cd mnt/volume001/master
```

The operation was successful. Then attempted to run the 'autogen.sh' script.

```
jm@uml-host1:~/xtreemfs/mnt/volume001/master$ ./autogen.sh
-bash: ./autogen.sh: /bin/sh: bad interpreter: Bad address
```

The system failed to execute the script directly, so we tried to execute the script passing it as an argument to a shell interpreter.

```
jm@uml-host1:~/xtreemfs/mnt/volume001/master$ /bin/sh ./autogen.sh
chmod: changing permissions of 'conf12304.sh': Function not
    (cont.)implemented
chmod: changing permissions of 'conf12525.sh': Function not
    (cont.)implemented
./autogen.sh: ./configure: /bin/sh: bad interpreter: Bad address
```

The script generated several warnings related to changing permissions being unimplemented and finally failed when trying to execute the 'configure' script directly. Again we proceeded to execute the configure script as a parameter to the shell.

```
jm@uml-host1:~/xtreemfs/mnt/volume001/master$ cat autogen.sh
#!/bin/sh
set -e
export CC='cc -m64';
export CXX='g++ -m64';
/usr/bin/aclocal
/usr/bin/automake -a -c
/usr/bin/autoconf
./configure --with-gs-prefix=/gpfs/apps/GRID-S/SSHGS64

for script in workerGS.sh workerGS_script.sh config_worker.sh
    (cont.)config_master.sh ssh_execute.sh; do
        if ( test -e $script && ! test -x $script ); then
            /bin/chmod +x $script;
        fi
done
jm@uml-host1:~/xtreemfs/mnt/volume001/master$ sh ./configure --
    (cont.)with-gs-prefix=/gpfs/apps/GRID-S/SSHGS64
chmod: changing permissions of 'conf12615.sh': Function not
    (cont.)implemented
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
```

```
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking if we should activate AIX workarounds... no
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... configure: error: cannot
   (cont.)run C compiled programs.
If you meant to cross compile, use `--host'.
See `config.log' for more details.
```

The 'configure' script failed when trying to run a binary located in an XtreemFS volume. The relevant part from the 'config.log' file follows:

```
configure:2799: checking whether the C compiler works
configure:2809: ./a.out
./configure: line 2810: ./a.out: Bad address
configure:2812: $? = 126
configure:2821: error: cannot run C compiled programs.
```

We investigated the failure further and tracked it down to a failure in mmapping the binary into memory. Then we contacted the XtreemFS developers regarding this problem and discovered that the 'mmap' system call on XtreemFS volumes is not implemented yet.

In order to have a propper environment for the following test cases, we proceeded to compile the binaries in the main host in a normal file system (uml-master) and created soft links from the binaries and shell scripts in the normal file system to the XtreemFS volume. This worked flawlessly.

Procedure Results The procedure resulted in several warnings and errors. All warnings were related to missing file protection handling. All errors were due to unimplemented functionality required by the 'mmap' system call over files in XtreemFS.

Incident Report Identifiers

Anomalous results are reported in wp34-xfs-ts01-gsdna01-tir02.

3.6.19 Test log

Test log identifier: wp34-xfs-ts01-gsdna01-tl03

Description

This test was performed on an Athlon64 3000+ Linux host (uml-master) running Debian sid with kernel 2.6.22 (linux-image-2.6.22-2-amd64). The host run all the

XtreemFS services and 4 virtual machines. The virtual machines (uml-host1, uml-host2, uml-host3 and uml-host4) were User Mode Linux (UML) instances running with a Debian sid User Mode Linux 2.6.22 (user-mode-linux_2.6.22-1um-1) kernel under a Debian sid distribution. All XtreemFS services were run directly on the host (uml-host) while the virtual machines functioned exclusively as clients.

The tested XtreemFS release is internal_release_1.

Activity and Event Entries

Execution Description The wp34-xfs-ts01-gsdna01-tps03 procedure was initiated on a UML instance (uml-host1) and executed through all UML instances. First the XtreemFS volume was mounted.

```
jm@uml-host1:~/xtreemfs$ /aplic/xtreemfs/current/AL/src/xtreemfs -
    (cont.)o volume_url=http://uml-master:32636/volume001,direct_io
    (cont.)mnt/volume001
```

The volume mounted successfully. Then we entered the master directory.

```
jm@uml-host1:~/xtreemfs$ cd mnt/volume001/master
```

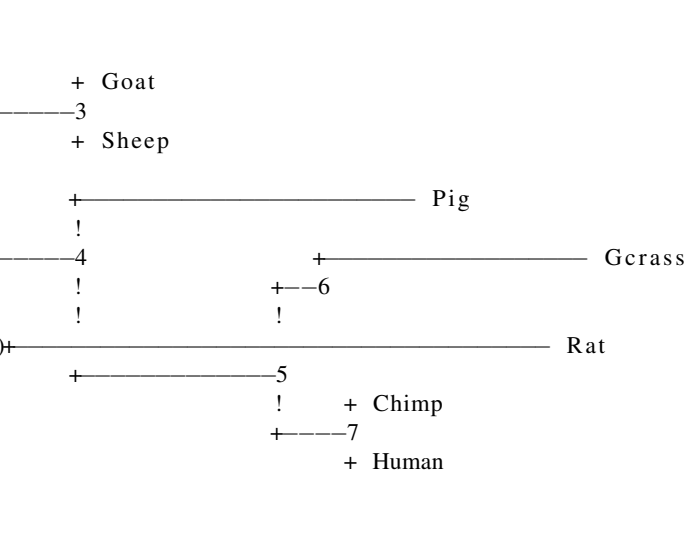
The operation was successful. Then we selected the 'test9_484.phy' input file and launched the application.

The application automatically launches processes on all configured hosts (all the UML instances) through ssh sessions. Those processes contribute to the global calculation by generating and producing data files. Files can be generated by one host and consumed by another host.

```
jm@uml-host1:~/xtreemfs/mnt/volume001/src/fastDNAml_GRIDss/master$
    (cont.) ./GSfastDNAml test9_484.phy
fallback to FIFO policy
PIPE FILE : /tmp/.sshgs-lYRvb3

fastDNAml, version 1.2.2, January 3, 2000,
Copyright (C) 1998, 1999, 2000 by Gary J. Olsen

Based in part on Joseph Felsenstein's

   Nucleic acid sequence Maximum Likelihood method, version 3.3


9 Species, 114 Sites

Quick add (only local branches initially optimized) in effect

Rearrangements of partial trees may cross 1 branch.
Rearrangements of full tree may cross 1 branch.

Total weight of positions in analysis = 114
There are 30 distinct data patterns (columns)
```

```
Empirical Base Frequencies:

   A       0.17446
   C       0.22027
   G       0.37719
  T(U)     0.22807

Transition/transversion ratio =   2.000000

(Transition/transversion parameter =   1.636628)

Adding species:
   Bos
   Rat
   Human
   Sheep
NTIPS 4
      Tested 3 alternative trees
      Ln Likelihood =    -276.62419
   Goat
NTIPS 5
      Tested 5 alternative trees
      Ln Likelihood =    -276.62432
      Doing local rearrangements
      Tested 4 alternative trees
   Cow
NTIPS 6
Send 2 files for evaluation in test insert context
      Tested 7 alternative trees
      Ln Likelihood =    -276.62445
      Doing local rearrangements
Send 2 files for evaluation in the optimization context
      Tested 6 alternative trees
   Pig
NTIPS 7
Send 2 files for evaluation in test insert context
      Tested 9 alternative trees
      Ln Likelihood =    -321.53680
      Doing local rearrangements
Send 2 files for evaluation in the optimization context
      Tested 8 alternative trees
   Gcrass
NTIPS 8
Send 2 files for evaluation in test insert context
      Tested 11 alternative trees
      Ln Likelihood =    -364.80728
      Doing local rearrangements
Send 2 files for evaluation in the optimization context
      Tested 10 alternative trees
   Chimp
NTIPS 9
Send 2 files for evaluation in test insert context
      Tested 13 alternative trees
      Ln Likelihood =    -364.80739
```

```
      Doing local rearrangements
Send 2 files for evaluation in the optimization context
      Tested 12 alternative trees

Examined 89 trees
```




```
Remember: this is an unrooted tree!

Ln Likelihood =     -364.80772
```

Between	And	Length	Approx. Confidence Limits	
1	Cow	0.00000	(zero,	0.01692)
1	2	0.00000	(zero,	0.02452)
2	3	0.02698	(zero,	0.05861) **
3	Goat	0.00000	(zero,	0.01692)
3	Sheep	0.00000	(zero,	0.01692)
2	4	0.02591	(zero,	0.06003) **
4	Pig	0.08432	(0.02809,	0.14679) **
4	5	0.04989	(0.00545,	0.09815) **
5	6	0.01425	(zero,	0.04149)
6	Gcrass	0.06083	(0.01223,	0.11402) **

```
6           Rat                 0.12062     (   0.05237,
   (cont.)0.19829) **
5             7                 0.02118     (      zero,
   (cont.)0.05186) **
7           Chimp               0.00000     (      zero,
   (cont.)0.01696)
7           Human               0.00000     (      zero,
   (cont.)0.01696)
1           Bos                 0.00000     (      zero,
   (cont.)0.01692)

  * = significantly positive, P < 0.05
  ** = significantly positive, P < 0.01


Tree also written to /home/jm/xtreemfs/mnt/volume001/src/
   (cont.)fastDNAml_GRIDss/master/test9_484.phy.trf

callback daemon pipe is closing ...
Total Execution Time 150.691564
```

The execution finished successfully and its results were written to a text file.

```
jm@uml-host1:~/xtreemfs$ diff -u ~/xtreemfs/reference/master/
   (cont.)test9_484.phy.trf test9_484.phy.trf
```

We compared the results to those produced by the sequential version of the application and they matched perfectly.

Procedure Results We executed the test six times. Of those, four finished successfully, one led to the XtreemFS client being hung in one of the hosts and one resulted in the client process dying in the host where the application had been launched.

Incident Report Identifiers

Anomalous results are reported in wp34-xfs-ts01-gsdna01-tir03.

3.6.20 Test incident report – Protection mechanisms, hard link counts and modification times

Test incident report identifier: wp34-xfs-ts01-gsdna01-tir01

Summary

Standard UNIX protection mechanisms, hard link counts and modification times are unimplemented or misbehaving in the internal_release_1 of XtreemFS.

Incident Description

The incident can be reproduced by extracting a tarball inside an XtreemFS volume. All created files and directories are created with 'rwx' permission for all users and their modification time is set to its creation time, regardless of whether tar has been instructed to keep the original time. Furthermore, trying to change file permissions with the standard 'chmod' command fails with a 'Function not implemented' error and touching files has no effect.

Impact

Confidentiality is compromised and elements in the filesystem may end up being removed or overwritten by accident.

3.6.21 Test incident report – Protection mechanisms and memory mapping

Test incident report identifier: wp34-xfs-ts01-gsdna01-tir02

Summary

Standard UNIX protection mechanisms are unimplemented or misbehaving in internal_release_1 of XtreemFS. The functionality required by the 'mmap' system call over files from XtreemFS is unimplemented at this time and is required to execute binaries residing in the file system.

Incident Description

The warnings can be reproduced by trying to change the protections of a file with the standard 'chmod' command from UNIX. The errors can be reproduced by trying to execute shell scripts and normal binaries residing in an XtreemFS volume.

Impact

The missing file protection handling could lead to confidentiality being compromised and elements in the filesystem ending up removed or overwritten by accident.

The missing 'mmap' functionality prevents using XtreemFS for many situations, including software development.

3.6.22 Test incident report – Reliability issues during concurrent accesses from several hosts

Test incident report identifier: wp34-xfs-ts01-gsdna01-tir03

Summary

The XtreemFS client in internal_release_1 has some reliability issues that cause it to hang or die in situations where a directory has some file operations being executed from several hosts at the same time.

Incident Description

We executed the GSfastDNAml application six times in a configuration with four client hosts and one server host. Four executions finished successfully. One execution hanged up due to one XtreemFS client hanging up and taking up all CPU time. Another execution finished abruptly due to the client running the main application having the XtreemFS client die and thus leaving the main application without access to the files.

Impact

Given the early stage of development, reliability issues are to be expected until all basic functionality has been put in place.

3.6.23 Test design specification

Test design specification identifier: wp34-xfs-ts01-wiss01-tds01
Test plan reference: wp34-xfs-ts01-wiss01-tds01

Features to be Tested

The following file system features and requirements will be tested:

1. Creating files: R50
2. Creating directories: R50, R76
3. Accessing files and directories: R50
4. Transferring files from an ext3 volume to the XtreemFS volume: R50
5. Reading files with Wissenheim, checking for consistency: R50

Test Identification

This test design specification is covered by test case wp34-xfs-ts01-wiss01-tcs01.

Feature Pass/Fail Criteria

The test succeeds if the following criteria are fulfilled:

- file and directory creation successful
- all files and directories are accessible
- copy operation performed without errors
- Wissenheim was able to read all files without reporting corrupted data

3.6.24 Test design specification

Test design specification identifier: wp34-xfs-ts01-wiss01-tds02
Test plan reference: wp34-xfs-ts01-wiss01-tds02

Features to be Tested

The following file system features and requirements will be tested:

1. GOM initialization: R83
2. creation of memory mapped region: R83
3. data exchange via memory mapping: R83
4. strict consistency: R83
5. transactional consistency: R83

Approach Refinements

The Wissenheim application tested the GOM layer of XtreemFS with the dynamic library libOSS.so provided by WP3.4. The library allows the application to establish a shared memory region which is shared among different machines. Different memory consistency models are available to the programmer. Due to the early release many features of the library which should work without altering the application had to be called directly via library functions. To test these features the Wissenheim application was altered to perform the tests. Because the whole functionality is not available special test cases had been developed and implemented to check e.g. the correctness off the memory consistency.

Test Identification

This test design specification is covered by test case wp34-xfs-ts01-wiss01-tcs02.

Feature Pass/Fail Criteria

For the test to be successful, all the tested features must be functional. Due to the early stage of the tested software, the test for the transactional consistency could not be performed because the feature is still missing. The test is passed although transaction consistency is not implemented yet.

3.6.25 Test case specification – standard file operations

Test case specification identifier: wp34-xfs-ts01-wiss01-tcs01

Test Items

This test will check whether:

- a copy operation from a standard linux ext3 file system will be performed correctly
- read operations by the Wissenheim application can be done
- files are in a consistent state

Input Specifications

An empty XtreemFS volume mounted within Linux and also a mounted Ext3 source partition. Wissenheim application and all necessary libraries installed.

Output Specifications

All files have been correctly transfered. Access to all files is possible. Read operations by the Wissenheim application have been completed correctly and no consistency problems have been reported.

Environmental Needs

Hardware Standard IBM-PC compatible computers.

Software Standard Linux with a bash shell and a mounted ext3 volume. For Wissenheim, an X-Server with DirectRendering and OpenGL is needed.

3.6.26 Test case specification – OSS tests

Test case specification identifier: wp34-xfs-ts01-wiss01-tcs02

Test Items

This test will evaluate the setup of the OSS (Object Sharing Service) features included with XtreemFS GOM.

Input Specifications

Starting the basic object sharing service on multiple machines, interconnecting those machines and data exchange via memory mapping. Wissenheim test cases for checking the strict consistency are started.

Output Specifications

The machines are able to interconnect with each other and the data exchange via memory mapping can be performed. The consistency tests have been successful.

Environmental Needs

Hardware Standard IBM-PC compatible computers. Fast Ethernet switched network interconnect.

Software Standard Linux with a bash shell. The libglib version 1.2 and lreadline are needed for the oss library.

3.6.27 Test procedure specification

Test procedure specification identifier: wp34-xfs-ts01-wiss01-tps01
Test log identifier: wp34-xfs-ts01-wiss01-tl01

Test design specification reference: wp34-xfs-ts01-wiss01-tds01

Purpose

This test should verify the correctness of the basic I/O operations of XtreemFS.

Procedure Steps

Set Up An already created and mounted XtreemFS volume with one or more OSDs must be installed.

Start Invoking a recursive copying of files and directories with cp from an ext3 source volume to the XtreemFS volume. Start of Wissenheim application with data directory set to the files on the XtreemFS volume.

Measure Integrity of the files and directory structure. The file integrity will be automatically tested when these files are loaded by Wissenheim.

3.6.28 Test procedure specification

Test procedure specification identifier: wp34-xfs-ts01-wiss01-tps02
Test log identifier: wp34-xfs-ts01-wiss01-tl02

Test design specification reference: wp34-xfs-ts01-wiss01-tds02

Purpose

This test should verify the basic functionality of the OSS feature of XtreemFS.

Procedure Steps

Set Up XtreemFS OSS is installed and ready to used. All necessary programs have been distributed to the test machines.

Start Starting the Wissenheim application in OSS test mode.

Measure All test application are writing into separated memory locations which are read by all the others. If the basic mechanisms are working correctly all machines should see the the data of all the others.

3.6.29 Test log

Test log identifier: wp34-xfs-ts01-wiss01-tl01

Description

The test as described in test procedure wp34-xfs-ts01-wiss01-tps01 was performed on two AMD64 3800+ machines with 1GB RAM running Ubuntu 7.04 hosting the OSD on one machine and the MRC and directory service on the other. The client machine invoking the copy operation and performing as host was a Pentium D 2.66GHz with 1GB RAM. All computers were connected by a switched fast Ethernet network. The computers were running no other programs but Gnome desktop environment and the standard system tools. For testing the internal_release_1 of XtreemFS was used with default parameters.

Activity and Event Entries

All tests have been performed by Michael Sonnenfroh (UDUS).

Execution Description Copy command was executed with -r for recursive copying. After copy operation has been finished, the Wissenheim application is started with data directory set to the file on the XtreemFS volume.

Procedure Results All files where copied correctly, directory structure was intact. Wissenheim started without any problems loading the files and confirming the consistent state of the data.

```
dlopen, dlsym, esp: B7F419C0, B7F40D10Liboss successfully
   (cont.)initialized
X11:: creating window
X11: display opened successfully
size 40GLX::version:1.3
GLX::checking for necessary GL extensions:
GLX::GL_ARB_vertex_buffer_object found
World 'TropicalIsland' created and registered.
Tropical Island loading:
 Sphere
MATERIAL
Heaven.jpg
UV
MATRIX
MESHLINK Sphere
REGISTERED::/root/wissenheim/data/tropicalIsland/meshes/Heaven
WMT......[OK]
RockMesh.001
MATERIAL
SandsteinSmall.jpg
UV
BeachMesh
MATERIAL
Sand.jpg
UV
OceanMesh
MATERIAL
Water.jpg
UV
RockMesh
MATERIAL
SandsteinSmall.jpg
UV
GeoSphere01.001
MATERIAL
SandsteinSmall.jpg
UV
MATRIX
MESHLINK RockMesh.001
MATRIX
MESHLINK BeachMesh
MATRIX
MESHLINK OceanMesh
MATRIX
MESHLINK RockMesh
MATRIX
MESHLINK GeoSphere01.001
REGISTERED::/root/wissenheim/data/tropicalIsland/meshes/Rocks
REGISTERED::/root/wissenheim/data/tropicalIsland/meshes/Beach
REGISTERED::/root/wissenheim/data/tropicalIsland/meshes/Ocean
```

```
REGISTERED::/root/wissenheim/data/tropicalIsland/meshes/Rocks_Walk
REGISTERED::/root/wissenheim/data/tropicalIsland/meshes/BeachStone
WMT......[OK]

.
.
.
.

Cylinder01
MATERIAL
TorchWood.jpg
UV
Box03
MATERIAL
TorchFlame.bmp
UV
MATRIX
MESHLINK Cylinder01
MATRIX
MESHLINK Box03
REGISTERED::/root/wissenheim/data/tropicalIsland/meshes/Torch
WMT......[OK]
body_01 - Standard
MATERIAL
UV
MATRIX
MESHLINK body_01 - Standard
REGISTERED::/root/wissenheim/data/avatars/meshes/Philosoph
WMT......[OK]
Import successful
```

Incident Report Identifiers

Anomalous results are reported in wp34-xfs-ts01-wiss01-tir01.

3.6.30 Test log

Test log identifier: wp34-xfs-ts01-wiss01-tl02

Description

The test as described in test procedure wp34-xfs-ts01-wiss01-tps02 was performed on two AMD64 3800+ machines with 1GB RAM running Ubuntu 7.04. All computers were connected by a switched fast Ethernet network. The computers were running no other programs but Gnome desktop environment and the standard system tools. For testing the first version internal available release of the libOSS.so library was used. On every machine one instance of Wissenheim was started running the special test cases.

Activity and Event Entries

The test where performed by Michael Sonnenfroh.

Execution Description The Wissenheim application is started on all nodes within 10s delay between the start of the first application and the start of the last application.

Procedure Results Initialization of the OSS sub system performed without problems. The interconnection of the nodes succeeded as well. The memory mapped region could be instantiated on all machines and data exchange could be performed. Test implementation of a simple barrier passed the test. The test of the strict consistency by testing for lost-updates of a distributed incrementation of shared memory locations revealed no problems.

Incident Report Identifiers

Anomalous results are reported in wp34-xfs-ts01-wiss01-tir02.

3.6.31 Test incident report – POSIX rights errors

Test incident report identifier: wp34-xfs-ts01-wiss01-tir01

Summary

Problems with POSIX rights found.

Incident Description

After the files have been copied, all user rights had been set to "xrw" for all groups instead of set according to the source directory.

Impact

The incident will have no side effects for the tests because Wissenheim is only reading files. But it is a serious problem for applications which rely on file access rights.

3.6.32 Test incident report – data inconsistencies

Test incident report identifier: wp34-xfs-ts01-wiss01-tir02

Summary

Stability problems have been found if a special constellation of clients/nodes is used.

Incident Description

If an active client node is acting as the home node there were sometimes memory inconsistency, especially the lost-update test revealed several missed updates. One rare occasions a dead lock occurred which resulted in a stall of all Wissenheim instances on all machines. If a dedicated home node was present the previous described errors did not occur. If the start time of the Wissenheim applications on the different nodes differed to much the nodes were unable to interconnect with each other.

Impact

The incident will have no side effects for the tests because it only occurs with a special constellation of node /clients which can be easily avoided. Nevertheless, in the final version of OSS these problems have to be fixed.

3.6.33 Test summary report – XtreemFS

Test summary report identifier: wp34-xfs-ts01-tsr

Summary

The internal release 1 of XtreemFS (svn branch `internal_release_1`) has been evaluated using the following applications:

- GSDNA in the test cases wp34-xfs-ts01-gsdna01-tcs01, wp34-xfs-ts01-gsdna01-tcs02, and wp34-xfs-ts01-gsdna01-tcs03
- MaxDB in the test case wp34-xfs-ts01-maxdb01-tcs01
- Bonnie benchmark in the test case wp34-xfs-ts01-maxdb01-tcs02
- WISS in the test case wp34-xfs-ts01-wiss01-tcs01 and wp34-xfs-ts01-wiss01-tcs02

The GSDNA tests focused both on rather standard procedures like directory listings, unpacking archives, compiling and also on running an application test (GSDNA). MaxDB was a pure application test that executed an industrial-strength DBMS that stored its backend data file on an XtreemFS volume. The Bonnie benchmark was a synthetic file performance test that stressed XtreemFS with sequences of long sequential read/write accesses and random accesses as well. The WISS test sequence evaluated XtreemFS copying WISS input files from a local file system to an XtreemFS volume and then running a WISS application instance with this input data on the XtreemFS volume. Additionally, WISS application tested the Object Sharing Service (implementation of GOM) of XtreemFS that enables data exchange via memory regions that are shared between clients.

Variances

During the tests of XtreemFS the flow of experiments had to be changed for the following actions:

- Running executables located on XtreemFS is not possible since Linux uses the `mmap` system call for loading them into RAM, and `mmap` is not yet implemented by XtreemFS. Therefore, GNU autoconf and GNU automake could not work since they generate and run binary executable files in the current (XtreemFS) directory. Additionally, the executable binaries belonging to GSDNA application had to be moved to a local files system for running the XtreemFS tests (wp34-xfs-ts01-gsdna01-tcs02).

- GNU autoconf could not be used directly on a source packages residing on XtreemFS because it uses the `chmod` operation (change ownership of a file) which is not yet implemented by XtreemFS (wp34-xfs-ts01-gsdna01-tcs02)

Comprehensiveness Assessment

The testing process was in line with the approach given in wp34-xfs-ts01-tp.

Summary of Results

During the tests there were a number of incidents registered. Some of them were not crucial for a successful execution of the tested application (wp34-xfs-ts01-wiss01-tir01, wp34-xfs-ts01-wiss01-tir02, wp34-xfs-ts01-gsdna01-tir01). Incident wp34-xfs-ts01-gsdna01-tir02 could be partly resolved with workarounds, e.g. that fall back on a local file system. Incidents wp34-xfs-ts01-gsdna01-tir03 and wp34-xfs-ts01-maxdb01-tir01 could not be resolved by the testers. All incidents have been reported to XtreemFS developers using email or/and the bugtracker at `http://gforge.inria.fr`.

Evaluation

During the evaluation of XtreemFS (internal release 1) by the mentioned above applications one arrives at the following conclusions:

- The most important POSIX functionality is implemented, which already enables execution of many applications.

- For simple/standard configurations XtreemFS was rather stable, no data corruption has been detected during the tests.

- The compilation and installation process of XtreemFS is relatively simple, no special knowledge or skills are required.

- Some of the tested applications were able to run on the tested XtreemFS version only with some workarounds that compensate only partly implemented POSIX standards
- Stability and reliability characteristics have to be improved in the next releases of XtreemFS. Handling non-standard/concurrent configurations has to be implemented and tested more carefully.
- XtreemFS (internal release 1) could not profit from more hardware resources (scalability issues).
- Absolute I/O performance (e.g. for file scanning) was unsatisfactory and far below the raw hardware (or native OS) capabilities.

Below we present a list of XtreemOS-related requirements and their fulfillment status. We also list requirements that are not implemented yet, and therefore had not been tested.

- R74 not fulfilled, since XtreemFS hangs sometimes if file operations are executed from several hosts (wp34-xfs-ts01-gsdna01-tir03)
- R75 not tested, since VO and XtreemFS integration is not completed, UNIX file permissions are not fully implemented
- R76 fulfilled
- R77 not tested, since access right management is not fully implemented by XtreemFS
- R78 not tested, since not implemented
- R79 could not be tested, because of errors occurring when handling large files (wp34-xfs-ts01-maxdb01-tir01)
- R80 not tested, since not implemented
- R81 not tested, since not implemented
- R84 not tested, since not implemented
- R83 partly fulfilled, only the basic functionality could be tested, transactions are scheduled for later testing (wp34-xfs-ts01-wiss01-tds02)
- R50 partly fulfilled
- R1 not fulfilled, scalability and performance of XtreemFS data management were rather poor
- R8 not tested, because of lack of hardware resources. It is planned to test this requirement later on the Grid5000 infrastructure with more stable and scalable XtreemFS releases

- R85 not fulfilled and not fully tested, since ACLs are not yet supported in XtreemFS

3.7 Evaluation of CDA

3.7.1 Test plan – CDA

Test plan identifier: wp35-cda-ts01-tp

Introduction

This test plan covers the Credential Distribution Agency (CDA). The functionalities of CDA include:

- correct generation of the XOS-Certificate,
- retrieving XOS-Certificate from a remote server,
- use the XOS-Certificate for Identity purposes.

More information about the CDA purpose and use-cases can be found in the XtreemOS deliverables of the WP 3.5 [5, 6].

Test Items

The current version of the software (as of the time of writing this test plan) includes the following parts and functionalities:

- A server accessible by TCP/IP that authenticates the user,
- a client script that authenticates the user,
- generation of an XOS-certificate protected by a passphrase.

The contact person for getting the software and documentation is Ian Johnson (`I.J.Johnson@rl.ac.uk`) who supported us during the testing process providing:

- source code and binaries,
- fake database of XtreemOS users,
- help about how to use the CDA development.

Features to be Tested

The following features will be tested:

- correct authentication of the users against the server
- correct generation of the XOS-certificate including:
  - usability and compatibility of the certificate with X509 standards
  - encryption of data using the certificate
  - signing of data using the certificate

Approach

The purpose of these tests is to evaluate the current version of the software, provide feedback to the developers, and to check in how far WP4.2 requirements are fulfilled.

During the test, we will focus on evaluating if the development works well (we can authenticate users and get their certificate, we can use the certificate, etc), but not the integration with the other XtreemOS components. The interplay with other components can be tested when an integrated version of XtreemOS will become available.

Tests were going to be done using Java and C applications. We are only presenting in this document the Java evaluation because, even when some work was done with the C applications, the results were not different. The C test that we carried out were only a wrapper of the Java functionality.

Environmental Needs

The required resources include two computers, one for the server and another for the client. Server and client can also be executed on only one computer running both. For the Java tests, the machines must run Linux, with the Java platform installed.

Responsibilities

The tests are responsibility of Javier Noguera (T6).

Schedule

Being that the software is in its early development stage, the installation, setup and familiarization are complex processes, requiring one or two weeks if the staff is not familiar with the procedure. However, after the preparation is finished, the tests are simple and will be done in a few days. All the planned tests should be finished in time for XtreemOS deliverable D4.2.4.

Risks and Contingencies

Due to the early stage of the development, tests have been realized using the software and configuration files provided by the partners involved in the development of the libraries.

Next releases should be more autonomous and allow best integration between the code and the libraries. We do not envision any specific risk.

3.7.2 Test design specification

Test design specification identifier: wp35-cda-ts01-tds01
Test plan reference: wp35-cda-ts01-tp

Features to be Tested

The following features are the subject of this test design specification:

1. correct authentication of the users against the server (test case specification wp35-cda-ts01-tcs01), related to requirements R89 and R91,

2. correct generation of the XOS-certificate (test case specification wp35-cda-ts01-tcs02), related to requirements R86 and R88,

Approach Refinements

CDA developers have provided us with main classes that can be used by command line for simulating the server process and the client remote calls. During the test:

1. we have used those classes in order to start the server

2. we have use those classes in order to request for authentication from the server

3. the execution of the authentication process has provided us with a public/private key and a certificate (XOS-Certificate)

4. we have used standard tools such as OpenSSL to manage certificates, sign, cipher and verify the identity of the users.

Test Identification

The test case specifications are wp35-cda-ts01-tcs01 and wp35-cda-ts01-tcs02.

3.7.3 Test case specification – Authenticating users

Test case specification identifier: wp35-cda-ts01-tcs01

Test Items

This test case evaluates the correct authentication of the users against the server.

Input Specifications

The server script launches the CDA server that receives as parameters the desired port, the public certificate of the CDA service and the CDA server private key.

Note that public CDA certificates and CDA private key for accessing those certificates are already created and were delivered to us by the CDA developers.

We use a client standalone program developed in Java that authenticates the user. This program must be launched specifying the server IP and port plus the desired file names where the private key and certificate for this user are going to be stored.

Once connected to the server the client will ask the user to enter the VO name and his username and password. Finally the client will ask for a pass-phrase to protect the new private key.

Output Specifications

If the authentication succeeds (the user introduces a correct username and password), the server creates an XOS-Certificate containing the VO attributes for that user. The certificate is finally stored in the desired location.

Environmental Needs

Hardware A single x86 machine with Linux is required.

Software The following software was used:

- Java Virtual Machine 1.5 or higher
- CDA Java libraries

3.7.4 Test procedure specification

Test procedure specification identifier: wp35-cda-ts01-tps01
Test design specification reference: wp35-cda-ts01-tds01
Test case specification reference: wp35-cda-ts01-tcs01

Purpose

This procedure executes the test case wp35-cda-ts01-tcs01.

Procedure Steps

Set Up In order to execute the tests we only need a Linux machine with CDA installed. We will run two different Java programs: the server and the client.

Start the server For starting the server, we execute the command `./runserver`, which is a wrapper of the following instructions:

```
\#!/bin/sh
java -server -Djavax.net.ssl.keyStore=server.jks
    -Djavax.net.ssl.keyStorePassword=serverPassword
    -cp CDA.jar org.xtreemos.wp35.cda.CdaServer cdakey.pem cdacert
       (cont.).pem 5555
```

Notice that 5555 is the desired port where the server will attend requests, server.jks is the java keystore where the information of the users is stored and cdakey.pem and cdacert.pem are the server key and certificate.

Start the client For starting the client, we simply execute the command:
`./runclient localhost 5555 userkey.pem usercert.pem`
As a result, the user needs to answer some questions:

```
Enter name of VO: esvo
Enter your username in VO esvo: bob
Enter password for username bob in VO esvo:
Passphrase to protect private key (at least 8 characters long):
Type pass-phrase again to confirm:
```

The user authenticates himself against the server, using his VO and his password. As a result, a private key and a certificate are returned.

Contingencies The only remarkable thing to say here is the importance of having an API for accessing the library functions. The implementation of this API does not seem too difficult.

3.7.5 Test log

Test log identifier: wp35-cda-ts01-tl01

Description

Tests where performed by Javier Noguera (T6). No special hardware was needed.
For the test we used:

1. computer: a laptop computer with dual core processor and 1Gb of memory
2. OS: Linux Ubuntu (desktop distribution)
3. other: Sun Microsystem's Java SDK

Execution Description Refer to wp35-cda-ts01-tps01 for a more detailed description.

For testing the client/server application, we have been given three users: Bob, Alice and Eve. Following the instructions, we successfully generated some private keys and their associated certificates.

Procedure Results The test succeeded. The user is authenticated by the server and the certificate is stored in the right location.

```
Enter name of VO: esvo
Enter your username in VO esvo: bob
Enter password for username bob in VO esvo:
Passphrase to protect private key (at least 8 characters long):
Type pass-phrase again to confirm:
```

After the execution, the bob certificate looks like this:

```
Owner: CN=/esvo/bob, OU=CDA, O=XtreemOS Project
Issuer: OU=CDA, O=XtreemOS Project
Serial number: 11614ae41dd
Valid from: Tue Nov 06 12:12:52 CET 2007 until: Thu Dec 06
   (cont.)12:22:52 CET 2007
Certificate fingerprints:
         MD5:  3C:A4:16:A0:BC:4F:79:53:7E:D5:1F:D4:1F:02:83:F2
         SHA1: 0B:78:51:DF:EC:A2:CB:89:AD:2D:4F:F9:5A:EB:24:CA:BC
            (cont.):3E:8D:1D
          Signature algorithm name: SHA1withRSA
          Version: 3

Extensions:

#1: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  DigitalSignature
  Key_Encipherment
]

#2: ObjectId: 2.5.29.37 Criticality=true
ExtendedKeyUsages [
  clientAuth
]

#3: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:false
  PathLen: undefined
]

#4: ObjectId: 1.34.5.0.14.8 Criticality=false

#5: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
```

```
0000: F9 82 74 01 CB 54 C8 B8    92 D4 EE 32 9E B8 C4 C1   ..t..T
   (cont.).....2....
0010: 4E 1A 97 3E                                          N..>
]
]

#6: ObjectId: 1.34.5.0.14.5 Criticality=false

#7: ObjectId: 1.34.5.0.14.4 Criticality=false

#8: ObjectId: 1.34.5.0.14.7 Criticality=false

#9: ObjectId: 1.34.5.0.14.6 Criticality=false

#10: ObjectId: 1.34.5.0.14.1 Criticality=false

#11: ObjectId: 1.34.5.0.14.2 Criticality=false

#12: ObjectId: 1.34.5.0.14.3 Criticality=false

#13: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 8E 12 B9 EC 7E 56 99 54    9C 16 38 B8 48 4B 83 D8   .....V.T
   (cont.)..8.HK..
0010: 44 3B B8 1D                                          D;..
]

[OU=CDA, O=XtreemOS Project]
SerialNumber: [     942306d5 f98640dd]
]
```

Anomalous Events

There were no anomalous events during the execution.

Incident Report Identifiers

There were no incidents during the execution of the test.

3.7.6 Test case specification – Correctness of generated XOS-Certificates

Test case specification identifier: wp35-cda-ts01-tcs02

Test Items

This test case tests the correct generation of the XOS-Certificate.

Input Specifications

After the authentication of the user, the client generates a XOS-Certificate compliant with the X509 v3 specification. The certificate is stored protected with a pass-phrase specified by the user at login time.

Output Specifications

As a result, a valid X509 certificate is saved in the filesystem. The certificate is accessible and can be used with usual tools such as OpenSSL or any Java development tool.

Hardware A single x86 machine with Linux is required.

Software The following software was used:

- Java Virtual Machine 1.5 or higher
- CDA Java libraries

3.7.7 Test procedure specification

Test procedure specification identifier: wp35-cda-ts01-tps02
Test design specification reference: wp35-cda-ts01-tds01
Test case specification reference: wp35-cda-ts01-tcs02

Purpose

This procedure executes the test case wp35-cda-ts01-tcs02.

Procedure Steps

Client CDA application generates PEM certificates and keys. PEM is the draft Internet Privacy-Enhanced Mail standard, designed, proposed, but not yet officially adopted, by the Internet Activities Board to provide secure electronic mail over the Internet. PEM includes encryption, authentication, and key management, and allows for using both, public-key and secret-key crypto systems.

Multiple cryptographic tools are supported; for each mail message, the specific encryption algorithm, digital signature algorithm, hash function, and so on are specified in the header. PEM explicitly supports only a few cryptographic algorithms; others may be added later.

Test private key Since the application does not provide any other functionality we have to cipher and decipher data with the private keys generated by the client application, using with "OpenSSL" command. The reason to do this was checking if the private key generated is well formed and valid.

Test certificate To check if the generated certificates are standard, compliant with the specification, and usable by any Java application, we have executed the following steps:

- we have generated a new keystore using the "keytool"java utility.
- we have added to the keystore the certificates obtained with the CDA client application
- we have proceeded to sign a jar file with each certificate using the "jarsigner" java application.

3.7.8 Test log

Test log identifier: wp35-cda-ts01-tl02

Description

Tests where performed by Javier Noguera (T6). No special hardware was needed.

For the test we used:

1. computer: a laptop computer with dual core processor and 1Gb of memory
2. OS: Linux Ubuntu (desktop distribution)
3. other: Sun Microsystem's Java SDK

Application was tested only in one computer, not in a grid.

Execution Description see wp35-cda-ts01-tps02

In order to test the certificate, we created a Java class named Cipher.java that uses the private key to cipher and decipher data.

```
   java TestCipher
Params:  encrypt|decrypt keyFile file_input file_output
```

The source code of the application is:

```
 import java.io.*;
import java.security.spec.*;

import javax.crypto.*;
import javax.crypto.spec.*;

class TestCipher {
       public static void main(String[] a) {
              if (a.length<4) {
                 System.out.println("Params:");
                 System.out.println("encrypt|decrypt keyFile
                    (cont.)file_input file_output");
                 return;
```

```
    }

    String algorithm = "DESede";
    String mode = a[0];
    String keyFile = a[1];
    String input = a[2];
    String output = a[3];
    try {
       SecretKey ky = readKey(keyFile,algorithm);
       secretCipher(algorithm, mode, ky, input, output);
    } catch (Exception e) {
       System.out.println("Exception: "+e);
       return;
    }
}

private static SecretKey readKey(String input, String
   (cont.)algorithm)
    throws Exception {
    String fl = input;
    FileInputStream fis = new FileInputStream(fl);
    int kl = fis.available();
    byte[] kb = new byte[kl];
    fis.read(kb);
    fis.close();
    KeySpec ks = null;
    SecretKey ky = null;
    SecretKeyFactory kf = null;
    if (algorithm.equalsIgnoreCase("DESede")) {
       ks = new DESedeKeySpec(kb);
       kf = SecretKeyFactory.getInstance("DESede");
       ky = kf.generateSecret(ks);
    }

    return ky;
}

private static void secretCipher(String algorithm,
   (cont.)String mode,
    SecretKey ky, String input, String output) throws
       (cont.)Exception {
    Cipher cf = Cipher.getInstance(algorithm);
    if (mode.equalsIgnoreCase("encrypt"))
       cf.init(Cipher.ENCRYPT_MODE,ky);
    else if (mode.equalsIgnoreCase("decrypt"))
       cf.init(Cipher.DECRYPT_MODE,ky);
    else
       throw new Exception("Invalid mode: "+mode);

    FileInputStream fis = new FileInputStream(input);
    FileOutputStream fos = new FileOutputStream(output);
    int bufSize = 1024;
    byte[] buf = new byte[bufSize];
    int n = fis.read(buf,0,bufSize);
```

```
                int fisSize = 0;
                int fosSize = 0;
                while (n!=-1) {
                    fisSize += n;
                    byte[] out = cf.update(buf,0,n);
                    fosSize += out.length;
                    fos.write(out);
                    n = fis.read(buf,0,bufSize);
                }
                byte[] out = cf.doFinal();
                fosSize += out.length;
                fos.write(out);
                fis.close();
                fos.close();
            }
}
```

The same operation could be done using shell commands such as openssl.

Procedure Results The test succeeded. XOS-Certificates provided by the server after the user authentication can be used with generic tools such as openssl and Java security classes. This means that applications that want to manage XOS-Certificates can use standard tools and common algorithms in order to work with them.

Anomalous Events

There were no anomalous events during the execution of the test.

Incident Report Identifiers

There were no incidents to be reported during the execution of the test.

3.7.9 Test summary report – CDA

Test summary report identifier: wp35-cda-ts01-tsr

Summary

The CDA (first release for developments) was evaluated using Java programming language in test case wp35-cda-ts01-tcs01 and wp35-cda-ts01-tcs02. Test cases are related to the previously enumerated requirements: R86, R88, R89, and R91.

The first test case focused on authentication of the user against the Virtual Organization (VO), using the name of the organization, the name of the user and the password.

The second test is focused on the resulting certificates that can be used by the user to identify himself within the system.

Comprehensiveness Assessment

The testing process was in line with the approach given in wp35-cda-ts01-tp.

Summary of Results

All tests where executed successfully and they worked as expected. The user can get his own certificate once authenticated against the VO. The result is a compliant XOS-certificate. The Java test was very straightforward, due the fact that the original development was ready to run in the Java environment and we had no compilation problems.

Evaluation

Certificates are a necessary item in order to provide single sign-on, data integrity, identity and privacy (using encryption) of the information. More work is needed during the next months in order to integrate the CDA with the whole system.

Regarding the requirements, we arrive to the conclusion that all all requirements are partially fulfilled because, even if the CDA provides the tools, however, the fulfillment of the requirements can only be evaluated in an integrated system.

- R86, partially fulfilled using the XOS-Certificate for encrypting information
- R88, partially fulfilled using the XOS-Certificate for this purpose
- R89, partially fulfilled using the XOS-Certificate for validating users
- R91, partially fulfilled using a centralized server

Next steps should take into account the development of a specific API that helps application to integrate with the authentication method. The DBE application from T6 is considered as one suitable candidate for further testing the functionalities of the CDA as it uses standard java classes in order to manage and verify certificates. The interplay with other XtreemOS components can be tested when the integrated version of XtreemOS will become available.

3.8 Summary

The previous sections presented the test documentation and the evaluation results for the node-level VO support, the checkpointing and restart mechanism, LinuxSSI, SAGA API, XtreemFS and CDA. In the following the main results are summarized.

3.8.1 Overview of Evaluation Results

In this section, the evaluation results of the selected XtreemOS components are summarized. It must be noted that the evaluations have been performed on early releases which are still being improved. Therefore, performance and stability issues as well as incomplete functionalities have to be expected. This evaluation has a rather informative and guiding character. More substantiated evaluations can be carried out as soon as the integrated and packaged releases of XtreemOS become available.

Considering the **node-level VO support** of the XtreemOS PC flavor the current solution allows for logging into a node by means of a security token which is mapped one way to one non-privileged user and group ID. Potentially, this would prevent many Linux applications – in particular business and legacy applications – from using XtreemOS. The test results received by WP4.2 complement the evaluation in deliverable D3.5.4 [5] which gives an overview of the disadvantages: installers are not able to operate since they are not allowed to configure the system appropriately. Furthermore, installers are not able to create additional users or groups which are needed to execute the application. The static mapping of GlobalUserID to local UID may also result in conflicts as legacy applications may dictate a certain naming convention. It also needs to be checked if the behavior of the proposed solution is compliant with POSIX. D3.5.4 further elaborates on the security issues arising from the isolation via the user context. In particular, the proposed approach does not allow for a separation of name spaces nor any fine-grained access control mechanisms. Furthermore, to the user the proposed solution does not give the illusion of being alone on the platform. Various Linux applications require to change system variables which in turn may influence the operation of other applications. Thus, the proposed approach does not accomplish an appropriate degree of isolation. Deliverable D3.5.4 already provides various recommendations and inspiration for work arounds. Furthermore, the experiments in deliverable D4.2.4 revealed that after a successful login is done on behalf of the VO in the amappedfile, the same user can log in on behalf of another (incorrect) VO and is mapped either to the same local account or to root. The bug represents a fatal security hole in the component. WP2.1 has been notified about the problems.

Regarding the **checkpointing and restart mechanism**, various applications with different complexities have been used for evaluation. Only a very simple applications printing a counter value on stdout could be checkpointed and restarted successfully. Another multithreaded Java application could only be checkpointed after disabling name caching for passwd/groups. A simple text editor like *gedit* could not be restarted from the checkpoint. SAP WebAS cannot be started normally after installing checkpointing and restart software. The corresponding output provides the information that some dynamic libraries could not be handled properly. Thus, the SAP WebAS cannot be used at all in combination with the currently provided checkpointing and restarting functionality. Further problems were reported for a Java application using communication sockets. Even when the ap-

plication was able to restart, communication sockets where not valid anymore, and the communication with the clients was lost. The error message received is related to Java classes, and it indicates that the socket identifier is not available anymore. The main application would have to be changed in order to cope with this issue. The failure in open server sockets would have to be controlled by the application, reopening it in case the binding fails. It turned out that using the Linux term signal when checkpointing an application often prevents the application from restarting correctly. This is because the term signal automatically cleans any temporary files that the process was using. The kill signal does not do any clean up and will allow the applications to restart correctly, however, in the case of migration, any temporary files or environment settings will need to be accessible from the new node. When the distributed file system is incorporated with the checkpointing module it should be able to support this easily. It should be considered that at this early stage of the development, it is no surprise to find that there were issues of robustness and problems trying to execute some of the tests. The developers have been informed and it is expected the these issues can be resolved in the following releases.

A further sequence of tests has been conducted on **LinuxSSI** to evaluate the following features with the GALEB and ELFIPOLE applications: process migration, checkpointing, scheduling, SMP support and the customizable scheduler. The latter two features could not be tested as both were not available in the last stable version (Kerrighed 2.1.1). Therefore development versions were used, however, these were not stable enough to perform the tests. At the time of testing, the 64 bit version was also not ready for evaluation. Some SSI features were found to be working properly: Unified /proc (it was possible to monitor and kill processes from any nodes of the cluster) and migration of simple processes. Regarding the migration feature, it must be noted that restarting a checkpointed parallel application never worked correctly. Moreover, a stand-alone calculating process failed to restart from a checkpoint that was made after migration. A further potential problem may be that the restarted process always carries the same PID as the original one, so restart is not possible if there exists another process with the same PID. The described errors are critical bugs in LinuxSSI and as such should be fixed quickly. They prevent testing more complicated combined migration/checkpointing scenarios. More problems and bugs were reported to the Kerrighed team considering, e.g., the Kerrighed TIPC module, the deployment via nfsboot, limitations in the krgadm cluster start command, or the installation procedure using make install. Finally, setting up Kerrighed turned out to be very time consuming. Therefore it is recommended that WP2.2 maintains a ready-to-use installation of the latest stable Kerrighed versions on the Grid5000 testbed. The testbed installation of Kerrighed on Grid5000 should fulfill the following requirements: KVM like access (using Kaconsole), easy deployment (using Kadeploy), support for SMP and 64bit CPUs.

The **SAGA API** for C++ could be tested on a single Linux machine using the ZEPHYR application. Although the tests could not be run through the XtreemOS execution environment (SAGA API has not been ported on XtreemOS yet), one could easily prepare, start, monitor, end or cancel a job, which is very encouraging.

An exception error was encountered wrong probing an ended job. The status of this job is not readable anymore causing the exception. This behavior does not agree with the SAGA specification stating that the status of the terminated job should be “done”. A further problem was discovered: After calling the wait() method, the job status also remains According to the SAGA specification, however, the status should either be “canceled”, “done” or “failed”. SAGA successfully passed the file management tests, one could easily create, rename and delete files. The full POSIX compliance still needs to be evaluated in the forthcoming releases of SAGA.

The file system **XtreemFS** (internal release 1) was evaluated using the applications GSDNA, maxDB and WISS. It can be concluded that the most important POSIX functionality is implemented, which already enables execution of many applications. However, changing permissions and mmap system calls were not implemented. Furthermore, problems with POSIX rights were found. After copying or creating files, all user rights had been set to "xrw". This can be a serious problem for applications which rely on file access rights. For simple/standard configurations XtreemFS was rather stable, no data corruption has been detected during the tests. The compilation and installation process of XtreemFS is relatively simple, no special knowledge or skills are required. Some of the tested applications were able to run on the tested XtreemFS version only with some workarounds that compensate only partly implemented POSIX standards. Stability and reliability characteristics have to be improved in the next releases of XtreemFS, e.g. the Bonnie benchmark on large files of 2 GB caused a reproducible error on fclose system calls. Handling non-standard/concurrent configurations has to be implemented and tested more carefully. XtreemFS could not profit from more hardware resources (scalability issues). Finally, the absolute I/O performance (e.g. for file scanning) of the internal release 1 was unsatisfactory and far below the raw hardware (or native OS) capabilities.

All tests with the **Credential Distribution Agency (CDA)** were executed successfully and they worked as expected. The user can get his own certificate once authenticated against the VO. The result is a XOS-certificate for which no compliance problems could be discovered. Next steps should include the development of a specific API that facilitates applications to integrate authentication method. The interplay with other XtreemOS components can be tested when the integrated version of XtreemOS will become available.

3.8.2 Assessment of the Requirements Fulfillment Status

The experiments allowed for testing the functionalities of various isolated XtreemOS components since an integrated version was not available for evaluation. The documentations of these tests also include a tentative insight in how far certain requirements are fulfilled. In this section, we give an overview of these assessments which have been consolidated in the table below. A color coding is applied to visualize the fulfillment status:

- red: requirement is not fulfilled.
- yellow: requirement is "partially" fulfilled.
- green: requirement is fulfilled.
- grey: fulfillment of requirement could not be tested (e.g. respective functionality was not available for testing)

If multiple tested components contribute to a certain requirement indented rows are added containing the assessment for each such component (subsequently referred to as sub-assessment). In this case, the overall assessment of the fulfillment of the requirement can not be better than the worst sub-assessment of all contributing components. Regarding requirements which have been affected by the revision process for this deliverable, we added character “n” (new requirement) or character “u” (updated requirement) behind the requirement number in the first column of the table. Since these requirements have been changed or introduced only recently, the assessment of the fulfillment status has to be considered as tentative. Overall, this overview provides feedback to the developers and to project management using it to identify gaps, errors, white areas and to trace the progress of the project with respect to the fulfillment of requirements. Note, for reasons of providing a compact and consolidated overview of the requirements fulfillment, the table does not show requirement titles nor descriptions. Hence, it is proposed to read the table in the electronic version of the document to make use of the clickable links to the requirements text given in the Appendix.

R.no.	Fulfillment	References	Comments
R1	red	wp34-xfs-ts01-maxdb01-tds01	Scalability and performance of XtreemFS data management were rather unsatisfactory
R2	grey		Integrated XtreemOS version needed for evaluation
R3	grey		Integrated XtreemOS version needed for evaluation to be done on large Grid testbed.
R4	grey		
R5	grey		
R6	red		tests of migration outside SSI cluster needed
	red	wp22-fm-ts01-galeb01-tcs01 and wp22-fm-ts01-elfi01-tcs01	has bugs, see wp22-fm-ts01-galeb01-tir01, wp22-fm-ts01-galeb01-tir03 and wp22-fm-ts01-elfi01-tir01
	red	wp21-cr-ts01-tsr	Checkpointing and restart worked only on simple programs without communication and without parallelism.

R7 (n)	grey		Not evaluated yet, new requirement.
R8	red	wp34-xfs-ts01-tp	XtreemFS failed benchmark tests on large files (2 GB). Could not test XtreemFS on very large data because of lack of hardware resources.
R9	grey		
R10	grey		
R11 (n)	grey		Not evaluated yet, new requirement.
R12	grey		
R13	grey		
R14	grey		
R15	grey		
R16	grey	wp22-fm-ts01-tsr	LinuxSSI (64 bit version) was not ready for evaluation.
R17	grey		
R18	red		
R19 (u)	grey		
R20	grey		
R21	grey		
R22	yellow		tests on other components needed
	green	wp21-vo-ts01-galeb01-tcs01, wp21-vo-ts01-galeb01-tcs03, wp21-vo-ts01-jcae01-tcs01	
R23 (u)	red		differentiated user and groups within a VO not supported yet (cf. Chapter 2 and D3.5.4 [5])
R24	yellow		tests on other components needed
	yellow	wp21-vo-ts01-galeb01-tcs01, wp21-vo-ts01-galeb01-tcs03, wp21-vo-ts01-jcae01-tcs01	managing of resource (node) usage OK, global VO management not tested
R25	red		tests on other components needed
	red	wp21-vo-ts01-galeb01-tcs02, wp21-vo-ts01-galeb01-tcs03, wp21-vo-ts01-jcae01-tcs01	has bugs – see wp21-vo-ts01-galeb01-tir02
R26	red	wp21-vo-ts01-galeb01-tcs01, wp21-vo-ts01-galeb01-tcs03	not fulfilled in wp21-vo; can possibly be implemented elsewhere
R27	grey		
R28	yellow		testing with other components needed

	green	wp21-vo-ts01-galeb01-tcs01, wp21-vo-ts01-galeb01-tcs02, wp21-vo-ts01-galeb01-tcs03	
R29	grey		
R30	grey		
R31	grey		
R32	yellow		tests on other components needed
	yellow	wp22-fm-ts01-galeb01-tcs01	see eval. of R47
R33	red	wp21-cr-ts01-spec01-tcs01, wp21-cr-ts01-webas01-tcs01, wp21-cr-ts01-webas01-tcs01	Errors in both test cases suggest that necessary files are being deleted on shutdown, and preventing original environment from restarting.
R34	grey		
R35	yellow	wp21-cr-ts01-spec01-tcs01, wp21-cr-ts01-webas01-tcs01, wp21-cr-ts01-webas01-tcs01	partially fulfilled
R36	grey		
R37	yellow	wp21-cr-ts01-spec01-tcs01, wp21-cr-ts01-webas01-tcs01, wp21-cr-ts01-webas01-tcs02, wp21-cr-ts01-spec01-tcs01	Implemented as a kernel module, however some problems occurred while installing on certain kernel versions.
R38	grey		
R39	red		tests on other components needed
	red	wp22-fm-ts01-galeb01-tcs01	IPC is not saved correctly – see wp22-fm-ts01-galeb01-tir02)
	red	wp21-cr-ts01-spec01-tcs01	Name Service Caching Daemon (NSCD) causing problems; see wp21-cr-ts01-spec01-tl01
R40	grey		
R41 (n)	grey		Not evaluated yet, new requirement.
R42 (u)	grey		
R43 (u)	grey		
R44	grey		
R45	grey		
R46	grey		
R47	yellow	wp22-fm-ts01-galeb01-tcs01	automatic failure detection not yet implemented; also, bug described in wp22-fm-ts01-galeb01-tir03 could be correlated to checkpoint/restart
R48	red		

R49 (n)	red	cf. Chapter 2 and D3.5.4 [5]	Various applications will not be able to be installed within a VO. Differentiated user and groups within a VO not supported yet.
R50	yellow	wp34-xfs-ts01-gsdna01-tds01, wp34-xfs-ts01-gsdna01-tds01, wp34-xfs-ts01-wiss01-tds01	partly implemented, e.g. ACL is not supported
	yellow	wp34-xfs-ts01-gsdna01-tds01	XtreemFS
	green	wp34-xfs-ts01-maxdb01-tds01	XtreemFS
	yellow	wp34-xfs-ts01-wiss01-tds01	XtreemFS
	red	wp31-api-ts01-zephyr01-tds01	not implemented yet. Still the SAGA specification makes several reference to POSIX. To be checked further and later
R51	red	wp31-api-ts01-zephyr01-tds01	not implemented yet, but a C++ API is already available on Linux regular boxes.
R52	red	wp31-api-ts01-zephyr01-tds01	not implemented yet
R53 (u)	grey		
R54 (u)	grey		
R55 (u)	grey		
R56 (u)	grey		
R57	grey		
R58	grey		
R59	grey		
R60	grey		
R61	grey		
R62	grey		
R63	grey		
R64	grey		
R65	grey		
R66	grey		
R67	grey		
R68	grey		
R69	grey		
R70	grey		
R71	grey		
R72	grey		
R73	grey		
R74	red	wp34-xfs-ts01-gsdna01-tir03	XtreemFS hangs sometimes if file operations are executed from several hosts

R75	yellow	wp34-xfs-ts01-gsdna01-tds01	VO and XtreemFS integration is not completed, UNIX file permissions are not fully implemented
R76	green	wp34-xfs-ts01-tp	no problems with directory support observed
R77	yellow	wp34-xfs-ts01-gsdna01-tds01	Access right management is not fully implemented by XtreemFS
R78	grey		
R79	grey	wp34-xfs-ts01-tp	Internal release 1 of XtreemFS is not intended for performance tests
R80	grey		
R81	grey		
R82	grey		
R83	yellow	wp34-xfs-ts01-wiss01-tds02	Only the basic functionality could be tested, transactions are scheduled for later testing.
R84	grey		
R85	yellow		testing on other components needed
	yellow	wp21-vo-ts01-galeb01-tcs02, wp21-vo-ts01-galeb01-tcs03	referenced tests OK, further testing needed (of group-level access and access to other objects)
	grey	wp34-xfs-ts01-gsdna01-tds01, wp34-xfs-ts01-wiss01-tds01	ACLs are not yet supported in XtreemFS
R86	yellow		testing on other components needed
	yellow	wp21-vo-ts01-galeb01-tcs02, wp21-vo-ts01-galeb01-tcs03	referenced tests OK, further testing needed
	yellow	wp35-cda-ts01-tds01	referenced tests OK for using XOS-certificates for encrypting informations, further testing on integrated XtreemOS needed.
R87	grey		
R88	yellow	wp35-cda-ts01-tds01	referenced tests OK for using XOS-certificates for ensuring integrity, further testing on integrated XtreemOS needed.
R89	red		Further testing on integrated XtreemOS needed.
	red	wp21-vo-ts01-galeb01-tcs01, wp21-vo-ts01-galeb01-tcs03	subject of the user's certificate proxy must be entered into *ammappedfile*
	yellow	wp35-cda-ts01-tds01	referenced tests OK for using XOS-certificates for validating users.

R90	grey		
R91	yellow	wp35-cda-ts01-tds01	referenced tests OK, further testing on integrated XtreemOS needed.
R92	grey		
R93	grey		
R94	grey		
R95	grey		
R96	yellow		tests on other components needed
	green	wp21-vo-ts01-galeb01-tcs01, wp21-vo-ts01-galeb01-tcs02, wp21-vo-ts01-galeb01-tcs03	
R97	grey		
R98	grey		
R99	grey		
R100	grey		
R101	grey		
R102	grey		
R103	grey		
R104	grey		
R105	grey		
R106	grey		
R107	grey		
R108	grey		

Out of the 108 requirements defined by WP4.2, 15 requirements have been rated as not fulfilled, 16 requirements are fulfilled partially, and one requirement is rated as fulfilled. For the remaining 78 requirements, it was not possible to provide an assessment due to the lack of an integrated XtreemOS version and only a few XtreemOS components were ready for evaluation at the time of testing. In this deliverable, only early (development) releases could be evaluated which will be subject to improvement in the coming project phases. Therefore, the table above can only give a preliminary assessment of the requirement fulfillment. It is intended to maintain this assessment technique throughout the coming evaluations in order to trace to progress in the fulfillment of the requirements.

Chapter 4

Conclusion

The deliverable provides the updated and extended set of application requirements which have been discussed in cooperation with the development work packages. The revision was driven by the ongoing design and development activities, the evolution of the project vision and also by the deeper insights into the interplay between XtreemOS and the application references. The developers of XtreemOS need to be aware of the entire set of requirements as many requirements are interdependent thereby affecting multiple development work packages. The requirements serve as a source of inspiration and as a guideline. However, the distribution of responsibilities and the coordination of work regarding the implementation of the required functionalities belong to the responsibilities of the project and technical management as well as the work package leaders in SP2/SP3. During the upcoming phases of the project, the requirements will continuously be revised and extended if necessary, driven, e.g., by evaluation results and by the activities in SP2 and SP3.

The second main contribution of this deliverable is the evaluation of the first internal releases of various XtreemOS components comprising node-level VO support (from WP2.1), checkpoint and restart (from WP2.1), LinuxSSI (from WP2.2), SAGA API (from WP3.1), XtreemFS (from WP3.4) and CDA (from WP3.5). The deliverable provides the entire test documentation based on the IEEE standard 829-1998 including test plans, specification, logs and results, which allows to easily access relevant information and to retrace test procedures. Test results were summarized to give a first overview of the fulfillment status of the application requirements.

The main findings of the evaluation can be summarized as follows:

- The current approach for providing node-level VO support allows for logging into a node by means of a security token which is mapped one way to one non-privileged user and group ID. This approach may prevent many Linux applications – in particular business and legacy applications – from using XtreemOS. Here we support the security evaluation published in D3.5.4 [5]. As the node-level VO support is the base technology for sharing distributed resources in the Grid, the adoption of XtreemOS among Linux users

and developers may be endangered. Furthermore, a security hole was reported which would allow a user to log in on behalf of another (incorrect) VO with a mapping to either the same local account or to root.

- With the checkpointing mechanism for the XtreemOS PC flavor it was possible to checkpoint and restart simple local applications. In forthcoming releases, it is expected that also more complicated applications, e.g. parallel and multithreaded programs using communication sockets can be handled successfully.

- During the evaluation of LinuxSSI/Kerrighed it was possible to use a unified /proc (which allows for monitoring and and killing processes from any node of the cluster) and to migrate simple processes. Unfortunately, SMP and 64 bit support and the customizable scheduler were not available in stable releases at the time of testing. The evaluation allowed for identifying various bugs which were reported to the Kerrighed developers.

- No major issues could be identified during the evaluation of the job and file management features of the SAGA API for C++. So far, it was only possible to run SAGA on a single Linux machine using local adaptors. More in-depth experiments can be carried out after XtreemOS adaptors will have been implemented which will allow for communicating with XtreemOS services.

- Regarding the evaluation of XtreemFS, it can be concluded that the distributed file system already supports the most important POSIX functionality. Furthermore, no problems on data integrity could be identified. However, XtreemFS needs to be improved in terms stability, performance as well as handling of large files and access rights. It must be noted that recently internal release 2 has been dispatched, which could not be taken into consideration for the deliverable at hand. For internal release 2, it was reported that stability has been improved and that it passed a wide range of tests, including Bonnie and IOZone, both on striped and non-striped volumes. Moreover, the new version is supposed to provide SSL support, links and authorization. The new capabilities and features belong to the potential scope of forthcoming evaluations of XtreemFS.

- No problems were encountered during the preliminary evaluation of the Credential Distribution Agency (CDA). It is recommended to implement an API which allows applications for integrating authentication methods. A more thorough evaluation of the CDA as part of an entire security solution can be tested when an integrated XtreemOS version will have been released.

In succeeding deliverables, we will continue the evaluation of XtreemOS (also including new components and packaged releases provided by WP4.1). All experiments in this deliverable were carried out on the local testbeds of the various partners. After the successful completion of inhouse tests with the integrated version

of XtreemOS, the experiments can be repeated on GRID5000 on a larger scale with geographical dispersion among computational nodes. Depending on the availability and stability of the integrated release, the results of the GRID5000 evaluation are also foreseen to be reported in forthcoming deliverables of WP4.2.

The evaluation reports and in particular the results of the evaluation are being communicated through various channels, including wiki pages, mailing lists, bug trackers, phone calls and finally this deliverable. All intermediate and final test documents are made available on the internal SVN server (`https://scm.gforge.inria.fr/svn/xtreemos-deliv/WP4.2/experiments`). We hope that the evaluation provides useful feedback to the development work packages and we are looking forward to a successful continuation of the cooperation established.

Chapter 5

Acknowledgments

We would like to thank the developers in SP2 and SP3 and in particular the respective work package leaders for their cooperation regarding the identification of interesting test scenarios and for their support during the installation XtreemOS components. Furthermore, they provided useful feedback inspiring the revision of requirements.

Bibliography

[1] XtreemOS consortium. Early experiments and evaluation. XtreemOS deliverable D4.2.2, 2006.

[2] XtreemOS consortium. Requirements capture and use case scenarios. XtreemOS deliverable D4.2.1, URL: http://www.xtreemos.eu/publications/project-deliverables/d4-2-1_main.pdf, 2006.

[3] XtreemOS consortium. Application references, requirements, use cases and experiments. XtreemOS deliverable D4.2.3, 2007.

[4] XtreemOS consortium. Design and implementation of node-level vo support. XtreemOS deliverable D2.1.2, 2007.

[5] XtreemOS consortium. Second specification of security services. XtreemOS deliverable D3.5.4, 2007.

[6] XtreemOS consortium. Security services prototype month 18. XtreemOS deliverable D3.5.5, 2007.

[7] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150, 2001.

[8] IEEE. Ieee standard for software test documentation, ieee 829-1998. IEEE Computer Society, 1998.

Appendix A

Requirements

In this appendix, we present the updated and extended set of requirements. The changes and extensions of requirements were explained in greater detail in Chapter 2. Due the number of extensions modifications performed on the requirements, it was decided that deliverable D4.2.4 should provide more than a list of changes such that the partners in the consortium have access to the entire updated requirements catalogue as a whole with no need to cross-reference deliverables D4.2.1 or D4.2.3.

As before, the requirements are maintained according to the following structure:

- General requirements
- Virtual organization support in XtreemOS
- Checkpointing and restart
- Federation management
- XtreemOS interfaces
- Highly available and scalable Grid services
- Data management
- Security in virtual organizations
- Support for mobiles devices

Although this structure suggests a close relation to the organization of work packages, the requirements are strongly inter-related and inter-dependent. Therefore, the members of SP2 and SP3 are asked to read all the requirements to asses the relevance for their own work. The requirements are presented by their requirement number (Rx), a requirement title and a corresponding detailed description. Furthermore, the following fields provide additional information:

Previous number (D4.2.3): In order to maintain a continuous numbering within this document, some requirement numbers are changed due to the insertion of one additional requirement. Therefore we provide a reference to the requirement number used in deliverable D4.2.3.

Importance: "Obligatory" indicates that the respective requirement must be implemented to ensure that the applications can be executed, whereas "Optional" means that the specific requirement would increase the usability of XtreemOS. However, in the "Optional" case, the applications can still be executed without the requested additional functionality.

Updated: This field signifies whether the title, description or importance of the requirement has been changed in comparison to Deliverable D4.2.3.

Importance for MDs: This field, if given, indicates whether the importance or relevance of the respective requirement is different for the mobile devices (MDs) flavor of XtreemOS.

Implementation order: On a scale between 1 (this functionality has to be provided very early) and 10 (may be implemented during later phases of the project) the implementation order (average over all responses received) proposes a guideline in which phases of the project lifetime the requested functionality should be provided.

A.1 General Requirements

R1: XtreemOS equally supports data-intensive and computation-intensive applications

Most of the applications considered are either data-intensive or computation-intensive. Some of them (SPECWEB, SIMEON) are both whereas SRC, IMA and JOBMA are neither data nor computation-intensive. Regarding data-intensive applications it is essential that XtreemOS can efficiently manage access to central and distributed databases and can also handle file-based applications (cf. requirements for data management).

Previous number (D4.2.3): R1
Updated: no
Importance: obligatory
Importance for MDs: optional
Implementation order: 2.9

R2: XtreemOS supports heterogeneous hardware

The XtreemOS system shall run on heterogeneous nodes with different architectures and different memory & storage capacities as visualized in Figure A.1. Some

applications are programmed to be platform-independent and do therefore support heterogeneous architectures. Certain applications like ELFIPOLE or ZEPHYR benefit from homogeneous architectures to facilitate runtime prediction. Within

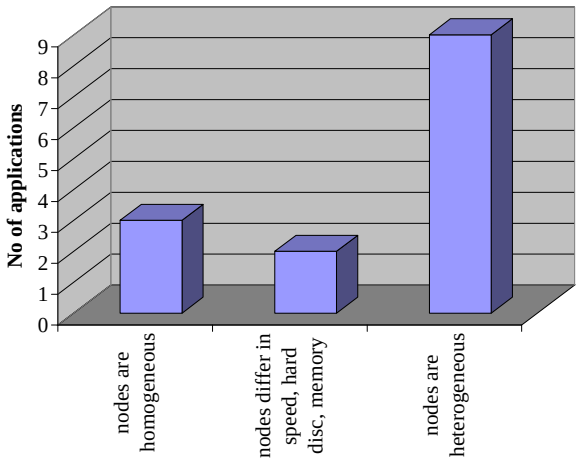


Figure A.1: Hardware diversity

a single federation, heterogeneity that must be supported includes different CPUs and different amount of memory, processor clockspeed etc.

Previous number (D4.2.3): R2
Updated: no
Importance: obligatory
Importance for MDs: Only ARM architecture
Implementation order: 10.0

R3: XtreemOS must support Grids with variable number of nodes

The XtreemOS system must be able to handle from a few up to more than thousand of computing resources.

Figure A.2 demonstrates that 5 applications require relatively "small" Grids whereas the majority of applications may also demand Grids with up to thousands of nodes.

Previous number (D4.2.3): R3
Updated: no
Importance: obligatory
Implementation order: 3.5

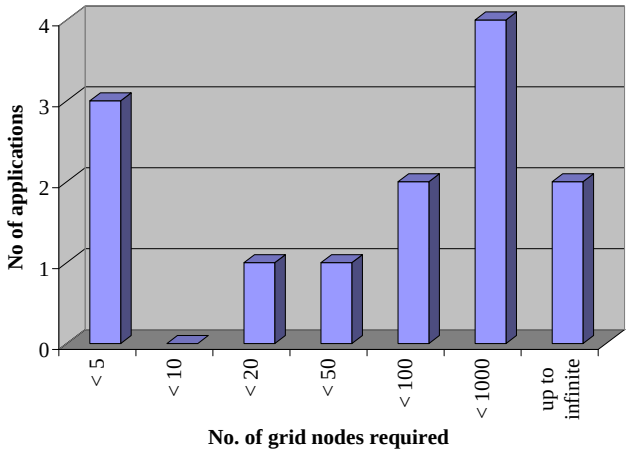


Figure A.2: Number of Grid nodes required by the applications

R4: XtreemOS needs to handle virtual (replicated) nodes in order to increase robustness in case of failures

XtreemOS should provide two ways to increase robustness: a checkpoint/restart mechanism, as defined in deliverables D2.1.1 and D2.2.1, and a replication mechanism termed virtual nodes, as defined in D3.2.1.

Using virtual nodes, a service, application or their individual parts whose robustness and availability is critical is in fact being run replicated on multiple nodes. This is seen by the application as a single "virtual" node. On node or link failure, another copy can take its place immediately, which makes this mechanism suitable for interactive applications and for services whose availability is critical. Resource usage is multiplied by the number of replicas.

Four applications require and benefit from virtual nodes. Typically, the number of required replicas is low (1 or 2) whereas DBE can benefit from as many replicas as possible.

See also R46 for more details on how to extend this mechanism to the federation level.

Mechanisms/suggestions: the application specifies which processes are critical and therefore require replication. The number of replicas can be specified by the application as well. Alternatively, a mechanism could allow specifying the desired robustness, after which the system would choose a suitable number of replicas, taking into account the estimated robustness of each node.

Previous number (D4.2.3): R4
Updated: no

Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 5.7

R5: XtreemOS must support that nodes can dynamically be added to the Grid and removed from the Grid

During runtime XtreemOS shall be able to dynamically adapt to the available amount of resources. It must be possible to add new nodes to the Grid. Furthermore, the scheduling system must consider these additional resources. If nodes are removed from the Grid (for various reasons) XtreemOS must handle the migration of running applications (see R6). This capability to dynamically react to a changing number of Grid nodes is required by about half of the applications.

Previous number (D4.2.3): R5
Updated: no
Importance: obligatory
Implementation order: 3.7

R6: XtreemOS must support migration of running applications

XtreemOS must support the migration of running applications. The migration functionality must be customizable allowing e.g. to specify the maximum time to complete this migration or to explicitly allow/deny migrations.

Previous number (D4.2.3): R6
Updated: no
Importance: obligatory
Importance for MDs: optional
Implementation order: 3.7 (corresponds to R5)

R7: Execution and Migration of Java Applications

It must be possible to at least migrate an individual JVM and the Java application running in it as a whole.

Previous number (D4.2.3): new
Updated: yes, new
Importance: obligatory
Importance for MDs: not applicable
Implementation order: to be determined

R8: XtreemOS must support Grid nodes with up to several Terabytes storage capacity

The storage requirements range from only a few MBs up to more than 500 GB. However the actual amount of storage required depends on the size of input/output

data and the number of nodes. Furthermore WEBAS requires that Grid nodes containing the database can store up to 5 TB of data.

Previous number (D4.2.3): R7
Updated: no
Importance: obligatory
Implementation order: 8.1

R9: XtreemOS must support Grid nodes with up to several Gigabytes working memory capacity

The working memory requirements vary between a few MBs and several GBs. However the actual amount of memory required depends on the size of input/output data and the number of nodes.

Previous number (D4.2.3): R8
Updated: no
Importance: obligatory
Implementation order: 4.6

R10: XtreemOS needs to provide for software licensing mechanisms

Several applications require local license files or license servers for their execution. Accordingly, XtreemOS must provide for the distribution of license files to the respective Grid nodes (respectively connectivity to license servers) taking into account that license files may be bound to specific MAC/IP-addresses or dongles. It must also be ensured that licensing software is installed in the Grid nodes prior to starting the applications. Overall, this requirement demands that the scheduling mechanism must incorporate the connectivity between nodes.

Previous number (D4.2.3): R9
Updated: no
Importance: obligatory
Implementation order: 6.0

R11: Packaging of XtreemOS releases

In order to increase the acceptance of XtreemOS in the users and developers community it is required that all intermediate and the final releases of XtreemOS are distributed as self-installing software packages including but not restricted to the rpm and deb format. These packages must be compatible with at least the following Linux distributions: Mandriva, Red Flag, and Debian.

Previous number (D4.2.3): new
Updated: yes, new
Importance: obligatory
Implementation order: to be determined

R12: XtreemOS must provide for fast and reliable communication

The performance of applications relies very much on fast and reliable network connections. Therefore, XtreemOS has to ensure that the available network resources are managed and used efficiently. Especially data-intensive applications require high-speed connection to transfer large data sets or to process frequent database transactions. Several applications demand a permanent connectivity between nodes. Reliable connectivity is also a crucial requirement by various applications, which may be tremendously affected by network failures (see Figure A.3).

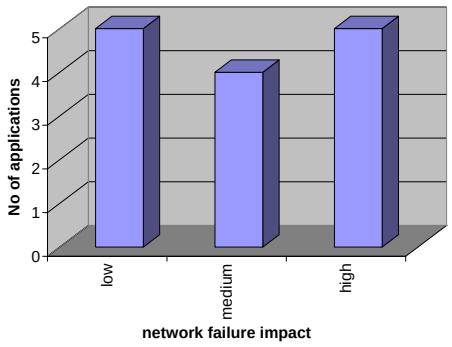


Figure A.3: Expected impact of network failures

When mapping applications onto Grid resources XtreemOS also must take into account the respective network characteristics. It should therefore be possible that applications can define certain parameters like maximum network delay, minimum bandwidth and reliability. These parameters need to be considered by XtreemOS when allocating and re-allocating applications.

Previous number (D4.2.3): R10
Updated: no
Importance: obligatory
Implementation order: 3.4

R13: XtreemOS must be compatible with highspeed interconnect standards

XtreemOS must support highspeed communication including Infiniband, Myrinet and SCI. Regarding the SSI flavour of XtreemOS, support for Infiniband is obligatory.

Previous number (D4.2.3): R11
Updated: no
Importance: obligatory

Implementation order: to be determined

R14: XtreemOS must support IPv6

The XtreemOS must support IPv4 and IPv6.

Previous number (D4.2.3): R12
Updated: no
Importance: obligatory
Implementation order: 7.5

R15: Semaphores

XtreemOS must support semaphores, in particular System 5 semaphores are required.

Previous number (D4.2.3): R13
Updated: no
Importance: obligatory
Implementation order: To be determined

R16: XtreemOS must support 64 bit architectures

Since almost no pure 32 bit systems are sold anymore in today's HPC market. Furthermore several upcoming releases of applications will only be executable on 64-bit processors. Therefore it is essential that XtreemOS supports x86_64 and Power PC architectures.

Previous number (D4.2.3): R14
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable (No MD currently uses 64 bit architecture)
Implementation order: 2.4

R17: SMP and multicore support

XtreemOS must support SMP and multicore architectures.

Previous number (D4.2.3): R15
Updated: no
Importance: obligatory
Implementation order: to be determined

R18: XtreemOS must support virtual machines (e.g. XEN, VMWare, OpenVZ/ Virtuozzo)

This requirement is actually twofold.

1. XtreemOS should be able to run inside a virtual machine.
2. XtreemOS should be able to run virtual machines (i.e. act as the host operating system).

The second point is important for business applications with strict isolation requirements between different VOs. This is especially true for the case of multiple VOs sharing the same physical hardware. For such scenarios with strong isolation requirements it must therefore be possible to execute applications in VMs or virtual containers/compartments, and a VO is created across a group of selected VMs/containers.

Remark: This requirement is closely related to the other security requirements in this document.

Previous number (D4.2.3): R16
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 6.0

R19: XtreemOS shall support multicast

It is required that XtreemOS supports multicast within a cluster.

Previous number (D4.2.3): R17
Updated: yes
Importance: optional
Implementation order: 8.0

R20: XtreemOS needs to provide access to various Grid services

The main Grid services required by the applications are file transfer, resource discovery and job submission, as visualized in Figure A.4.

Further services required are related to scheduling and monitoring jobs as well as viewing and modifying user information, in particular user location, presence information etc. Various applications also offer Grid/web services, e.g., resource scheduling or resource and file management, instant messaging or job monitoring. Therefore it must ensured that these services can be accessed and executed. Services and resources are discovered mainly by Uniform Resource Identifiers (URI), full description or by an approximate/semantic description (cf. Figure A.5).

Previous number (D4.2.3): R18
Updated: no

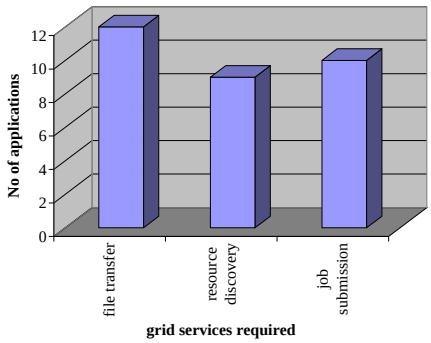


Figure A.4: Main Grid services required

Importance: obligatory
Implementation order: 2.8

R21: XtreemOS VOs shall manage a large number of users

The number of users within a VO is very different for the applications considered. About half of the applications can be executed with at most 4 different users whereas the other half demands to manage several thousands of users, which may also concurrently work with the application.

Previous number (D4.2.3): R19
Updated: no
Importance: obligatory
Implementation order: 4.7

R22: XtreemOS must support the execution of interactive and batch jobs

The majority of applications in WP4.2 is executed interactively as shown in Figure A.6. Some of them are purely interactive programs whereas others are combinations of interactive and automated steps. Several applications may be started in either interactive or batch mode. Five applications are typically run in batch mode. Consequently, XtreemOS must be able to facilitate the distribution and execution of both, interactive applications and batch jobs.

Previous number (D4.2.3): R20
Updated: no
Importance: obligatory
Importance for MDs: optional
Implementation order: 2.2

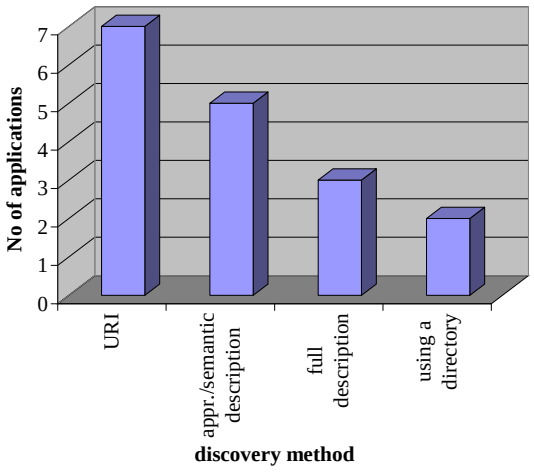


Figure A.5: Methods for resource and service discovery

R23: XtreemOS VOs must provide role management

XtreemOS must be able to manage users through roles – each role having its own rights. Users must be able to read, write, change and delete files and to execute applications. Administrators need the permission for installation, maintenance and to manage accounts (add/remove accounts, change of permissions). Furthermore, it must be possible to execute different parts of the applications, e.g. the database in a multi-tier business application stack, in the context of different users and different groups.

Remark: Middleware systems like WEBAS also require a complex role management at the application layer.

Previous number (D4.2.3): R21
Updated: yes
Importance: obligatory
Implementation order: 7.2

A.2 Virtual Organization Support in XtreemOS

R24: XtreemOS has to provide the means to manage VOs

VOs must be manageable. Management actions include the creation, change and destruction of a VO. Three different roles will be involved in managing VOs: domain administrators, VO administrators and VO users. Domain administrators

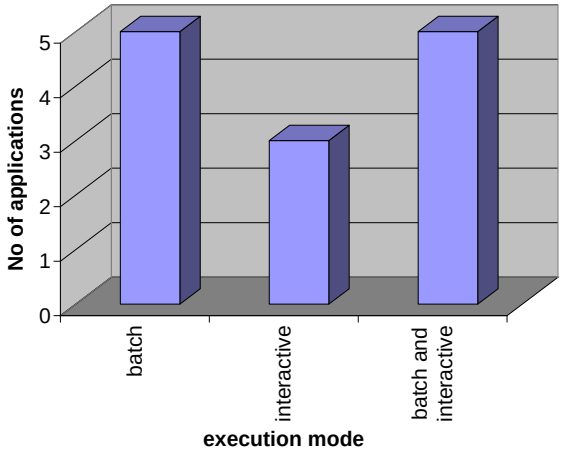


Figure A.6: Application execution modes

maintain a pool of resources that are allowed to be integrated into a VO. VO administrators are allowed to compose VOs from the resources provided by various domains. They are also responsible for creating VO user accounts and for configuring and modifying the permissions that VO users have within a VO. Participating institutions agree on one or more persons to be VO administrators. Therefore it must also be possible that each participating institution has at least one person that acts as VO administrator. VO users also require the right to manage a VO. The respective (restricted) permissions are granted by the VO administrator. This allows for instance that a VO is created when an application is started and the VO is destroyed after the termination of the application.

Previous number (D4.2.3): R22
Updated: no
Importance: obligatory
Implementation order: 2.5

R25: VO user accounts have to be independent from local user accounts

A person can have a user account in her/his local domain A and a VO user account in a VO comprising domains B, C and D. To obtain a VO user account it is not necessary to have a local user account in one of the domains belonging to the VO. The VO user need not have a local account anywhere in the Grid at all. However, it must be possible to transfer data, files and directories between local and VO accounts (e.g., by copy or mount).

Previous number (D4.2.3): R23

Updated: no
Importance: obligatory
Implementation order: 5.2

R26: XtreemOS allows to dynamically change the composition of VOs during application runtime

It must be possible to change the composition of resources within VOs while applications are executed. This dynamic adaptability is needed e.g. if certain computing or communication resources fail. In this case, the unavailable resources need to be automatically substituted by alternative resources (if available). This alternative resource is chosen from the pool of available resources as agreed (by contract) among the participating institutions.

Previous number (D4.2.3): R24
Updated: no
Importance: obligatory
Implementation order: 7.4

R27: The lifetime of a VO must be guaranteed

The lifetime of a VO (as required by the applications) may range from a couple of days up to infinite (i.e. not known in advance). Therefore it must be possible to a) guarantee the lifetime of a VO for a specified amount of time and to b) guarantee the lifetime of a VO until a notification that the VO is not required anymore.

Previous number (D4.2.3): R25
Updated: no
Importance: obligatory
Implementation order: 5.2

R28: XtreemOS must allow to establish multiple VOs on the same node within specified constraints

VOs comprise nodes from different domains. An overlapping of VOs is allowed. Therefore, it must be possible that a node is contained in multiple VOs. The domain administrator, however, is allowed to restrict the maximum number of VOs to which a certain node can belong to. One special - but also very important - case is that a VO comprises only a single node. Therefore it must also be possible that XtreemOS is executed on a single node with multiple concurrent VOs established. Consider isolation mechanisms within and between VOs (cf. WP3.5)

Previous number (D4.2.3): R26
Updated: no
Importance: obligatory
Implementation order: 7.0

R29: A VO management interface has to be provided

The management of VOs must be possible by an API (Application Programming Interface), a CLI (Command Line Interface) and a GUI (Graphical User Interface). The management interface must also provide means for monitoring the VO, e.g. information on the effectiveness of the changes made on VOs. The VO management service must be transaction aware to allow a consistent resume e.g after node failure.

Previous number (D4.2.3): R27
Updated: no
Importance: obligatory
Importance for MDs: optional
Implementation order: 4.6

R30: VO management actions must be completed within a specified maximum amount of time

Various applications require that a VO can be created, changed and destroyed within a certain maximum amount of time. In some cases (SIMEON, WISS) this response time needs to be a couple of seconds or at most 10 minutes. It is therefore necessary that VO users can in advance define how fast management actions need to be performed with respect to each application (to be agreed with the VO administrator).

Previous number (D4.2.3): R28
Updated: no
Importance: obligatory
Importance for MDs: optional
Implementation order: 7.3

R31: XtreemOS has to support communication between VOs

The applications require exchange of information between VOs by means of messages (also instant messages), shared memory and data transfer.

Previous number (D4.2.3): R29
Updated: no
Importance: obligatory
Implementation order: 7.1

A.3 Checkpointing and Restart

XtreemOS should provide two ways to increase robustness: a checkpoint/restart mechanism, as defined in deliverables D2.1.1 and D2.2.1, and a replication mechanism termed virtual nodes, as defined in D3.2.1.

A checkpoint made previously is used to restart an application that failed (either by its own fault or for other reasons). This mechanism does not need any additional resources, apart from the disk or memory space required for the checkpoint. It is appropriate for applications and services that either run isolated from the environment, such as off-line computations, or can adapt to changes in the environment. The failed application is available again after a certain amount of time.

R32: XtreemOS must support automatic failure detection, checkpointing and restart

XtreemOS must provide automatic failure detection (e.g. computer went down or network broke), checkpointing and restart at the system level preferably with no need to modify the application. *Mechanisms/suggestions* Restarts should be done from the last checkpoint, unless the user specifically requests the use of an older checkpoint. Alternatively, if the application fails a few times in a row from the last checkpoint, an older checkpoint can be tried automatically.

Previous number (D4.2.3): R30
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 4.7

R33: Restart must mimic the original environment

XtreemOS must provide the illusion of the original environment considering in particular IP addresses, hostnames and PID numbers. Accordingly, a mechanism for handling virtual IP addresses, hostnames and PIDs must be provided. If the restart takes place on a different VO resource, it must be ensured that the necessary security objectives are achieved during communication.

Previous number (D4.2.3): R31
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 5.7

R34: XtreemOS must be able to notify the application of checkpointing and restart

XtreemOS has to notify the running application prior to checkpointing and interruption such that the application has the opportunity to react appropriately (e.g. store data, close open files, send messages ...). Furthermore, XtreemOS must notify a restarted application of the changed execution environment, including - but not limited to - IP addresses, hostnames and PIDs.

Previous number (D4.2.3): R32
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 4.6

R35: XtreemOS has to support various ways of checkpoint initiation

It is required that XtreemOS provides the mechanisms for creating and storing sequences of checkpoints. The checkpointing mechanism needs to be configurable to support:

- Checkpointing initiated by the application independent from the OS
- Checkpointing initiated by the application to stable storage provided by the OS
- Checkpointing initiated by the OS at application's request
- Automatic checkpointing initiated by the OS (with and without notification)

A respective API has to be provided.

Previous number (D4.2.3): R33
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 7.4

R36: Checkpointing/restart performance

Checkpointing must be fast enough that the required checkpointing frequency will not represent a significant load to the system. Restart from a saved checkpoint must take less than the time between successive checkpoints, i.e. for one checkpoint each 20 minutes, at most 20 minutes should pass from failure to the moment when the application is up and running again. *Quantification:* One application has extremely high demands: at least one checkpoint per 30 seconds. The estimated amount of working memory per node for this application is 512 MB, but the size of the data requiring checkpointing is much smaller. The checkpointing frequency must be high for online application to allow restarting quickly after failure and without loosing much data; otherwise the user experience will suffer.

Other applications demand at least one checkpoint per hour. Their memory requirements are specified as up to a few GB.

Previous number (D4.2.3): R34
Updated: no
Importance: obligatory

Importance for MDs: Not Applicable
Implementation order: 7.0

R37: Checkpointing/restart must be implemented in the kernel space

Checkpointing and restart must be implemented on OS level in the kernel space.

Previous number (D4.2.3): R35
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 6.4

R38: XtreemOS allows to customize checkpointing and restart

It is required that XtreemOS allows the applications to specify on which nodes a checkpointed application will be restarted. This is important e.g. for applications that require user interaction. Restart from last and older checkpoints must be customizable to support:

- automatic restart
- restart with user-interaction

XtreemOS must allow the user to specify how often (or when) checkpoints are created. XtreemOS must allow to activate/deactivate checkpointing and automatic restart during application runtime with no need to reboot nodes.

Previous number (D4.2.3): R36
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 7.1

R39: Information on the process state that must be saved/restored during checkpointing/restart

Checkpointing and restart must save/restore the following information on the process state (customizable by the user/application):

- Threads
- IPC
- Network communication (in particular open MPI communication)
- Open files (contents must be saved)
- Registers

- Caches

Optionally, linked libraries should also be saved.

Mechanisms/suggestions: The automatic restart with the same threads etc may use the mechanisms already implemented in Kerrighed. The involved restart can perhaps use similar mechanisms to those that will be used for normal application startup, with a flag signalizing that this is a restart. Saving contents of large open files is not realistic. If the files are not open for very long, the checkpoint can be delayed automatically for a few seconds, hoping that there will be a moment when no large file will be open. Otherwise, the applications and the OS will have to cooperate on this issue.

Handling of IPC communication at restart: Checkpoint/restart mechanisms in the kernel should be able to save and restore the IPC state, but the decision may be left to the higher level mechanism that is able to coordinate a checkpoint between all participating processes (e.g. at the mpirun level for MPI). It is only at that level that it is possible to know if the current process restart is part of a coordinated restart or is a standalone restart.

Previous number (D4.2.3): R37
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 6.5

A.4 Federation Management

R40: Number of federation nodes used

It must be possible to specify the number of federation nodes to use because the number of nodes that can be used effectively is application-specific and thus cannot be determined by the system. None of the applications uses a fixed number of nodes (apart from IMA and JOBMA, which use one node only). Figure A.7 gives an overview over the maximum number of nodes required within a federation though it must be considered that various applications could use as many nodes as they can get.

Previous number (D4.2.3): R38
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 2.3

R41: Information available to SSI scheduler

The LinuxSSI scheduler must have access (at least) to:

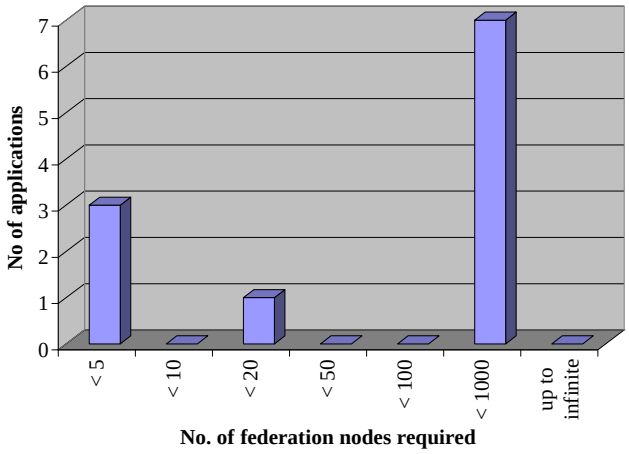


Figure A.7: Maximum number of federation nodes required by the applications

- node-level and cluster-level variables, for example per-node CPU and memory usage,
- application-specified (or process-specified) variables, for example application's requirement to be scheduled only to nodes with certain properties,
- grid-level variables, for example a requirement that two processes must never be scheduled to the same node within the cluster. Such conditions must be coordinated with the grid-level scheduler.

The above will allow the LinuxSSI scheduler to satisfy requirements R42, R43, and R46. It must also be possible to define new variables and new, custom scheduling policies that take them into account.

Previous number (D4.2.3): New
Updated: yes, new
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: To be determined

R42: Specification of service qualities in federations

It must be possible to specify service qualities (e.g. maximum network delay, availability of resources, throughput) for a certain application. The scheduler must not allocate resources that do not fulfill the defined service qualities to the application.

XtreemOS also allows applications to specify a topology of the resources which provide the best performance.

Previous number (D4.2.3): R39
Updated: yes
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 5.3

R43: Node properties constraints in federations

It must be possible to specify some required properties of federation nodes:

1. node architecture (homogeneous, heterogeneous),
2. installed libraries,
3. installed web services and other software.

Most applications depend on some libraries and other software. Furthermore, some applications would require additional effort to be able to distribute parallel execution among heterogeneous nodes. Although XtreemOS could optionally offer facilities to overcome some or all of the above difficulties automatically (which is not a subject of this requirement), such facilities would incur a performance penalty.

The XtreemOS API must provide means to specify the constraints when starting the application. The scheduler must not allocate nodes without the required properties to the application.

Previous number (D4.2.3): R40
Updated: yes
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 5.5

R44: Shared file system within a federation

XtreemOS must offer shared file system within federations. Also related to Section A.8.

Previous number (D4.2.3): R41
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 3.9

R45: Changing number of federation nodes

It must be possible to change the number of nodes that the application uses during runtime. If the number of available federation nodes changes, XtreemOS must notify the running applications. The application then decides whether it can adapt to the change.

Mechanisms/suggestions: if the application can adapt to the change, it is its responsibility to rearrange any variables and computations going on. If an application cannot adapt on-the-fly to fewer nodes being available, perhaps it can be checkpointed and restarted on fewer nodes. The notification mechanism can be decided on later.

It must also be possible that the running application requests a change of the number of federation nodes. A running application can request additional nodes to start processes. These additional resources have to be provided by XtreemOS (if nodes are available). Furthermore, a running application may release certain resources after terminating calculations on these nodes. These nodes are then available for the execution of other applications. XtreemOS must be able to dynamically consider these released nodes in resource management and to provide them to other applications.

Previous number (D4.2.3): R42
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 4.4

R46: SSI scheduling of replicated processes

For processes replicated by the virtual nodes mechanism provided by WP3.2 (see also R4, it must be possible to run multiple replicas on the same federation with the restrictions that they are never scheduled onto the same SSI node. The replicated processes are not required to be aware of being executed on a federation, and they do not need to use SSI mechanisms.

Previous number (D4.2.3): R43
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 8.3

R47: Checkpointing and restart

Automatic failure detection, checkpointing and restart must also be supported on federation nodes. The respective requirements are equivalent to those in Section A.3.

Previous number (D4.2.3): R44
Updated: no
Importance: obligatory
Importance for MDs: Not Applicable
Implementation order: 5.3

A.5 XtreemOS Interfaces

R48: Other API Standards as basis for XtreemOS API

XtreemOS API must consider the following standard as a basis: SAGA (especially the subsets DRMAA, GAT). Furthermore, any other standard allowing applications to access user and/or job information is welcome. WP3.1 must ensure that applications can adapt to XtreemOS in a way that complex start and stop procedures can be specified that are used to start/stop the various parts of the overall application. To this end, WP3.1 must ensure that the interfaces are sufficient enough and compatible with WP3.3

Additionally, one application requires the following functionalities that are actually provided by Globus/Globus-related components: GridFTP, Apache Axis, and the GSI public key infrastructure. Here, an equivalent XtreemOS functionality is needed.

Previous number (D4.2.3): R45
Updated: no
Importance: obligatory
Implementation order: 5

R49: XtreemOS must support Linux software with no need for modifications

It must be possible to install and execute Linux applications, in particular legacy applications, with no need for modifications neither to installers nor to application code. The Linux extensions and modifications produced by the XtreemOS project must be exploitable by such applications as far as possible. The fulfillment of this requirement largely leverages the acceptance of XtreemOS in the users (in particular industrial users) and developers community.

Previous number (D4.2.3): new
Updated: yes, new
Importance: obligatory
Implementation order: to be determined

R50: Demand for POSIX compliance

XtreemOS must provide access to the distributed resources in the Grid from any node using the standard Posix interface. Mandatory access control (ACL) as defined e.g. in POSIX .1e, IEEE 1003.1e/2c (which was withdrawn). It is yet to be

clarified which calls have to be provided. Furthermore, it would prove useful if functions for management of processes on remote machines would be part of the XtreemOS API.

Previous number (D4.2.3): R46
Updated: no
Importance: obligatory
Implementation order: 5.3

R51: XtreemOS API language support

XtreemOS must support several different programming languages. The mandatory languages and their priorities are C (high priority), C++ (high priority), Java (high priority) and Fortran 77 (low priority). Fortran 77 can be supported via C bindings. The optional languages are Python (medium priority), Perl (medium priority) and Ada (low priority).

Previous number (D4.2.3): R47
Updated: no
Importance: obligatory
Importance for MDs: In MDs, only Java is obligatory. The rest are optional.
Implementation order: 3.3

R52: Degree of Interoperability

It should be possible to use XtreemOS as a backend for GT4 WS-GRAM.

Previous number (D4.2.3): R48
Updated: no
Importance: optional
Implementation order: 6.4

A.6 Highly Available and Scalable Grid Services

R53: Availability of nodes for a specified time

In order to execute parallel applications on top of XtreemOS, it is necessary to provide for a high availability of allocated nodes during the whole job execution. To this end, a fault-tolerance mechanism termed virtual nodes replicating specified nodes is needed. However, this replication has to be specified explicitly by the user or application.

Previous number (D4.2.3): R49
Updated: yes
Importance: obligatory
Implementation order: 3.1

R54: High availability of bandwidth between the allocated nodes during the entire runtime of applications

For certain applications, a minimum bandwidth (user-defined) must be guaranteed to communicate efficiently between the allocated nodes. If a bandwidth guarantee cannot be given as the underlying infrastructure is not able to do that, the system must be able to estimate the available bandwidth with high precision. In this case, the goal is a stable minimum bandwidth with high probability (user-defined) during the whole application runtime.

Two types of applications require this feature :

- MPI applications that simultaneously run on several nodes exchanging messages from one to another. In this kind of application, the algorithm is usually built so that the overall process may be blocked till some messages are received.

- Applications that need a tight and permanent connection between two processes.

Previous number (D4.2.3): R50

Updated: yes

Importance: obligatory

Implementation order: 5.0

R55: Grid services which have to be provided

We gathered the priorities (1 = lowest, 10 = highest) of the services that must be implemented. In the following the average, minimum and maximum priorities are given:

Service	Avg.	Min.	Max.
Data management	8.1	1	10
Security	7.9	2	10
Authentification	7.7	5	10
Application Deployment	6.7	1	10
Job Monitoring	6.5	1	10
Resource Scheduling	5.8	1	10
Resource Allocation	5.8	1	9
Resource Monitoring	5.7	1	9
Scalability of the whole grid	5.6	1	9
Resource Exploration	5.2	1	8
Decentralization	5.3	1	10
Network Allocation	5.0	1	10
Network Monitoring	4.9	1	10
SLAs	4.9	1	10
Grid services for mobile devices	4.7	1	8
Services to manage Virtual Organization	4.6	1	11
Migration Services	4.0	1	10

Remarks:

- Priorities are not a scale of importance. The average priorities may be used as a guidance in which order the services could be developed. However, a low average priority may not reflect that some applications expressed a high priority for a certain service, e.g., GSDNA, WEBAS and WISS requiring migration services with high priority. Therefore, the respective minimum and maximum values are also given.

- All the applications expressed a high priority on the authentication and security.

- The data management and deployment issues are also vital for all the applications except IMA and JOBMA. GALEB is also less impacted by the data management needs.

- The priority of every single issue has been evaluated higher than 8 by at least three applications.

In conclusion, we can say that every feature and the corresponding interfaces seem equally important. At the beginning of the project, a particular effort could be made on authentication and security.

Previous number (D4.2.3): R51
Updated: yes
Importance: obligatory
Implementation order: 4.8

R56: Measurement criteria which must be provided to estimate the quality of Grid services

We gathered the priorities (1 = lowest, 10 = highest) of measurements which must be provided to estimate the quality of the Grid services. In the following the average, minimum and maximum priorities are given:

QoS Measurement	Avg.	Min.	Max.
Number of nodes failed during runtime	6.8	1	10
Ability to reserve resources in advance	6.2	1	10
Number of timeouts waiting for data	5.9	1	10
Time for Authorisation	5.3	2	10
Time to connect Mobile devices	4.0	1	10
Number of not satisfied SLAs	3.8	1	10
Time for Job-Migration	3.4	1	5
Time for Deployment	3.2	1	8
Time for Negotiation	3.0	1	10
Time for Scheduling	2.6	1	5
Time to set up Virtual Organizations	2.6	1	8
Time for Exploring appropriate resources	2.3	1	4

Remarks:

- Priorities are not a scale of importance. The average priorities may be used as a guidance in which order the measurements could be developed. However, a low average priority may not reflect that some applications expressed a high priority for a certain measurement. Therefore, the respective minimum and maximum values are also given.
- Grid services must be established that enable applications and users to gather those information.
- Additional QoS measurements which have to be provided include the time for exploring Grid users (needed by IMA with priority 8) and the time for collecting job information (needed by JOBMA with priority 10).

Previous number (D4.2.3): R52
Updated: yes
Importance: obligatory
Implementation order: 5.3

A.7 Application Execution Management

R57: Monitoring System

The monitoring system must provide information about running applications via a monitoring interface or tool and via a notification service.

Monitoring: Provides the means for monitoring the resource consumption during the whole application execution.

Notification service: It must be possible to notify running applications and users whenever predefined conditions are fulfilled.

An interface has to be provided such that an application or the user can configure the monitoring parameters including:

- The **kind of information** and the level of detail at which this information has to be provided (cf. R53, R54)

- The **means and conditions for notifications:** Half of the applications require a notification by email and the other half of the applications needs a callback function. Therefore both means of notification have to be provided. As discussed with WP3.3, the email notification might affect the performance of the overall system. However, it is part of the research to evaluate such effects at the end of the project. The conditions for asserting a notification comprise changes in the application, environment and about situations when certain stages of the overall application execution phase have been reached.

- Several applications require notifications whenever a job status changes (job started, job finished), a job is migrated (migration started, migration finished), or a certain job threshold is exceeded (job surpassed memory threshold, job surpassed storage threshold). Also refer to R56 and R57. Note that the different applications do not request a global job queue but require the functionality that XtreemOS behaves as if it would have one.

Previous number (D4.2.3): R53
Updated: no
Importance: obligatory
Implementation order: 3.7

R58: Job Dependencies

XtreemOS must provide the possibility to specify dependencies between jobs. Thus, whenever a job is submitted to the system, the user must have the possibility to specify jobs that the new job depends on. These dependencies are used for a comprehensive accounting, monitoring, and for the killing of dependent job sets.

Previous number (D4.2.3): R54
Updated: no
Importance: obligatory
Implementation order: 3

R59: Resource Accounting

It must be possible to record the usage of a resources (supported by the kernel) by a user at any given time. A way to implement / use special cost models must also be provided. Furthermore, the accounting has to provide details at what time or within which time interval a certain amount of resources were used. Thus, the final billing must be able to modify cost models according to different times and overall system states, e.g. a computation during the night is normally cheaper than during the day. Note, that it must be possible to automate the resource accounting for jobs using the job dependencies. Thus, it must be possible to ask for the accounting of a specific job and all its dependent jobs.

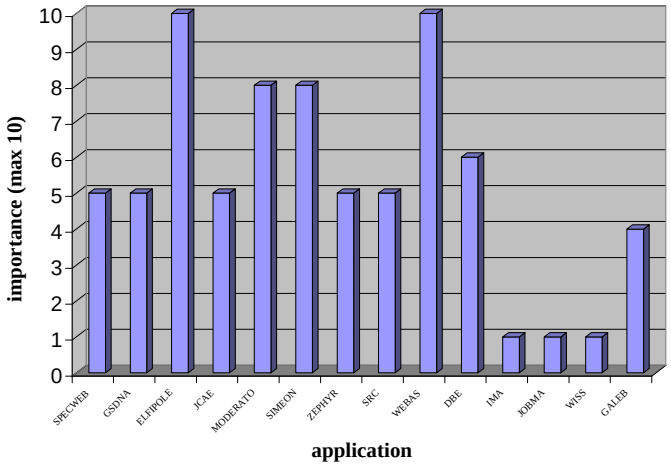


Figure A.8: Rating for resource accounting

Previous number (D4.2.3): R55
Updated: no
Importance: obligatory
Implementation order: 5

R60: The kind of monitoring data that the applications need to query

The monitoring system must provide the following monitoring data to several applications (including dependencies):

- **Job execution:** number of waiting jobs, number of running jobs, number of jobs being migrated, expected time before a job is started, number of jobs earlier in the queue (again, XtreemOS must only provide the functionality as

if a global job queue would exist. The applications do not require a global job queue directly.)

- **Job's resource usage:** job CPU usage (in %), total CPU time, memory usage (current/peak/page faults), I/O (bytes read, bytes written, number of reads/writes), threads (number, priorities, start time of each, current/total CPU usage of each thread), number of files currently opened, name of files that are currently opened, file system space currently allocated, number of bytes transferred through the network
- **Job performance metrics:** job's effective file throughput, job's effective network throughput
- **Hardware performance metrics:** job's cache hit rate, job's cache miss rate, job's effective timeslice, jobs priority
- **Host load metrics:** host CPU load, host network load, host file system load

Furthermore, the monitoring system should provide:

- **Host load metrics:** host file system load

Previous number (D4.2.3): R56
Updated: no
Importance: obligatory
Implementation order: 3.7

R61: The kind of information that can be queried from the command line while the application is running

Several of the applications require to gather the following information from the command line during the runtime of the application (including dependencies):

- **Application execution:** number of jobs that an application has submitted, number of jobs that have finished, number of jobs that are running, number of jobs that are waiting (again, XtreemOS must only provide the functionality as if a global job queue would exist. The applications do not require a global job queue directly.)
- **Job execution:** which host is each job assigned to, expected time before a job is started, number of jobs earlier in the queue
- **Job's resource usage:** job CPU usage (in %), total CPU time, memory usage (current/peak/page faults), I/O (bytes read, bytes written, number of reads/writes), threads (number, priorities, start time of each, current/total CPU usage of each thread), number of files currently opened, which files are currently opened, file system space currently allocated, number of bytes transferred through the network

- **Job performance metrics:** job's effective file throughput, job's effective network throughput

- **Host load metrics:** host CPU load, host network load, host file system load

Previous number (D4.2.3): R57
Updated: no
Importance: obligatory
Implementation order: 3.4

R62: Tracing System

Tracing must be provided for both the application execution and the resources being used. Different levels of details are required for both of these, depending on the application, a mechanism should be in place to set these. This requirement only describes the general demand. The specific demand is detailed in the next requirement. Again, the dependencies between jobs must be incorporated.

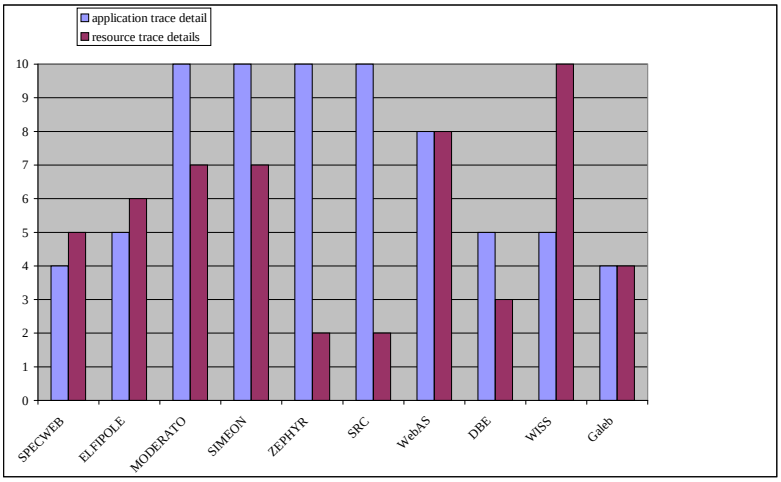


Figure A.9: Rating tracing of application execution vs. tracing of resource

Previous number (D4.2.3): R58
Updated: no
Importance: obligatory
Implementation order: 4.2

R63: Kind of events required to trace from the application's execution

Kind of events required to trace from the application's execution The system must trace the following events. Note that it is not expected that WP3.3 provides all of these functionalities alone but needs to get support from other work packages.

- **Grid scheduling events:** job submitted, job started, job finished, job started migration, job finished migration
- **Job's kernel level events:** system calls, page misses, preemption, timeslice exhaustion, other blocking conditions
- **Resource events:** enqueue job, start job, finish job
- **Others:** application level events triggered using an API designed for that purpose, aggregated information on all/specific events (hardware performance metrics, ...)

Furthermore, the system should trace the event:

- **Communications/parallel library events:** MPI primitives (send, receive, broadcast and variants), OpenMP primitives (enter and exit parallel regions, etc). As the MPI libraries are not directly part of the project, a sophisticated mechanism to add events to the system might be sufficient.

Previous number (D4.2.3): R59
Updated: no
Importance: obligatory
Implementation order: 4.2

R64: Kind of state changes that must be traced from the application's execution

The system must trace the following changes. Note that it is not expected that WP3.3 provides all of these functionalities alone but needs to get support from other work packages.

- **Grid scheduling states:** job waiting, job running, job finished, job migrating
- **Kernel level states:** thread blocked, thread running user space, thread running kernel space
- **Communications/parallel library events:** waiting data, sending data, waiting on synchronization point (barriers)
- **Resource events:** CPU running, CPU idle

Previous number (D4.2.3): R60
Updated: no
Importance: obligatory
Implementation order: 4.3

R65: Trace Format

XtreemOS must provide support for the Paraver format and can additionally develop an XtreemOS-specific trace format.

Previous number (D4.2.3): R61
Updated: no
Importance: obligatory
Implementation order: 5.7

R66: Scheduling

A co-allocation of application on resources of several different sites must be possible. The response times of certain applications of specific customers should be privileged while avoiding starvation of other jobs.

Previous number (D4.2.3): R62
Updated: no
Importance: obligatory
Implementation order: 5.4

R67: Resource Planning

Reservation of resources for specific intervals is necessary and also being able to specify certain characteristics of the required resources (i.e. CPU speed and load, disk space, memory) and to define further constraints regarding the execution of applications (e.g. penalties for late execution, restrictions regarding the choice of nodes, etc.). As XtreemOS is a distributed system, the individual users need information with a minimum accuracy. This accuracy must be defined for individual requests. Furthermore, some applications require changing of resource requirements during runtime. Applications will need detailed information to plan resources in advance and they should be able to confirm resources prior to allocation. Applications need to execute some parts in parallel on different resources (up to 1000 parts). Furthermore, XtreemOS must be able to run a workflow engine on top. Thus, WP3.3 must prove by an example that such a workflow engine can be used to specify startup and stop sequences of whole applications consisting of individual dependent jobs. Furthermore, WP3.3 must ensure that the SAGA interfaces generated in WP3.1 are sufficient to fulfill these requirements. Otherwise, WP3.3 is requested to provide the missing functionality.

Previous number (D4.2.3): R63

Updated: no
Importance: obligatory
Implementation order: 5.4

R68: Stopping execution

It should be possible to stop the execution of an individual application as well as the execution of a more complex application that consists of several jobs. These job interrelationships are described using the job dependencies.

Previous number (D4.2.3): R64
Updated: no
Importance: obligatory
Implementation order: 3.2

R69: Changing owner permissions during Application Runtime

It must be possible to modify the owner permissions during the application runtime. To this end, the User ID and the corresponding credentials must be modified. This requirement must be discussed with WP 3.5. Furthermore, the role of the administrator as the only actor that is allowed to initiate such a change must be discussed.

Previous number (D4.2.3): R65
Updated: no
Importance: obligatory
Implementation order: 8.3

R70: Spawn jobs to other VOs

An application running on a virtual organization can spawn a job to another virtual organization.

Previous number (D4.2.3): R66
Updated: no
Importance: obligatory
Implementation order: 7.9

R71: Manual exploration and selection of hardware

The application needs to be able to select the resources it uses.

Previous number (D4.2.3): R67
Updated: no
Importance: obligatory
Implementation order: 5.1

R72: Co-Allocation

Certain applications need co-allocation of resources of several different sites. It must be possible to prohibit the co-allocation on the application level, e.g. for security reasons.

Previous number (D4.2.3): R68
Updated: no
Importance: obligatory
Implementation order: 4

R73: Distribution

An application must be able to limit its geographical distribution. This includes simple distance measurements between resources as well as the specification of continents and countries. The restriction to execute certain jobs only in certain countries is based on differing laws in different countries and the corresponding security demand of the individual applications. Thus, the system must know where the different resources are located (countries etc.).

Previous number (D4.2.3): R69
Updated: no
Importance: obligatory
Implementation order: 6.9

A.8 Data Management

This section starts with an overview of data characteristics of the applications visualized in figures A.10-A.16. These characteristics should be taken into consideration when designing the data management facilities. Note that a lot of questions in the requirements questionnaire only apply to the 12 file-based applications (cf. Figure A.10). The application that uses a database uses one database whose size ranges from ca. 16 GB to 6 TB.

R74: Concurrent access to open files

It must be possible to concurrently read from files which are open for write access. It must also be possible to concurrently write to files open for reading. The consistency requirements and respective mechanisms remain to be discussed.

Previous number (D4.2.3): R70
Updated: no
Importance: obligatory
Implementation order: 3.7

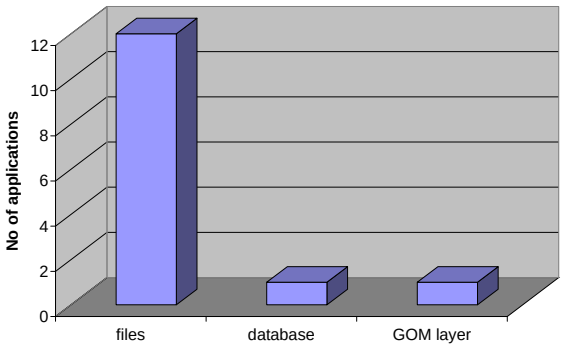


Figure A.10: Files, data bases, Global Object Management (GOM) layer

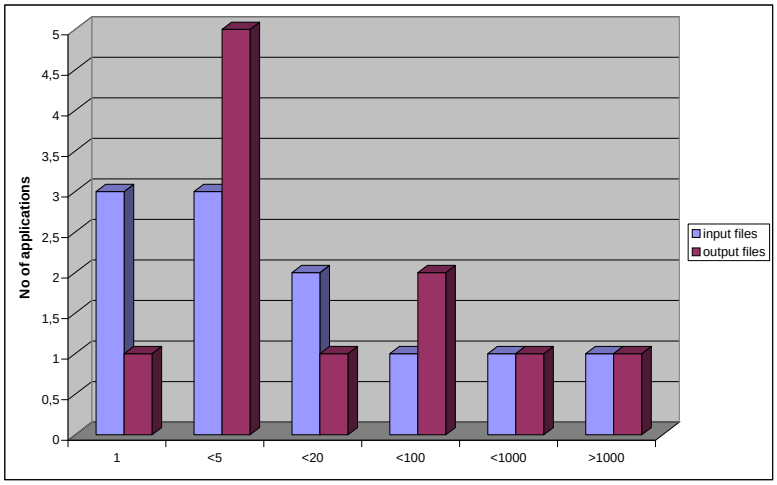


Figure A.11: Number of input and output files

R75: Required Meta Data

The usual UNIX metadata must be accessible, or at the minimum the following: the full path, global Grid user name of the owner, global virtual organization identifier (VO ID) of the owner's VO where the file has be opened for read/write, and the time of last change.

Previous number (D4.2.3): R71
Updated: no
Importance: obligatory

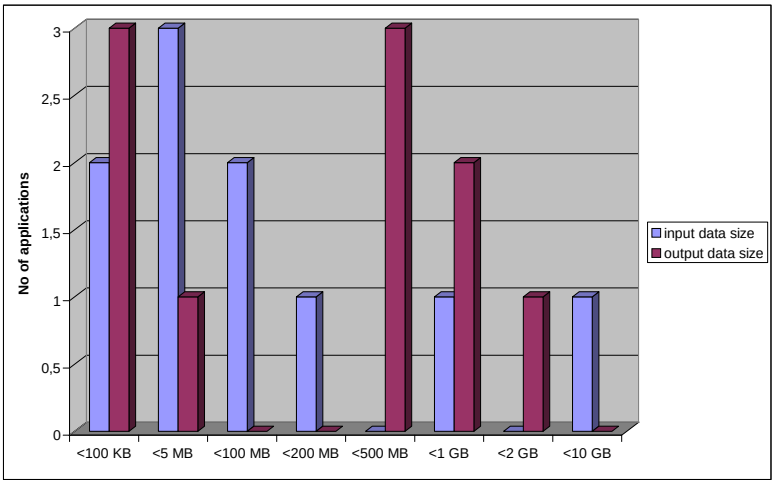


Figure A.12: Estimation of the size of input and output data in MB

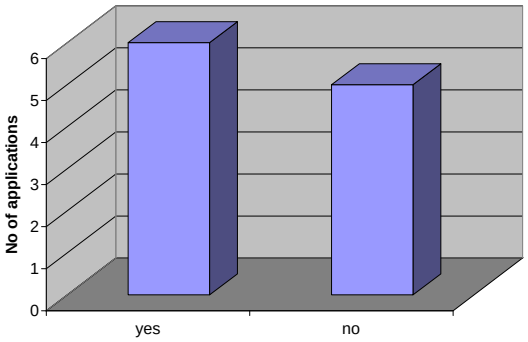


Figure A.13: Can all input and output files be specified at initialization?

Implementation order: 4.1

R76: Directories

Application defined directories are required. The applications must be able to define and use directory structures, which are then used to organize the files.

Previous number (D4.2.3): R72
Updated: no
Importance: obligatory

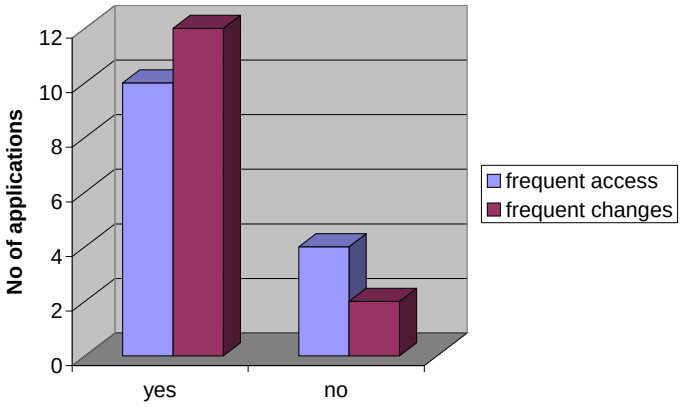


Figure A.14: Are applications accessing the corresponding data frequently?

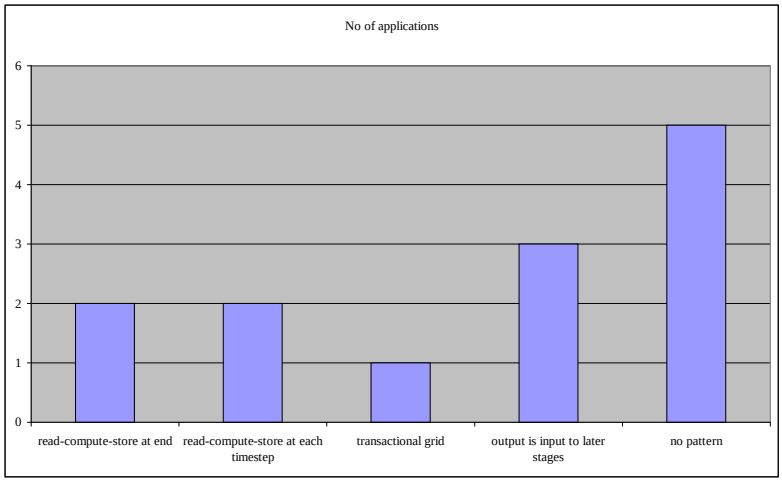


Figure A.15: Data workflow and data access patterns

Implementation order: 4.5

R77: File metadata access control granularity

It must be possible to set the access rights on parts of individual metadata.

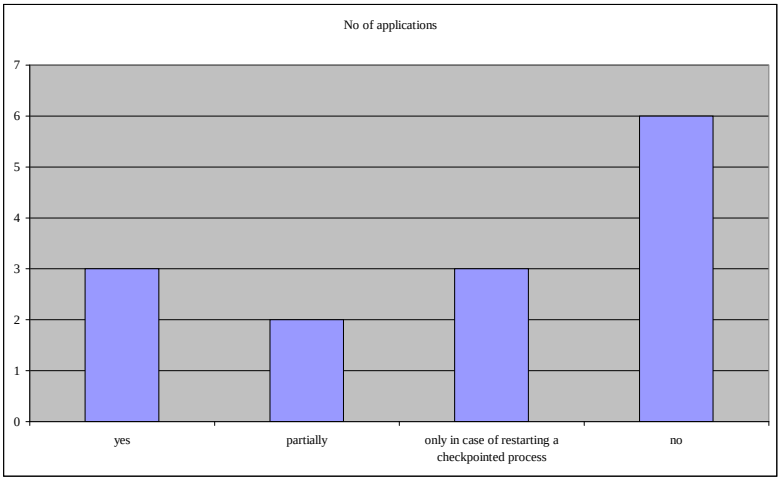


Figure A.16: Pre-fetching of application data before job execution

Previous number (D4.2.3): R73
Updated: no
Importance: obligatory
Implementation order: 8

R78: File data and metadata change monitoring/notification

It should be possible to check for file data and metadata changes. It should also be possible to subscribe to the information on file data and metadata changes, e.g. through registering a callback function.

Previous number (D4.2.3): R74
Updated: no
Importance: optional
Implementation order: 7.4

R79: Data access time

File access time must be below 10 s to prevent time-out errors in applications. The data access time for interactive applications using a database must be below 150 ms. Furthermore, data access is very frequent.

Previous number (D4.2.3): R75
Updated: no
Importance: obligatory
Implementation order: 4.3

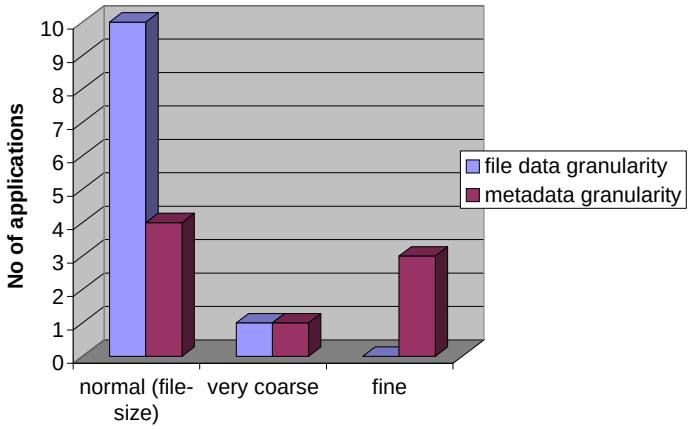


Figure A.17: Granularity of metadata access control required

R80: Replica limitation

It must be possible to limit the number of replicas. Some applications need this possibility because of copyright, storage space, security, and performance constraints.

Previous number (D4.2.3): R76
Updated: no
Importance: obligatory
Implementation order: 6.5

R81: Data versioning

The versioning of data to allow incremental changes is required.

Previous number (D4.2.3): R77
Updated: no
Importance: obligatory
Implementation order: 8.9

R82: Explicit move/copy between resources

It must be possible to explicitly copy/move data between different resources.

Previous number (D4.2.3): R78
Updated: no
Importance: obligatory
Implementation order: 4.7

R83: GOM object layer

The GOM layer must support an object based access/sharing. The GOM layer must support transactional consistency for object sharing. This includes re-startable transaction combined with optimistic synchronization.

Previous number (D4.2.3): R79
Updated: no
Importance: obligatory
Implementation order: 6

R84: Data transmission monitoring

A facility to monitor ongoing data transmission must be provided.

Previous number (D4.2.3): R80
Updated: no
Importance: obligatory
Implementation order: 3

A.9 Security in Virtual Organizations

This section consolidates the requirements for work-package 3.5 on security services for the Grid-OS. Note that the responses to the questions on security services were rather sparse, indicating that either the questions were too vague, or that it is difficult for application providers to effectively make judgments concerning their security requirements. We therefore need to have some agreement on architecture and use cases as soon as possible. The goals of the requirements analysis were to:

1. Discover and prioritize particular threats that we need to address with respect to confidentiality, integrity, availability, accountability and isolation. Note that in WP 3.2 there is a 7.3 prioritization of security services, indicating that participants recognize security as a high priority capability of the Grid and a Grid-OS.

2. Identify the set of security policies that need to be specified and their expressiveness. Note that many participants did not answer questions regarding policies, which suggests that work on policy languages should be treated as a minimal priority.

3. Determine the mechanisms required in order to enforce security policies and the effort associated with developing and integrating them as services in the OS. One comment made by one participant is that we should also include the middleware-level in the assumption of mechanisms, but we should however focus on what needs to be integrated in the OS itself.

4. Derive the requirements for administration support for OS security services, such that we need to consider what new mechanisms would introduce

5. Determine effective means of evaluating the security services once developed, integrated and deployed in a Grid environment

6. Determine if preliminary tests need to be carried out before committing to explicit requirements. This final aim of the requirements analysis is particularly important, as it was not possible for all participants in the questionnaire to commit to answers for every question. Again this can be attributed to either insufficiency in precision of the questions or the inherent difficulty of being qualitative or quantitative where security is concerned.

The requirements have been grouped according to security control objectives that have been determined according to a generic architecture for possible distribution and administration of resources in a Grid Environment, as depicted in figure 3.5.1. This is also consistent with the proposals of the Grid Architecture in [7]. This will be referenced within each requirement specification by way of clarification.

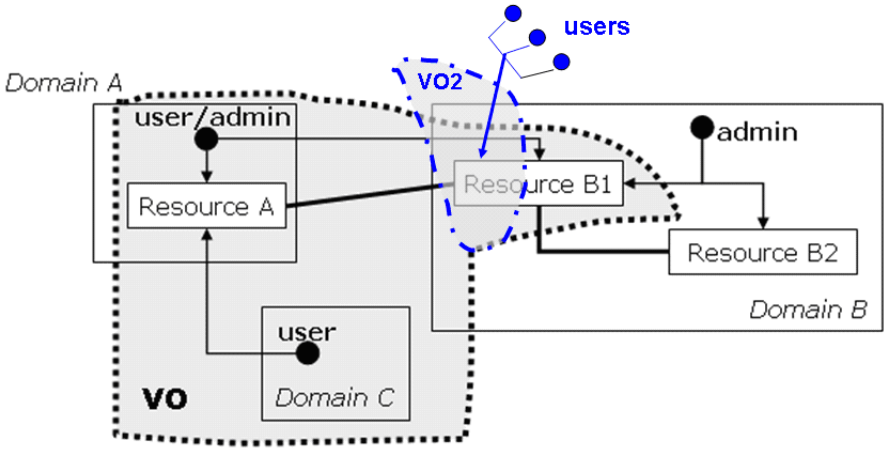


Figure A.18: A generic architecture for distribution, usage and administration of resources in a Grid environment

The possibility of covert channels to data and resources cannot be ignored, as it is possible for a malicious party to bypass the messaging infrastructure if covert channels or alternative mediums for accessing resources are established. Furthermore, the assurance of the administrative processes associated with computational nodes impacts on the way in which security requirements can be enforced. A goal of the security requirements is therefore to determine what additional information is required in messages exchanged between computational nodes, and hence what

represents a “correct” message before it can be processed by an application or security service. Finally, as we are aiming towards the development and integration of security mechanisms as services in the OS-layer, security becomes a recursive requirement, where the question of how to secure security services in a Grid OS also arises.

The overall question we ask is therefore, if XtreemOS is installed on all resources (A, B1, B2), can we improve the way the following objectives are met?

1. Confidentiality of stored data
2. Confidentiality of data being communicated between end points
3. Integrity of stored data
4. Integrity of communicated data
5. Identification and authentication of users
6. Authorized access to application services
7. Guaranteed access to application services by authorized parties
8. Accountability of data access and service execution
9. Isolation of data per-VO
10. Isolation of services per-VO

R85: Data stored on resources must only be accessible by users and administrators that are members of a VO with the appropriate read access rights *(objective 1)*

Confidentiality is a fundamental requirement of systems that store, process and exchange sensitive data and information. In a Grid-enabled system the requirement for confidentiality of stored data is to ensure that data can only be accessed and read by services, users and administrators (together known as Principals) that have a "need" to read the data. A principal has a need if the following conditions are true: owner of the data OR (registered as a member of a VO with rights to the data AND assigned to a task that requires access to the data). A principal with such properties is referred to as a "valid principal" otherwise we refer to the principal as an "invalid principal". For example, the user of Domain C can only have access to data on Resource A, may have access to data on resource B1 but no access to data stored on resource B2. A local administrator who belongs to more than one VO at the same time should only be able to act according to a policy or a contract between the VOs he is a member in. This requires a solution where dedicated resource monitoring (e.g. file access) are relying on other roots of trust (e.g. hardware trust anchors) than the local administrator.

Based on the questionnaire, 50% of the participants indicated that confidentiality of stored data is a highly critical requirement. It must be possible to enforce that only owners of data, members of VOs and parties assigned tasks (as roles, rights or responsibilities) in the VO can read data stored on resources (i.e. computational nodes). A difficult requirement is to stop the administration and users of say domain B from accessing data stored on resource B1. 25% of the participants however did not answer, which is perhaps because of a difficulty with understanding how to measure the criticality of security. One potential requirement for the overall framework is therefore to provide some comprehensive metrics that can be applied for future evaluation of the security needs of applications being deployed in a Grid OS. 2 participants (12.5%) did however explicitly state that the confidentiality of stored data was not critical for their application, highlighting the flexibility that must be enabled should a shared Grid OS be used for multiple applications.

Previous number (D4.2.3): R81
Updated: no
Importance: obligatory
Implementation order: 3.9

R86: Confidential data communicated between resources in the same VO must be transported via confidential channels associated with the VO resources. *(objective 2)*

It is assumed that data is transmitted over secure, e.g internal networks but also over insecure channels such as the Internet. There is a need for mechanisms that protects messages and responses in transit between computational nodes over insecure channels. More specifically, confidentiality of data is concerned with the protection of message inputs and the corresponding outputs of responses from invalid observers. An invalid observer has similar properties as that of an invalid principal of stored data in R3.5.1. However, the security policies and mechanisms are now concerned with the properties of the channels over which messages and responses are transmitted. In order to not transmit confidential by an insecure channel, all the data for which a user or application owner can not adjust security preferences (e.g. IPC, process migration, ?) are secured by a cryptographic scheme and protocol by default.

Users and application owners should specify individual security preferences for their communication.

Again 50% indicated that confidentiality of transmitted data was highly critical, assumedly with the same reasoning as by stored data. However, fewer participants were clear about the perceived difficult of preventing and detecting breeches to this requirement. Especially at audit time only 25responded, with one participant stating high and another stating low difficulty, whereas all but one of the respondents indicated that there is a high difficulty associated with detecting runtime breaches. A typical rule-of-thumb is to aim for prevention before detection,

such that it should first be endeavored to restrict illegal principals from reading data in transit.

Previous number (D4.2.3): R82
Updated: no
Importance: obligatory
Implementation order: 3.1

R87: Loss of integrity of stored data must be preventable and detectable using hash mechanisms *(objective 3)*

Storage integrity is typically stated as the ability to prevent illegal changes to data. As data in a VO may be sourced from different participants, who may not be "owners" of the data, it must be possible to validate that the data has not been altered by illegal parties. Data should be hashed and digitally signed by a trusted key stored on the operating system. Again a legal party must be a member of a VO and have the appropriate rights to make changes to data.

Previous number (D4.2.3): R83
Updated: no
Importance: obligatory
Implementation order: 5.1

R88: The integrity of data transferred between resources or received from users must be validated before being committed *(objective 4)*

There is then a need for an OS reference monitor mechanism to capture and validate all incoming and outgoing network traffic. The integrity of communicated data is concerned with ensuring that illegal change is not possible to data in transit. This differs from data in storage as the properties of communication channels tend to be more dynamic, based on the location, operating system and medium used by end point nodes. The operating system must therefore be capable of signing and verifying signatures of data in an end-to-end manner. The term "committed" suggests that a transaction framework is necessary, considering the distributed nature of the resources.

Previous number (D4.2.3): R84
Updated: no
Importance: obligatory
Implementation order: 4.9

R89: It should be possible for a user to use a single method of authentication (i.e. single sign-on) to gain authorized access to resources in a VO *(objective 5)*

Identification and authentication are again fundamental requirements for security, as integrity and confidentiality are difficult without the capability to identify and

authenticate principals. Identification ensures that different principals (e.g. a source or receiver of a message) are repeatedly distinguishable from each other, while authentication associates attributes used to identify a principal with a unique root attribute such as a legal name or public key. It must be possible for all resources in a VO to identify and authenticate users requesting access to data. Users of resources should not have to be bothered with changing the way they interact due to changes in the hosting of the resource. One example is cross-domain single-sign-on (SSO), which requires an agreement of how tickets and attributes are encoded and verified, which asserts that users have been authenticated and possess the appropriate authorizations to perform actions in the VO.

Network access via socket interfaces and IPC can also be treated as resources but wasn't covered by the questionnaire. There are currently no widely used monitoring frameworks. This is of upcoming importance since these resources can become scarce.

Previous number (D4.2.3): R85
Updated: no
Importance: optional
Implementation order: 4.8

R90: It must be possible to transfer and validate authorizations to virtualized resources when the host is changed *(objective 6,7)*

Authorization requirements precede confidentiality and integrity requirements and depend on identification and authentication. It was therefore not surprising that this was the requirement indicated by most (62.5%) of the participants as highly critical. Authorization is the requirement that principals can only access data that they are authorized to use in order to perform tasks, or, in the case of multilevel security systems, that they have the requisite clearance in the system. The indication of a principal's rights to perform a task is usually indicated using a token, ticket or credential, which are different forms of associating an identity with a specific right. This follows from R3.5.5, as the authorizations should also be consistent across the domains providing resources. However, the challenge is still making sure that the local administrators of hosts do not have to breach the private policies enforced by their operating systems.

The functionality to this requirement must be available all the time in order to not interfere with the quality of service. In the case of a centralized management high availability or failover measures have to be used.

Previous number (D4.2.3): R86
Updated: no
Importance: obligatory
Implementation order: 5.4

R91: It must be possible to validate membership in VOs and ensure access to resources given proven membership and rights *(objective 5,6,7)*

Authorization and guaranteed access are two different requirements although enforced by interdependent security mechanisms/ services. That is, a principal may have been provided with a token, ticket or credential but the appropriate access control policy or service interface is not available at the time of request. The locally evaluated rules that determine if a party is authorized or not (beyond the possession of a token, ticket or credential), must also be agreed to across the set of resource providers. It may not be possible to implement this in the OS, but there need to be "hooks" to higher level services that can perform such evaluations.

Previous number (D4.2.3): R87
Updated: no
Importance: obligatory
Implementation order: 4.4

R92: It must be possible for administrators to record usage (by whom and when) of resources without users being able to deny (repudiate) usage *(objective 8)*

Accountability is the ability to enforce and prove that a principal has performed an action on a given resource at a given time. The requirements for accountability are typically a secure audit service with the ability to timestamp messages. This is for the purposes of non-repudiation, should there be a case where it must be proven that a principal has indeed performed an action, as well as billing. Should also be possible to record within which VO the resource was used.

Previous number (D4.2.3): R88
Updated: no
Importance: optional
Implementation order: 6.3

R93: Isolation of VO users - It must be possible to maintain users for different VOs separately *(objectives 9, 10)*

As users may be involved in multiple VOs, it is then necessary to separate their user data and have a means of determining for which VOs are they currently working in, when accessing data.

Previous number (D4.2.3): R89
Updated: no
Importance: obligatory
Implementation order: 5.6

R94: Isolation of data per-VO: Data of different VOs, hosted on the same physical resource must show non-interference *(objective 9)*

The isolation of data per-VO enables data to be separated between different groups and contracts. Data belonging to a VO should be logically isolated only for that VO, such that changes made in one VO, although referring to the same data element, should not be. A local administrator who can belong to more than one VO at the same time should only be able to act according to a policy or a contract between different these VOs. There is an absolute need for having automated enforcement of policies. This is however not that surprising, as confidentiality is a fundamental security requirement and most organizations with sensitive data would have already invested time and money in acquiring, developing and integrating mechanisms to enforce confidentiality policies. Most participants in the questionnaire indicated that they do have utilities integrated with their applications that already meet the basic confidentiality requirements. However, there is a mix of implementation dependent on the OS-Layer and integrating at Application Layer, which suggests that there still needs to be a consolidating framework that allows reuse of OS security services as well as application layer libraries and security modules. A multi-layered architecture for security services is therefore foreseen, which however means that the integration points between layers must also be analyzed and secured. The deferral of the enforcement of confidentiality policies to third parties was not accepted amongst the participants, such that owners of resources and data must be able to maintain control of who accesses their data, even if the data is stored remotely in a different domain. This has implications for the administration of policies, resources and security services.

Previous number (D4.2.3): R90
Updated: no
Importance: obligatory
Implementation order: 5.2

R95: Isolation of services per-VO: secure access to virtualized resources and services must be customized for each VO *(objective 10)*

In addition to isolation of data, services must also be isolated per VO. That means that each VO can try to achieve a different set of security objectives and information flow policies. This may also apply to a set of instantiations of a VO (VOs that are set up in an automatic way by, e.g. by using template).

It must not be possible for parties in different VOs to recognize that they are sharing resources nor to gain knowledge of what other parties are doing with those resources. If one of two virtualized services to the same physical resource fails, this should not interfere with the other.

Previous number (D4.2.3): R91
Updated: no
Importance: obligatory

Implementation order: 5.4

R96: The reuse and realization of established security standards and utilities is suggested *(all objectives)*

It must be possible to reuse and realize established standards for authentication and authorization in the OS ? e.g. PKI (public key infrastructure), PAM (pluggable authentication modules) and SSH (Secure Shell)

Previous number (D4.2.3): R92
Updated: no
Importance: optional
Importance for MDs: optional: similar or compatible standards will be used for MDs
Implementation order: 1.9

R97: Linking of Trust Management services with OS mechanisms *(objectives 3,4,5)*

There is a need to link mechanisms implemented at the OS layer to higher level reputation and third party trust management services, which influence access control decisions. Decision of the OS rely securely on third party information but are enforced in the kernel, e.g. pluggable reference monitors. Availability of this information must be ensured.

Previous number (D4.2.3): R93
Updated: no
Importance: obligatory
Implementation order: 6.0

R98: Semi-automation of administration and configuration of security infrastructure is necessary *(all objectives)*

It must be possible to set up and configure an XtreemOS security infrastructure in less than 1 working day, and, in worse case, less than 10

Previous number (D4.2.3): R94
Updated: no
Importance: obligatory
Implementation order: 5.0

R99: Semi-automation of adaptation and reconfiguration of the security infrastructure is necessary *(all objectives)*

It must be possible to make adaptations to the infrastructure in less than 1 working day and, in worse case, less than 5. This therefore implies a high degree of automation or a very simple set of guideline for flexible modifications to the infrastructure

Previous number (D4.2.3): R95
Updated: no
Importance: obligatory
Implementation order: 4.0

R100: Multiple bundles or configuration for XtreemOS crypto have to be considered to support different CPU performance constraints. The minimal set of resources and properties (e.g. CPU performance, memory) can be specified per VO resource. *(all objectives)***.**

In some cases partners have indicated that they expect a solution that consumes <0.5% of the CPU, while others have indicated as much as <50%.

Previous number (D4.2.3): R96
Updated: no
Importance: obligatory
Implementation order: 7.2

R101: A standard security assessment criteria and profile should be followed for evaluating the XtreemOS *(all objectives)*

Either we will need to extend existing metrics and evaluation criteria for security architectures, or we can adopt one such as Common Criteria, and first define a Protection Profile according to their specification

Previous number (D4.2.3): R97
Updated: no
Importance: obligatory
Implementation order: 5.2

A.10 Support for Mobile Devices

Note that the implementation orders for requirements R94 to R99 were rated by less than 5% of the applications. Therefore the results may not be representative.

R102: Hardware restrictions: XtreemOS for MD must support ARM architecture for PDAs and mobile phones

Three applications (21%) point out hardware restrictions concerning mobile devices. One of them considers PDA's and Mobile phones to be inappropriate for their requirements, while the other two specifically require ARM processors.

Previous number (D4.2.3): R98
Updated: no
Importance: obligatory
Implementation order: 5.0

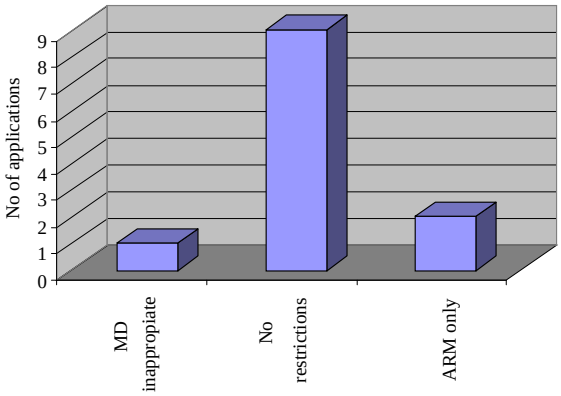


Figure A.19: Typical executing environment

R103: XtreemOS for MD must support Java

All applications which pointed out software restrictions (21%) needed some kind of Java support. Anyway, they will only need Java for monitoring, managing and instant messaging purposes. This means that support for a certain version of Java over ARM architecture will be needed.

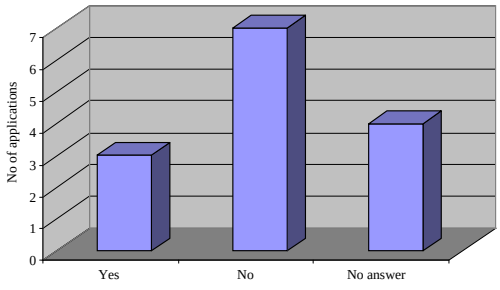


Figure A.20: Java support

Previous number (D4.2.3): R99
Updated: no
Importance: obligatory
Implementation order: 5.3

R104: XtreemOS for MD must support some basic web services protocol stack

One of the applications needs support for a basic web services protocol stack that allows the MD to function as a client to web services.

Previous number (D4.2.3): R100
Updated: no
Importance: obligatory
Implementation order: 5.5

R105: XtreemOS for MD should allow VO management

One application would benefit from using MDs for managing VOs. Only authorized users shall be able to manage VOs from MDs.

Previous number (D4.2.3): R101
Updated: no
Importance: optional
Implementation order: 8.4

R106: MDs should be considered as special nodes

More than 35% of the applications want to identify MDs as special nodes. Due to their limited processing and storage capacity is not expected to execute compute/storage intensive applications on MDs. MDs will be mainly used for monitoring and managing purposes, instant messaging and transaction performance. This means that neither their small processing capacity nor their small storage capacity will be considered an additional resource by applications.

This low capacity together with the fact that MDs aren't permanently connected, makes interesting to mark them as special nodes warning XtreemOS from scheduling a job on them.

Previous number (D4.2.3): R102
Updated: no
Importance: obligatory
Implementation order: 4.2

R107: XtreemOS for MD must provide communication and job monitoring and management facilities

50% of applications benefit from the use of a MD as a node in the Grid: most of them for managing and monitoring purposes, and only one application for performing transactions. Another application needs MDs to exchange instant messages with other users, but this can be achieved by providing a standard socket interface.

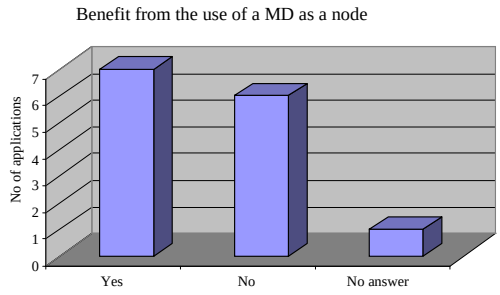


Figure A.21: Expected benefit from using MDs as a node

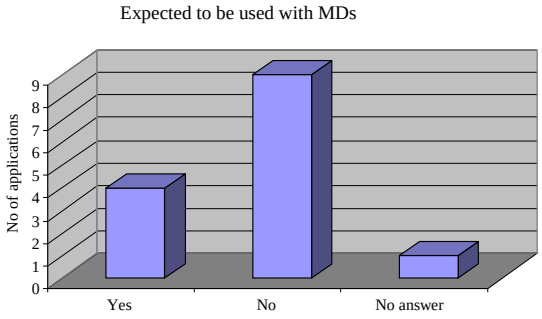


Figure A.22: No. of applications expecting to use MDs

Thus, apart from being marked as special nodes (previous requirement), basic services offered by XtreemOS for mobile devices have to be job management (launch, stop, resume, cancel, result view) and monitoring.

Previous number (D4.2.3): R103
Updated: no
Importance: obligatory
Implementation order: 6.5

R108: XtreemOS for MD should also support lightweight security methods to improve MDs' performance

42% applications allow MDs to use lightweight security methods (e.g. shorter keys to cypher communications) to improve their performance, due to their small

processing capacity.

Previous number (D4.2.3): R104
Updated: no
Importance: optional
Implementation order: 8.8