Project no. IST-033576

XtreemOS

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

Requirements Capture and Use Case Scenarios
D4.2.1

Due date of deliverable: November 30th 2006
Actual submission date: January 11th 2007

Start date of project: June 1st 2006

Type: Deliverable
WP number:4.2
Task number (optional): T4.2.2

Name of responsible:Bernd Scheuermann
Editor & editor's address:Bernd Scheuermann
SAP Research, CEC Karlsruhe
Vincenz-Prießnitz-Str. 1
76131 Karlsruhe,Germany

Version 5.0/ Last edited by Bernd Scheuermann / Date January 11th 2007

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)	

Keyword List:
Application, requirement, use case, scenario

Revision history:

Version	Date	Authors	Institution	Sections Affected / Comments
1.0	25/10/2006	Bernd Scheuermann	SAP	Initial Version
2.0	09/11/2006	WP4.2	All teams	1st draft for internal review
3.0	17/11/2006	WP4.2	All teams	2nd draft for internal review
4.0	30/11/2006	WP4.2	All teams	Revised according to internal reviewers' comments. Final version.
5.0	11/01/2007	Bernd Scheuermann	SAP	Minor modifications

Executive Summary

WP4.2 (“Applications, Experiments and Evaluations”) is responsible for defining requirements from the application point of view and for testing and evaluating the XtreemOS implementations provided by the development work packages in SP2 and SP3 using reference applications thereby ensuring that the end-user perspectives from different sectors will be incorporated into XtreemOS development.

In this deliverable, we describe the requirements of the applications of eight different teams in WP4.2. These requirements have been acquired in cooperation with the XtreemOS development work packages to make the composition as complete as possible. They serve as a source of inspiration and as a guideline, but they will also be used as a basis for quality management. In later phases of the project, however, these initial requirements will be revised, extended and modified where appropriate driven by evaluation results and by the activities in SP2 and SP3.

Furthermore, two types of use cases are provided: application-specific use cases and general use cases. Application-specific use cases describe the usage scenarios and conditions for experimental studies with applications executed on XtreemOS. General use cases illustrate typical usage scenarios of XtreemOS irrespective of a specific application. Pairwise combinations of both types of use cases define the experimental setup for the evaluation that will be carried out in the following phases of the project. Like the requirements, also the use cases will evolve during the project lifetime.

Table of Contents

Introduction

1.1 Teams Involved

Table 1 gives an overview of the teams involved in WP4.2. Each team contributes up to three applications which will be used for experiments and evaluations with XtreemOS. The applications cover a wide spectrum of areas from academia and industry including computationally-intensive as well as data-intensive applications. From applications, the requirements for XtreemOS have been gathered according to the following process.

Table 1: Overview of teams in WP4.2

Partner	Name	Country
BSC	Barcelona Supercomputing Center	Spain
EADS	European Aeronautic Defence and Space Company	France
EDF	Electricité de France	France
SAP	SAP AG	Germany, United Kingdom
T6	T6	Italy
TID	Telefónica I+D	Spain
UDUS	University of Düsseldorf	Germany
XLAB	Xlab	Slovenia

1.2 Requirements Consolidation Process

The acquisition and consolidation of requirements during the first six months of the project has been a very interactive process is visualized in Figure 1.

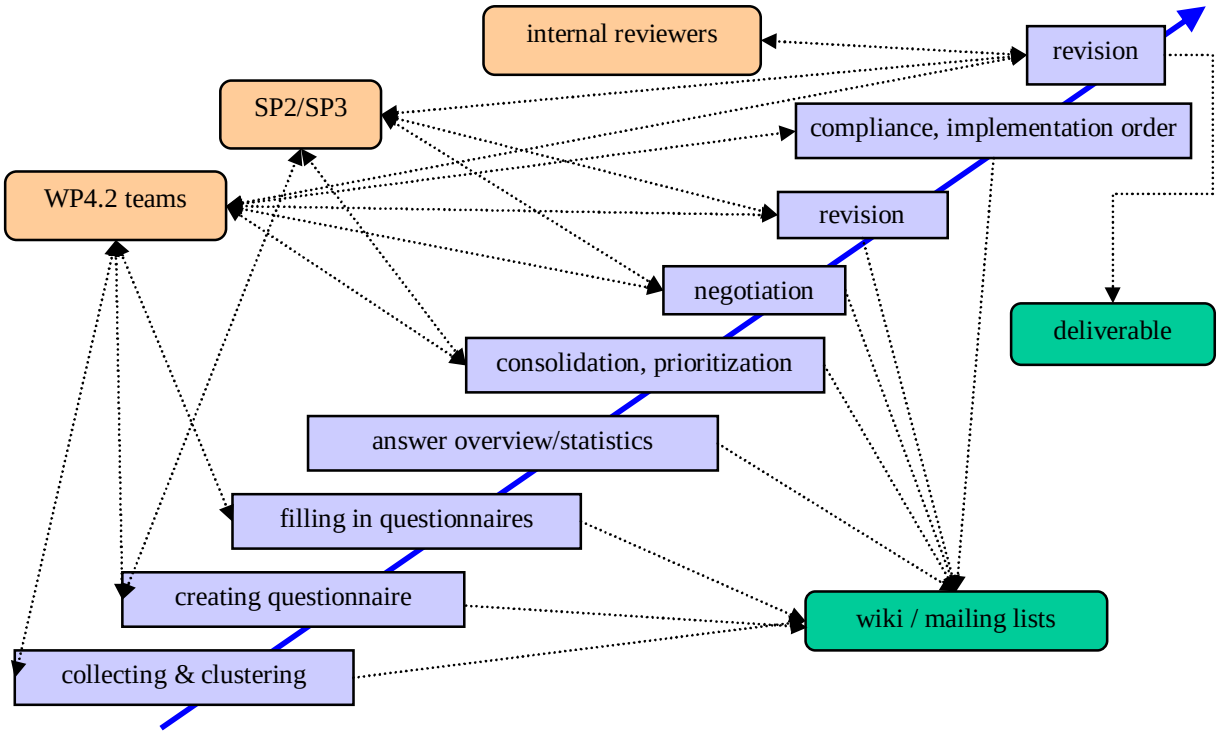


Figure 1: Process of acquiring and consolidating requirements during the first six months of the project

During the kick-off meeting in Rennes we performed a brain storming and clustering of requirements on a meta-level. Afterwards, a questionnaire was created in cooperation with implementation work packages in SP2 and SP3. These work packages described which aspects and topics should be covered within the questionnaire. The questionnaire queried general requirements as well as specific requirements directly addressing the various implementation work packages. Each team in WP4.2 filled in the questionnaires (one questionnaire per application) which were then evaluated to create an answer overview and statistics. Based on the answers we formulated the first draft of consolidated requirements and assigned priorities. These priorities express how important a certain requirement is for the applications: priority “obligatory” indicated that the respective requirement must be implemented to ensure that the application can be executed, whereas priority “optional” means that the requirement is useful but the application can still be executed without the requested functionality.

During the XtreemOS workshop at Düsseldorf we negotiated the requirements with all partners in the consortium receiving manifold feedback which was incorporated to produce the next revision of consolidated requirements. Afterwards all teams filled in checklists (one per application) in which they could verify if the requirements (and the assigned importance) are compliant with their applications, i.e. it is expected that the application can be executed on XtreemOS as defined by the requirements (or at least can be executed after an acceptable amount of modifications to the application). Furthermore, for each requirement, the teams specified in which order the respective functionalities should be implemented by SP2 and SP3. The implementation order was expressed 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 feedback of internal reviewers has been considered during the final revisions of this deliverable. As indicated in Figure 1, the intermediate documents of the acquisition and consolidation process have always been reported on the wiki pages and mailing lists to provide SP2 and SP3 early feedback during the XtreemOS specification phases.

1.3 Outline

The remainder of this deliverable is structured as follows: Section 2 presents an overview and statistical information of the applications in WP4.2 followed by brief descriptions of the respective applications. Section **3** outlines the two types of use cases: General XtreemOS use cases and application-specific use cases. Application requirements are described in Section 4. This is followed by a conclusion and an the references list. The Appendix (published in separate document) comprises the requirements template and the overview of the responses for the individual applications.

2 Applications

In this section, we give an overview of all applications in WP4.2 along with a statistical evaluation of the application characteristics. This is followed by brief descriptions of the applications, their implementation status and expectations towards XtreemOS.

2.1 Overview

Table 2 gives an overview of the applications considered in WP4.2, together with the areas they are used in. Many of the applications belong to the domain of enterprise solutions. Two applications are in the field of optimization algorithms. Two mobile solutions (namely TIFON and JOBMA) are contributed by TID. The remaining applications are widely spread over different application areas.

Table 2: Overview of applications in WP4.2

Partner	Application Name	Short Name	Application Area
BSC	Specweb2005	SPECWEB	Enterprise solutions
BSC	GRID superscalar fastDNAml	GSDNA	Bio-informatics
EADS	Elfipole	ELFIPOLE	Electromagnetics
EADS	HLA	HLA	RT simulation
EDF	Moderato	MODERATO	Particle physics
EDF	Simeon	SIMEON	Optimization
EDF	Zephyr	ZEPHYR	Fluid mechanics
EDF	Secured Remote Computing	SRC	Enterprise solutions
SAP	SAP Web Application Server	WebAS	Enterprise solutions
T6	DBE	DBE	Enterprise solutions
TID	Tifon	TIFON	Instant messaging
TID	Job Management Application	JOBMA	XtreemOS job management
UDUS	Wissenheim	WISS	Virtual Presence
XLAB	Galeb	Galeb	Economics, optimization

In the following , we give a statistical evaluation of the application characteristics. The respective data has been collected using the requirements questionnaire (see Appendix).

Current Application Status

Figure 2 shows the status of the applications at the beginning of the XtreemOS project. Five of the applications are already developed (either finished or undergoing further improvements) but some of these applications potentially need to be modified to efficiently utilize XtreemOS capabilities. The remaining applications are still being developed and can therefore specifically target XtreemOS features during the development phase.
Question referenced: Q1.1

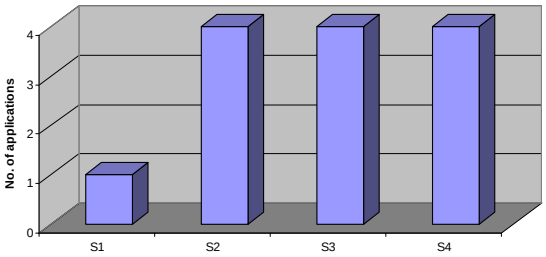


Figure 2: Current status of applications. S1 = application is finished, S2= application is finished but undergoing further improvements, S3 = application is being developed and runs, but is not stable, S4 = application is being developed and does not run at all

Programming Languages

Figure 3 shows the programming languages that are/have been used for the different applications. Here, C/C++ and Java clearly dominate. Other languages like Fortran, MPI, HLA and ABAP are used only rarely.
Question referenced: Q1.2

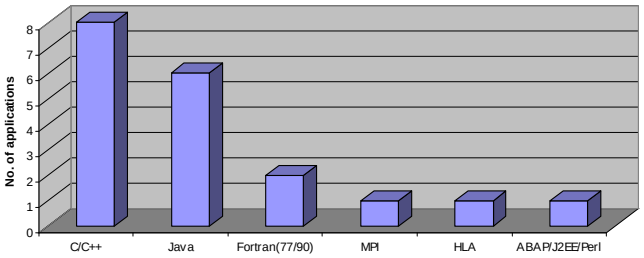


Figure 3: Programming Languages used

Layer of Application
The vast majority of the applications (12) belong to the end user layer, whereas five of the applications are listed as being a part of the middleware layer.
Question referenced: Q1.3

Operating Systems Supported
Figure 4 shows the actual operating systems which are supported by the different applications. The majority use Linux and Unix, while some applications run on Windows, MaCOS, Z/OS, OS/400, and OS/390. Two applications will support Linux MD by the end of the project. Only a single application supports Plurix and only one application is platform independent.
Question referenced: Q1.4, Q1.5

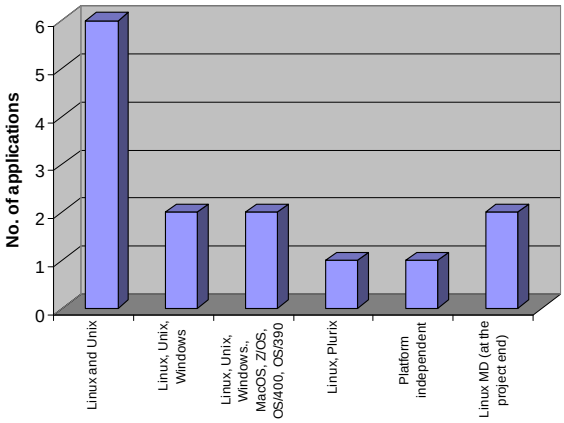


Figure 4: Operating systems supported by the different applications.

Linux packages required
One application requires Openldap, IBM Java2-JrC, and bc. Several applications need Java 1.5. Another application requires X11, GLX, and MESA installed to run.
Question referenced: Q1.6

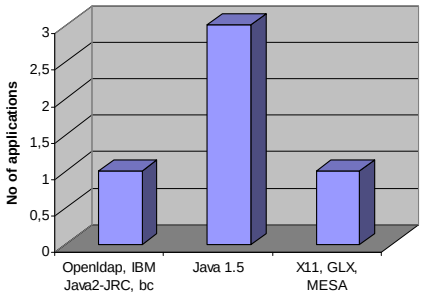


Figure 5: Linux packages

Linux Daemons
Linux daemons required are sshd and sldap.
Question referenced: Q1.7

Average Application Runtime
The average running time for the different applications is given in Figure 6. The majority have a runtime of less than 10 hours. However, some applications have an average runtime of between several days to a month, and even up to infinity.
Question referenced: Q1.8

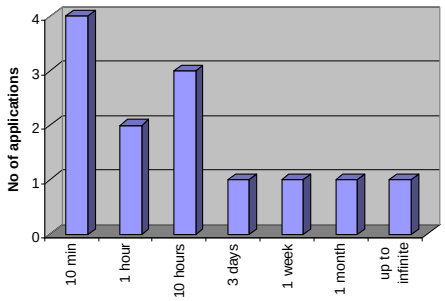


Figure 6: Average Application Time.

Maximum Application Runtime
Figure 7 illustrates the maximum application runtime. The maximum runtimes are much higher than the average runtimes. Here several applications have a possible infinite runtime, e.g. such applications that produce even better results with an increasing execution time. The smallest maximal runtim is about 30 minutes.
Question referenced: Q1.9

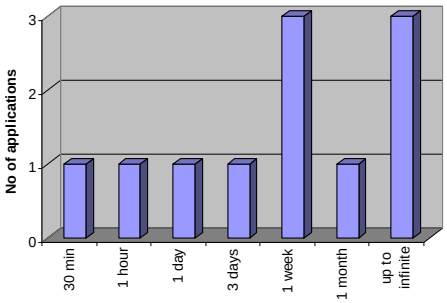


Figure 7: Maximum Application Time.

Readiness to be Executed on a Grid
Currently four applications are ready to be executed on a Grid or have already been executed on Grid using different toolsets. The majority however (9 applications), cannot run on a Grid at the moment.
Question referenced: Q1.10

2.2 Application Descriptions

In the following, we give brief introductions to the applications in WP4.2 including architectural and functional descriptions, the current development status and roadmap as well as the expectations with regard to XtreemOS.

2.2.1 Description of SpecWEB2005 (BSC)

2.2.1.1 Introduction

SPECweb2005 is a software benchmark product developed by the Standard Performance Evaluation Corporation (SPEC), a non-profit group of computer vendors, system integrators, universities, research organizations, publishers, and consultants. It is designed to measure a system's ability to act as a web server servicing static and dynamic page requests.
SPECweb2005 is the successor of SPECweb99 and SPECweb99_SSL, offering the capabilities of measuring both SSL and non-SSL request/response performance, and continues the tradition of giving Web users the most objective and most representative benchmark for measuring web server performance.

Rather than offering a single benchmark workload that attempts to approximate the full range of web server workload characteristics found today, SPECweb2005 has chosen 3 typical workloads; one for banking, one for e-commerce, and another for a support site. These workloads are based on real data obtained in the industry. Additionally, the inclusion of a concurrent connection-based workload metric is intended to offer a more direct correlation between the benchmark workload scores and the number of users a web server can support.

2.2.1.2 Current status

At this stage in time, SPECweb2005 is a well-established and stable application (currently at version 1.10). It is versatile and can be used on various versions of Unix as well as Windows. One restriction, pertaining to the Web Server being tested by this application, is that it must provide JSP or PHP technology. A license can be ordered from the following website, which also provides support to the user: http://www.spec.org/web2005/

2.2.1.3 Detailed overview

Logical Components of SPECweb2005

SPECweb2005 has four major logical components: the clients, the prime client, the web server, and the back-end simulator (BeSim). These logical components of the benchmark are illustrated, below:

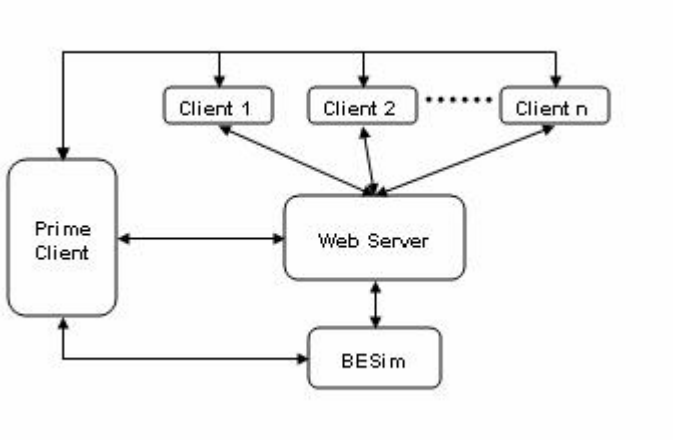


Figure 8: Logical Components of SPECweb2005

Client

The benchmark clients run the application program that sends HTTP requests to the server and receives HTTP responses from the server. For portability, this application program and the prime client program have been written in Java. Note that as a logical component, one or more load-generating clients may exist on a single physical system.

Prime Client

The prime client initializes and controls the behavior of the clients, run initialization routines against the web server and BeSim, and collect and store the results of the benchmark tests. Similarly, as a logical component it may be located on a separate physical system from the clients, or it may be on the same physical system as one of the clients.

Web Server

The web server is the collection of hardware and software that handles the requests issued by the clients. In this documentation, we shall refer to it as the SUT (System Under Test) or Web Server. The HTTP server software may also be referred to as the HTTP daemon.

Back-End Simulator (BeSim)

BeSim is intended to emulate a back-end application server which the web server must communicate with in order to retrieve specific information needed to complete a HTTP response (customer data, for example). BeSim exists in order to emulate this type of communication between a web server and a back-end server. The BeSim design documentation is located here:

http://www.spec.org/web2005/docs/1.10/design/BeSimDesign.html

Performance Metrics

The primary performance metric for reporting benchmark results is SPECweb2005. Additionally, there are sub-metric scores for each workload displayed in the workload-specific results files. It is important to understand the significance of both the primary metric scores and the sub-metric scores for SPECweb2005 results.

Primary Metric (SPECweb2005)

The primary metric, SPECweb2005, is the geometric mean of the ratio of each of the workload sub-metric scores to their respective workload reference scores, with that result multiplied by 100, as shown in the formula, below:

$$SPECweb2005 = \sqrt[3]{\left(\frac{bank\_measured}{bank\_reference}\right) \times \left(\frac{ecommerce\_measured}{ecommerce\_reference}\right) \times \left(\frac{support\_measured}{support\_reference}\right)} \times 100$$

Figure 9: SPECweb2005 Primary Metric Calculation

The SPECweb2005 primary metric provides an overall system score relative to the sub-metric scores for the reference system. Since the reference system would have a score of 100 using this calculation, a score of 200 would represent an overall score that is double the reference score. The geometric mean was chosen in order to prevent any single workload sub-metric score from having excessive weight in the primary metric score.

Workload Sub-metric

Workload sub-metric scores are in units of simultaneous sessions. They represent the number of simultaneous user sessions the SUT was able to support while meeting the quality-of-service (QOS) requirements of the benchmark. While the primary metric offers a relative, overall performance score for the system being measured, the sub-metric scores offer a workload-by-workload view of a system's performance characteristics in units of interest for real-world web sites.

SPECweb2005 Internals

Architecture

The SPECweb2005 benchmark is used to measure the performance of HTTP servers. The HTTP server's workload is driven by one or more client systems, and controlled by the prime client. Each client sends HTTP requests to the server and validates the server's responses. When all of the HTTP requests have been sent and responses received to make up a web page (typically, this is a dynamic response plus any embedded image files), the number of bytes received, the response time, and the QOS criteria met for that web page transaction is recorded by the client.

Prior to the start of the benchmark, one or more client processes is started on each of the client systems. These processes either listen on the default port (1099) or on another port specified by the user in Test.config. Once all client processes have been started, the client systems are ready for the workload and run-specific initialization by the prime client.

The prime client will read in the key value pairs from the configuration files, Test.config, Testbed.config, and the workload-specific configuration file (ex: SPECweb_Banking.config), and perform initialization of the web server and BeSim. Upon successful completion, it will initialize each client process, passing each client process the configuration information read from the configuration files, as well as any configuration information the prime client calculated (number of load generating threads, for example). When all initialization has completed successfully, the prime client will start the benchmark run.

At the end of the benchmark run, the prime client collects this result data from all clients, aggregates this data, and writes this information to a results file. When all three iterations have finished, an ASCII text report file and an HTML report file are also generated.

Implementation

The number of simultaneous sessions corresponds to the number of load-generating processes/threads that will continuously send requests to the HTTP server during the benchmark run. Each of these threads will start a "user session" that will traverse a series of workload-dependent states. Once the user session ends, the thread will start a new user session and repeat this process. This process is intended to represent users entering a site, making a

series of HTTP requests from the server, and then leaving the site. A new user session starts as soon as the previous user session ends, and this process continues until the benchmark run is complete.
The prime client controls the phases of the benchmark run. These phases are illustrated in the diagram, below:

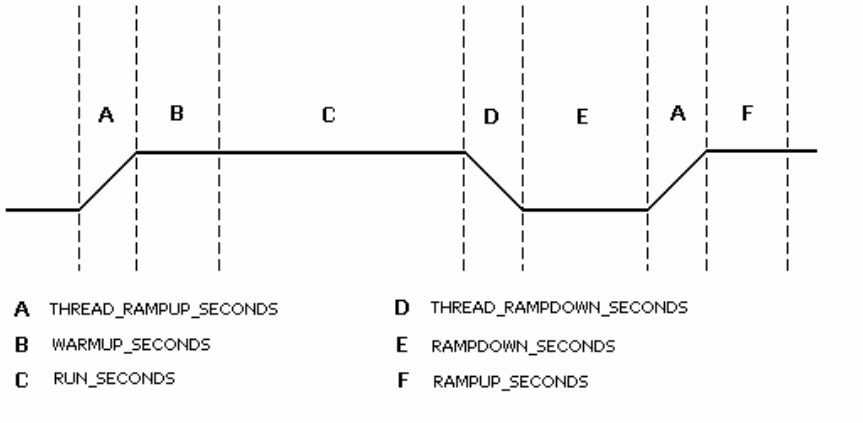


Figure 10: SPECweb2005 Benchmark Phases

The thread ramp-up period, A, is the time period over which the load generating threads are started. This phase is designed to ramp up user activity, rather than beginning the benchmark run with an immediate and full-load spike of requests.
The warm-up period, B, is intended to be a time during which the system can prime its cache prior to the actual measurement interval. At the end of the warm-up period, all results are cleared from the load generator, and recording starts anew. Accordingly, any errors reported prior to the beginning of the run period will not be reflected in the final results for this benchmark run.
The run period, C, is the interval during which benchmark results are recorded. The results of all HTTP requests sent and responses received during this interval will be recorded in the final benchmark results.
The thread ramp-down period, D, is simply the inverse of A. It is the period during which all load-generating threads are stopped. Although load generating threads are still making requests to the server during this interval, all recording of results will have stopped at the end of the run period.
The ramp-down period, E, is the time given to the client and server to return to their "unloaded" state. This is primarily intended to assure sufficient time for the TCP connection to clean-up before the start of the next test iteration.
The ramp-up period, F, replaces the warm-up period, B, for the second and third benchmark run iterations. It is presumed at this point that the server's cache is already primed, so it requires a shorter period of time between the thread ramp-up period and the run period for these subsequent iterations in order to reach a steady-state condition.
A load-generating thread will make a dynamic request to the HTTP server on one connection, and then it will reuse that connection as well as an additional connection to the server to make parallel image requests for that page. This is intended to emulate the common browser behavior of using multiple connections to request the page image files from the server. Note that the load-generating thread does not extract the images from the web page returned, as that would create unnecessary page parsing burden for the load-generating threads. Instead, the page image files to be requested for each dynamic page are retrieved from the workload-specific configuration file.
For each page requested by a load-generating thread, the load generator will start a timer immediately before sending the page request to the HTTP server, and it will stop the timer as soon as the last byte of the response for that page is received. It will likewise time the responses for all supporting image files and add those response times to the dynamic page response time in order to arrive at a total response time for that page that includes all supporting image files. Valid responses will then have their aggregate page response time checked against their respective QOS values for the workload, and the value for the corresponding QOS field (TIME_GOOD, TIME_TOLERABLE, or TIME_FAIL) will be incremented. For some workloads a byte rate-based QOS may be used where it is more appropriate.
At the end of a run, the prime client aggregates the run data from all of the clients and determines whether the run has met the benchmark QOS criteria.

Think Time and HTTP 304 Server Response

Another new feature for the SPECweb2005 benchmark is the inclusion of "think time" between user session requests (i.e. between workload "states"). After the initial request in a user session, the load generating thread

will calculate a random, exponentially-distributed think time between the values of THINK_INTERVAL and THINK_MAX, with an average value of THINK_TIME, as specified in each workload's configuration file.
The load generating thread will then wait that amount of time before issuing another request for that session. This delay is meant to more closely emulate end-user behavior between requests. In doing so, connections to the server are kept open much longer than they would otherwise, and benchmark tuning requires more judicious choices for the server's keep-alive timeout value (particularly for SSL connections), as is the case in real-world web servers.
SPECweb2005 has also added support for generating server 304 responses (not-modified-since). This is accomplished by calling the HTTP server's init script, which returns the current time on the server. The harness then uses that value as the time value in all subsequent "if-modified-since" requests to the HTTP server, assuring that the server will return an HTTP 304 response. How frequently each static image request results in a 304 response is controlled via the "304 response %" value assigned to each of the static image files in the workload-specific configuration file.

Workload Design

Due to the high variability in the security requirements and differences in the dynamic content in various web server workloads, it is becoming impossible to characterize the web performance of any server with any single workload. It is for this reason that SPECweb2005 will use three different workloads to characterize the performance of a web server. All three workloads are based on real applications. The first workload is based on online banking and all requests in this workload are SSL based. The second workload is based on an Ecommerce application so this workload includes SSL as well as non-SSL based requests. The third workload is based on downloading support patches for computer applications. The intention of the third workload is to test the ability to download large files and no SSL is used in the third workload. As an attempt to represent the user's behaviour, all three workloads are based on requests for web pages, as opposed to requests for individual files. Each page request involves running a dynamic script at the server end, returning a dynamically created file. This is usually followed by requests for embedded images and other static files associated with the file. A page is therefore fully received when all the files associated with the page are received. This new benchmark also includes QOS requirements for each page. For all pages except the large downloads, the QOS is based on page return delay. For the large pages, the QOS is based on byte rate.

2.2.1.4 Expectations and Roadmap

Since SPECweb2005 is already a stable application, at this stage we don't expect too much modification of the application itself for use with XtreemOS. It will be interesting to see any pitfalls that may be encountered while trying to get it to work under the new OS and it could be used as a guide for other pre-built applications that are attempting to port themselves to XtreemOS.
It is very likely that the Web Servers, which are going to be tested by SPECweb2005, may require heavy modifications to be able to run efficiently over the distributed environment provided by XtreemOS. It's envisioned that the availability of SPECweb2005 on XtreemOS will be extremely useful to both the development and fine tuning of Web Servers on XtreemOS, as well as a great advantage to the field of capacity planning in this distributed environment. Using this application will make it possible to quantify the capabilities of running a particular Web Server over XtreemOS.
The roadmap is not concrete at the moment since it's dependant on certain things being supported by XtreemOS. The provision of a JSP enabled Web Server and a running JVM on XtreemOS are two key items that need to be addressed before decisions can be made on this. Proposals worth considering are developing/extending an open source web server, such as Tomcat, purposed specifically for use in XtreemOS. Also, it could be used to help SAP port their WebAS application to XtreemOS efficiently.

2.2.2 GRID superscalar fastDNAml (BSC)

2.2.2.1 Introduction

Maximum likelihood methods of statistical inference were first developed in the 1930's by R.A. Fisher. Theoretical application to phylogenetic analysis was developed by Felsenstein in the 70's and early 80's [Felsen81]. Maximum likelihood methods of phylogenetic inference are superior to some other methods, particularly when the data set includes highly divergent sequences, which are desirable but increase the computational difficulty enormously.
Olsen's et al. fastDNAml, is a very well established and commonly used code for maximum likelihood phylogenetic inference [Olsen94]. The original sequential version of the code has been parallelized by different research groups. MPI and PVM versions by a group from the Indiana University [Stewart01] have been available for some time.

GRID superscalar is a programming environment that allows easily porting non grid applications to the grid and parallelising them at the same time.
In this section we discuss the GRID superscalar version of fastDNAml which was developed at Barcelona Supercomputing Center.

2.2.2.2 Current status

The GRID superscalar version of fastDNAml was developed in 2005 and was presented at the Cluster Computing and Grid 2005 conference in Cardiff [Dialinos05]. It is written in C and relies on the GRID superscalar environment for all its Grid related functionality. GRID superscalar in turn uses the Globus GRAM client API.
It runs on Linux and several other UNIX implementations. At this point, the application has been fully developed and tested. Nevertheless, the GRID superscalar environment on which it is laid on, is being enhanced on a regular basis.

2.2.2.3 Detailed overview

This section gives an overview of the original application structure, the GRID superscalar version and the underlying GRID superscalar runtime.

GRID superscalar overview

GRID superscalar is a Grid programming environment that enables simple programming of applications that will be efficiently run on a computational Grid. GRID superscalar is able to parallelize, at the runtime and at the task level, a sequential application and execute the application in a computational Grid. The approach used is able to take advantage of those applications that are composed of coarse grained tasks. These tasks can be of the size of a simulation, a program, a solver... This kind of application is very common in bioinformatics, computational chemistry and other scientific fields.
The following figure shows the GRID superscalar system architecture. The process of generating and executing an application with GRID superscalar is composed of a static phase and a dynamic phase. In the static phase, a set of tools for automatic code generation, deployment and configuration checking are used. In the dynamic phase the GRID superscalar runtime library and basic Globus Toolkit 2.4 services are used.

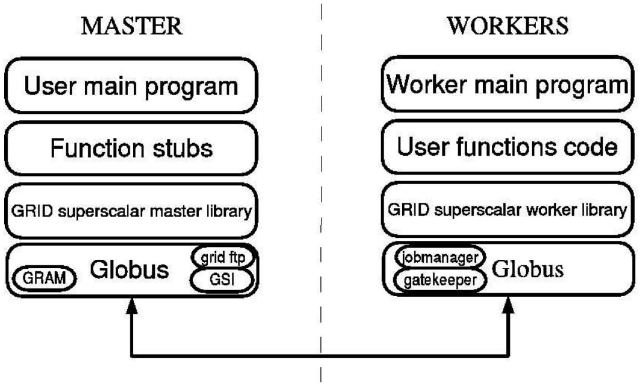


Figure 11: GRID superscalar system architecture

The application developer provides a sequential program, composed of a main program (User main program) and application tasks (User functions code), together with an IDL file that identifies the coarse grained tasks. . The IDL definitions of the tasks also specify the directionality of each of their parameters. That is, whether a parameter is an input parameter, an output parameter or an input and output parameter.
In the static phase, two sets of files are automatically generated by the GRID superscalar tools (gsstubgen) and can be used to build an application ready to run in the Grid (between them, the Function stubs and the Worker main program).
With these two sets of files a master-worker based application is built, which has the same functional behaviour of the initial user application. The master is built from the User main program, the Function stubs, the GRID superscalar master library and the middleware libraries (Globus in this case).
The master binary is run in the localhost and submits calls, through the GRID superscalar library, to the worker binaries in remote hosts of the computational Grid. The worker binaries are built from the Worker main program, the User functions code and the GRID superscalar worker library. The workers will only execute the functionality of the tasks listed in the IDL file. The main program (master) executed in the local host will execute the rest of the application code.

Still in the static phase, a graphical interface is provided that helps users setting the Grid configuration and deploying the application (Deployment Center). After this step the application is ready to be run in a computational Grid.
In the dynamic phase, the application is started in the local host the same way it would have been started originally. While the functional behaviour of the application will be the same, the basic difference is the fact that the GRID superscalar library will exploit the inherent parallelism of the application at the coarse grain task level and execute these tasks independently in remote hosts of the computational Grid.
To exploit the parallelism of the application, the GRID superscalar runtime builds a data dependence graph where each node of the graph represents one of the coarse grained tasks of the application. The edges between the nodes of the graph represent file dependences between those tasks.
File dependences are data dependences due to files that are read/written by the tasks. Whether a task reads or writes a file is determined by its declaration in the IDL file. In this sense, a task that writes to a given file should be executed before another that reads from that file. Therefore, an edge from the first task to the second will exist in the graph. These edges define a relative execution order that should be respected.
From this task graph, the GRID superscalar runtime is able to exploit the parallelism, by sending tasks that do not have any dependency between them to remote hosts in the Grid. In each case, the GRID superscalar broker will select between the set of available hosts, which is the best suited. This selection favours reducing the total execution time, which is computed not only as the estimated execution time of the task in the host, but also the time that will be spent transferring all files required by the task to the host. This allocation policy exploits the file locality, reducing the total number of file transfers.
All Grid related actions (file transfer, job submission, end of task detection, results collection) are totally transparent for the user. For each task, the GRID superscalar runtime transfers the required files from their current locations to the selected host, submits the task for execution, and detects when the task has finished. At the end of the execution of one task, the data dependence graph is updated. As a result, a resource is now free and any ready tasks may be submitted for execution in this resource and dependant tasks on the finished task may become ready for execution. When the application has finished, all working directories in the remote hosts are cleaned, application output files are collected and copied to their final locations in the local host and all is left as if the original application had been run locally.
Summarizing, from the users point of view, an execution of a GRID superscalar application looks like an application that was run locally, with the exception that the execution has been, hopefully, much faster by using the Grid's resources. Further details about GRID superscalar can be found in [Badia03] and [GRIDsup].

FastDNAml overview

The input to fastDNAml is a set of sequences of taxa (these can be groups of genes, gene products or taxa, that is, species or other taxonomic groups of organisms). From this set of sequences, the best tree (the tree with highest overall likelihood value) is sought. The problem of searching the best tree is NP-complete, and therefore fastDNAml implements a heuristic approach.

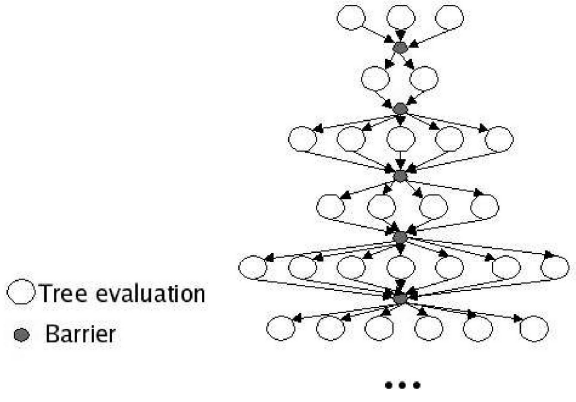


Figure 12: Architecture of the application

The general structure of the sequential application is the following: an initial tree is created using three taxa chosen randomly. The overall likelihood value for this initial tree is then evaluated. Next, a new taxon is randomly selected. There exists 2i – 5, with i = 4 in the first iteration, different possible trees when adding this taxon. Therefore, 2i – 5 evaluations are performed. Next, a local arrangement step is performed. In each iteration, this step will generate 2i – 6 evaluations. This local arrangement step may be repeated several times until any improvement is found. Next, a new taxon is added and the evaluation continues as explained before. The

process will finish when all taxa have been added. Each time a taxon is added and at each local arrangement step, several evaluations are started. These evaluations are independent from one other.
In GRID superscalar version of fastDNAml the tree evaluation function has been converted into a task. The following figure shows the architecture and attainable parallelism of the application.

2.2.2.4 Expectations and Roadmap

Currently, the fastDNAml application is running on the stable version of GRID superscalar. However, we are also developing another version in parallel which is not yet stable. This version has been designed in such a way that changing the underlying middleware is easy, while the stable version is tied a bit tighter to Globus. The programming syntax differs from one version to the other.
One possibility is to port the stable version of GRID superscalar to XtreemOS. The other possibility is to continue porting fastDNAml to the unstable version and then to port the unstable version of GRID superscalar to XtreemOS. As XtreemOS takes shape, a decision will be made on this.
We expect to benefit from porting the application to XtreemOS by taking benefit of the following services:

1 a resource discovery service
2 a resource selection service
3 a job execution service
4 a file replica management service
5 a unified global file system

Our roadmap depends heavily on the availability of the aforementioned services. We plan to port the GRID superscalar runtime in three phases. In the first phase, we will port all usage of the Globus library to the XtreemOS functional equivalent. This phase depends on the availability of the following services:

- a job execution service with end of job notifications
- a unified global file system

In the second phase, we will replace the current implementation of the static resource list with the resource discovery service provided by XtreemOS. Finally, in the third phase, we will replace the GRID superscalar's resource selection by using XtreemOS's equivalent service. This roadmap will be refined according to the roadmaps of the different work packages involved in the implementation of the required services.

2.2.3 Elfipole (EADS)

2.2.3.1 Introduction

Elfipole is a software chain performing electromagnetic wave propagation simulations. It is widely used internally by EADS business units for the design of aerospace products and their response to external events - should these be natural or not. Typical examples of such events are lightning (which is natural) and radar waves (which is not). Our airplanes must be protected from both kinds of events, and the purpose of the Elfipole software chain is to study the impact of such phenomena on our products.

2.2.3.2 Current status

Elfipole is used in enterprises covering the aerospace & car industry sectors around the world. Elfipole has been constantly improved over the years, integrating new numerical kernels and/or physical behaviors. The program is mainly coded in C & FORTRAN and used MPI for parallel processing. Operating Systems supported are Windows, Linux and most of Un*x (AIX, HP/UX, Solaris, …).

2.2.3.3 Detailed overview

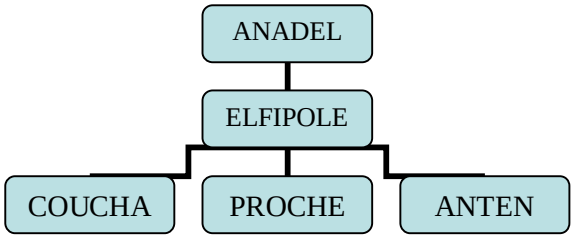


Figure 13: The Elfipole software chain is made of several applications that are interconnected.

ANADEL

ANADEL is a component of several simulation codes where the analysis only differs in the physical framework of the study (acoustic, electromagnetism, axi-symmetric geometries, surfacic/volumic formulation, etc). As a pre-processor component of the ELFIPOLE software package, it is responsible for the analysis of the input triangular mesh, where materials are defined. The aim of ANADEL is to decide which type of finite elements and degrees of freedom will be associated with each of the mesh triangles. The results of this analysis are the mesh files called "d01".

ELFIPOLE

The ELFIPOLE software is designed to solve Maxwell equations in an unbounded domain of the space by the Boundary Equations method. The resolution is performed with direct or iterative solvers; in the case of an iterative solver we can possibly use the Fast Multipole Method (FMM) in order to speed-up the matrix-vector products. ELFIPOLE allows the solving of an electromagnetic problem with three kinds of possible sources:

- Incident plane waves
- Generators
- Spherical waves (the input is an antenna diagram).

COUCHA

This post-process will output the surface potentials on the original unv mesh to be visualized by I-DEAS.

PROCHE

This post-process will output the field radiated by a structure on points chosen by the user or on a unv mesh created by the user to be visualized by I-DEAS.

ANTEN

The anten module allows the calculation of the far field radiated by the surface potentials stem from an elfipole (or BE) calculation.

2.2.3.4 Expectations and Roadmap

XtreemOS is the opportunity to gain experience of multi level infrastructures. For each of the individual components, it is very important that the execution can occur on very reliable clusters with a guaranteed bandwidth between the nodes. Since there is a workflow between each of the components, one can expect that XtreemOS can cope with such workflows by providing Grid services such as dynamic allocation of resources, checkpointing of parallel applications and migration of jobs.

2.2.4 HLA (EADS)

2.2.4.1 Introduction

The HLA simulation (http://en.wikipedia.org/wiki/High_Level_Architecture) aims to represent a distributed real time simulation such as those used to simulate an aircraft's systems. These applications are of paramount importance for EADS and Airbus as they are used to design onboard systems and software prior to any flight tests.

2.2.4.2 Current status

Most real time simulators and simulations are proprietary. Nevertheless, one can easily design a representative application using an RTI (Runtime Infrastructure). The RTI is a part of the HLA (High Level Infrastructure) standard which has many implementations. The most famous in Europe is done by Pitch (SE), but there are also some Open Source implementations. A simulation relying on an RTI is currently being designed to be tested on XtreemOS.

2.2.4.3 Detailed overview

The HLA application is a simulation communicating through a HLA middleware. The implementation will be using either OpenHLA (http://sourceforge.net/projects/ohla) or jaRTI (http://tim.littlebluefrog.com/?cat=5). It will be organized into four models (at least), represented in Fig 1.

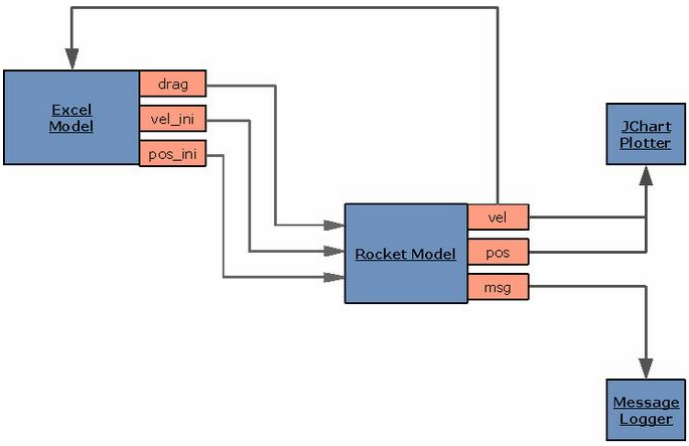
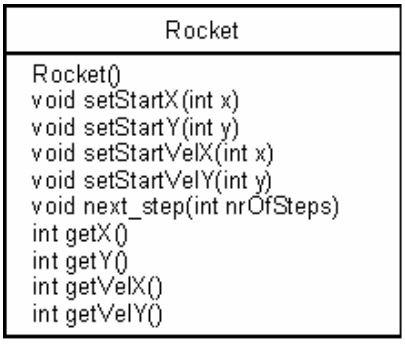


Figure 14: Data exchange in the planned simulation

Rocket Model

This model is a single Java class described by the following interface:



It publishes the following information:

- Velocity to be used by the plotter and atmosphere models
- Position to be used by the plotter model
- Message to be logged by the Message Logger model

Excel model

The atmosphere is initially modelled by an Excel model. For the purpose of XtreemOS, this model could be used to demonstrate interoperability with Windows OS or could be transformed to a java model.

Data Logger

This model aims to store all the messages provided by the rocket model.

Plotter model

This model aims at providing a graphical representation of the simulation.

2.2.4.4 Expectations and Roadmap

Obviously in an industrial project, most of the components are done by several different contractors that can be partners on one particular project and competitors on another. The concept of virtual organizations shall be evaluated to isolate users and models (possibly competitors) from each other. This aspect is especially important and complementary with EADS's previous application.

2.2.5 Simulations (EDF)

2.2.5.1 Introduction

We present here three different simulation applications – MODERATO, SIMEON and ZEPHYR - that intend to take advantage of different features of XtreemOS. For each application, we present the general problem they intend to solve, the current architecture they use and the basic services they would require in XtreemOS to run on this system.

2.2.5.2 Current status

- MODERATO has already been ported to innerGrid. We expect a quite straightforward and fast port to XtreemOS
- SIMEON has already been ported to innerGrid. Again, we expect a quite straightforward and fast port to XtreemOS
- The ZEPHYR application already exists but the interaction between the calculation process and the plotting process is currently handled by spawning a gnuplot application, and is made by the calculation process, when a request file appears in the file system. This way of communicating will have to be improved and cleaned out to run on XtreemOS

2.2.5.3 Detailed overview

MODERATO

What is MODERATO?

MODERATO is a C++ code that has been developed to simulate radiographic inspection and complex inspection configurations. A dedicated Monte-Carlo approach for the film-foil stack handles electron generation and propagation within the film cartridge. For optimal performance, MODERATO does not rely on a generic public domain Monte Carlo engine, but uses a customized engine which is tightly linked to a CAD model, allowing it to take into account complex geometries.
In order to calculate the final radiographic image, MODERATO combines a Monte-Carlo approach to estimate scattered radiation and a straight-line attenuation model to calculate the direct radiation's contribution. This combination enables a remarkable acceleration without any significant image distortion, even for thick components that generate a large amount of scattered radiation.
The code has been validated for typical configurations on experimental results, such as build up factor measurements and plane and volumetric defects. Furthermore, the analysis of certified inspector's interpretations has led to a detectability criterion suitable for automatic interpretation. In a measurement campaign conducted in parallel to the code development, defect detection limits were determined experimentally as a function of influential parameters, and were compared to results obtained by numerical simulation. The good agreement of numerical and experimental results qualifies the code for inspection procedure qualification purposes.

Architecture of MODERATO

The MODERATO code as been changed to best map a Grid architecture where neither the network nor the computing nodes are under control. This is often the case for Grid extending over multiple buildings in large organizations.
The main Grid module is the scheduler. This scheduler is fault tolerant. It means that if a node is not responding for whatever reason, the same computation must be redirected to another free node.
MODERATO nodes are each independent of each other. No communication is needed during the computation. But the final result depends on each node's intermediate results.
The scheduler initializes the computation by sending a small set of parameters to each node. At the end of the computation, each node saves its internal state into a result file. All result files are then used by a post processing node to compute the final result. The scheduler must keep a global state to control the end of the simulation. In the MODERATO case it is the number of simulated particles that is important.

SIMEON

What is SIMEON?

In order to optimize its energy generation, EDF R&D runs calculations simulating different scenarios of generation planning on mid-range periods. In the end, statistical processing is performed to select the combination of scenarios that should give the best results
SIMEON has 5,000 lines of C++ code to implement such a process.

Architecture of SIMEON :

SIMEON computing architecture maps nicely to Grid environments. It is a scenario based simulation where all nodes are fully independent. The computing nodes all use the very same algorithm on a different data set.
The scheduler initializes the computation by sending a large data set to each node; this is known as the scenario. At the end of the computation, each node saves the final result for the given scenario. At the end of the simulation each scenario must have been copied into the user's result directory for analysis.
For four hundred scenarios the data set is about 2 or 3 giga bytes of data.

ZEPHYR

What is ZEPHYR?

ZEPHYR is a Fortran 90 code developed during an Applied Maths PhD Thesis consisting of the iterative solving of a 2 dimensional non-linear unsteady viscous Bürger's equation in a bounded square domain [0,1]x[0,1] in the time interval [0,T].
Since its first release in 1997, it has been used as a benchmark case to estimate computer or cluster performance. Because ZEPHYR's behaviour is very similar to other major EDF internal sensible codes, it is a convenient way to predict the performance EDF can expect on these major codes without taking any risk to reveal sensitive pieces of information to eventual competitors.

Architecture of ZEPHYR

The solving of the Burgers equation can be driven at a different order in space or time. It can be processed on several processors using a classical domain decomposition algorithm. As far as calculation is concerned, we provide here a simplified version of the code limited to the second order and either running sequentially or running on a set of processors communicating via MPI remote processes to save the processed data and can plot them on demand.
What we need to demonstrate, is the ability to run on XtreemOS enough coherent processes to be able to browse results in quasi real time, synchronised with the ongoing calculation. The basic architecture of ZEPHYR could be graphed in this way:

Currently the calculation process is first started alone on one or several processors (depending if its execution is sequential or parallel). Regularly; the processor 0 saves the calculation results into files on the global file system. It checks as well the existence of a small user file describing the required fields to be plotted. If it exists, the processor 0 dumps the desired results on a separate file in the global file system and spawns a gnuplot process that will plot the requested data. From that moment until the closing of the plotting process, the processor 0 has complete control of the plotting process to which it sends eventual orders for redisplaying via a pipe connection.

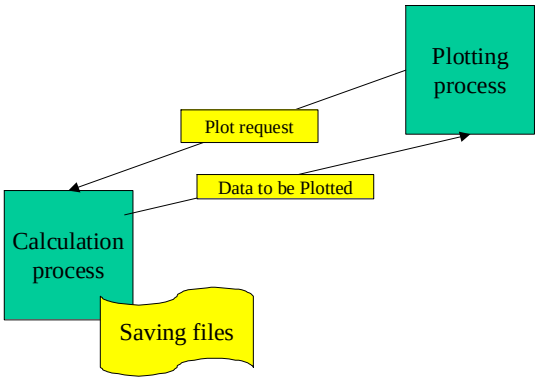


Figure 15: Architecture for ZEPHYR

By proposing ZEPHYR as a end-user application for XtreemOS, EDF wants to explore the capability of XtreemOS to handle tightly coupled applications. In this case, we wish to run a calculation on one set of nodes communicating with a single node running an MMI application which will be able to browse the data generated by ZEPHYR.

2.2.5.4 Expectations and Roadmap

Apart from the classical services of spawning, monitoring and gathering results of processes running on a Grid, MODERATO would also require the following services :

- A Fault tolerant scheduler
- A way to store the Global state for the simulation
- A way to start Post processing, when the simulation comes to an end

Apart from classical services of spawning, monitoring and gathering results of processes running on a Grid, SIMEON would require a way of setting a guaranteed high speed data transfer between nodes due to the huge amount of data that has to be routed to run a scenario.
For ZEPHYR, guaranteed synchronism as well as a decent reaction time of the MMI plotting system is expected from the end user. What we expect from XtreemOS is the possibility to guarantee a sustained bandwidth between the two processes

2.2.6 Secured Remote Computing (EDF)

2.2.6.1 Introduction

In order to run its modeling code, EDF R&D uses both internal and remote computing power. Using external resources is only possible today through a highly secured connection process that minimizes eventual threats to the EDF R&D internal information System. This connection must be a secured end to end channel that guarantees the safety of the EDF R&D intranet by preventing any access to it when the remote connection is opened.
This secured remote process is the purpose of this application.

2.2.6.2 Current status

Current solution is far from being transparent to the user, who is usually quite reluctant to have to include this additional manipulation.

2.2.6.3 Detailed overview

We hope that using XtreemOS will simplify the process for the end user. The ideal solution based on XtreemOS has still to be designed and depends on the final security features provided by XtreemOS.

2.2.6.4 Expectations and Roadmap

The goal of this application is to determine if XtreemOS could provide a satisfactory solution to our security needs. Also the Grid should grant access to remote resources in a transparent way for the user so that only one single authentification is required by the local Grid.

2.2.7 SAP Web Application Server (SAP)

2.2.7.1 Introduction

SAP Netweaver Application Server – replacing the former SAP R/3 - is the central middleware which the large majority of SAP enterprise business solutions are based on. Recently the name was changed from SAP Web Application Server to SAP Netweaver Application Server. For reasons of consistency with the XtreemOS project proposal we henceforth use the previous name SAP Web Application Server (WebAS). SAP WebAS is part of SAP Netweaver, which is used as a business process platform supporting Web Services, portal infrastructure, business process management etc.

2.2.7.2 Current status

SAP WebAS is used in enterprises practically covering all industry sectors around the world. SAP WebAS is improved constantly with version 7.1 being the current release. The program is mainly coded in C (Internet Communication Manager coded in C++). Operating Systems supported are Windows, Linux, AIX, HP/UX, Solaris, Z/OS, OS/400 and OS/390.

2.2.7.3 Detailed overview

This section gives a brief overview of the SAP Netweaver platform technology followed by a more detailed description of SAP WebAS.

SAP Netweaver

SAP Netweaver as a technology platform is the basis for the development of new software solutions and also for the integration of existing applications. This platform realizes the concept of enterprise service oriented architectures (ESOA) and covers the following four areas (cf. Figure 16):

- **People Integration**: provides the right information and functionalities to the right persons.
- **Information Integration**: composes and consolidates diverse types of information.
- **Process Integration**: allows for cooperation among heterogeneous types of components used within business processes.
- **Application Platform**: SAP Web Application Server (SAP WebAS) provides the environment for the execution of ABAP and J2EE applications. ABAP is an interpreter programming language developed by SAP for Business Processes.

Across all four areas, SAP Netweaver offers Life Cycle Management for all phases of the software life cycle including design, development, testing, operation, maintenance and management changes. The Composite Application Framework allows the creation of applications consisting of services from various areas. In the following, the SAP Netweaver technological components are described in more detail.

Multi-Channel Access

The multi-channel component belongs to the area of People Integration and allows access to enterprise systems via PC, Internet, mobile devices or speech controlled systems. Next to remote access this component also supports working offline (data can be replicated and synchronized accordingly).

SAP Enterprise Portal

The SAP Enterprise Portal offers role-based, customized but consistently designed access to enterprise information.

Business Intelligence

Integrating the SAP product Business Information Warehouse (BW), it is possible to process operative and historical data. Online Analytical Processing further allows the running of multi-dimensional analysis considering various business objectives.

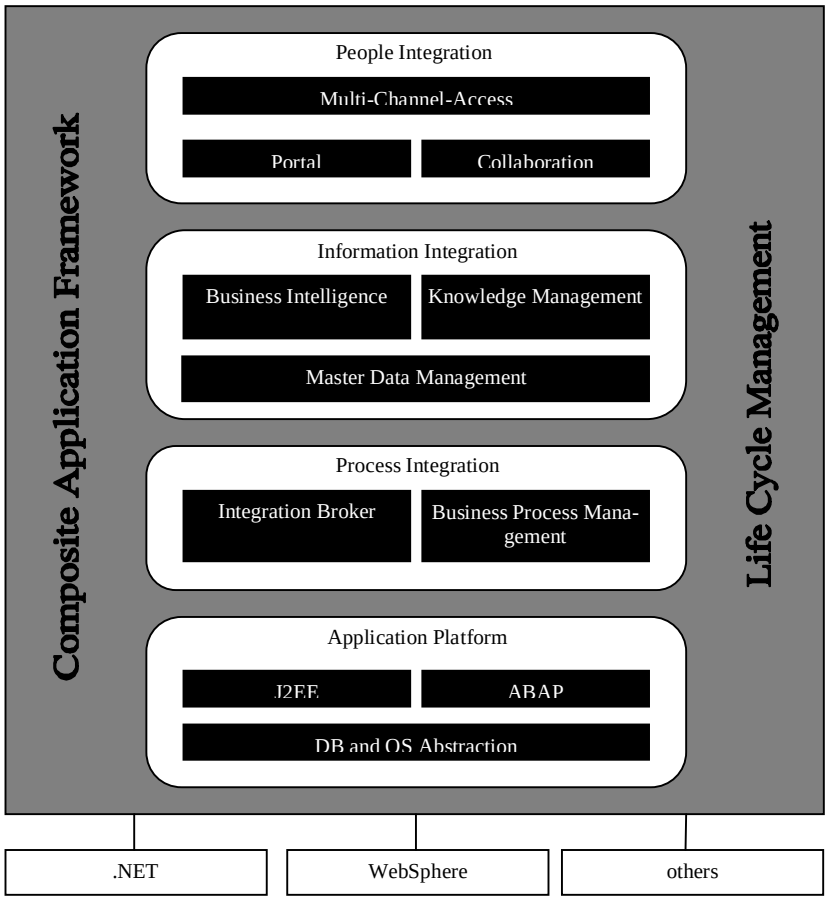


Figure 16: SAP Netweaver architecture

Master Data Management

SAP Master Data Management (MDM) consolidates master data to assure information integrity throughout the entire business network. MDM includes searching for distributed data stored on various systems and also identifying identities or similarities within the data.

SAP Exchange Infrastructure

The SAP Exchange Infrastructure (XI) is part of the area of Process Integration and comprises of the Integration Broker and Business Process Management components. The Integration Broker realizes the communication between heterogeneous application components (also from different vendors) via XML/SOAP. Based on open standards, XI allows for the definition of software components, interfaces, mapping and content-dependent routing rules.

SAP Web Application Server

SAP Web Application Server (WebAS) is the application platform for development and execution of web services and applications based on J2EE and ABAP. Refer to Section 0 for a more detailed description.

SAP Web Application Server (WebAS)

Apart from the Internet Communication Manager, SAP WebAS consists of the following three main components (cf. Figure 17):

- **Presentation**: contains presentation logic and provides for the communication with client programs (e.g. browsers).
- **Business logic**: contains business rules and processes operating on a database
- **Integration**: allows to integrate external resources via web services

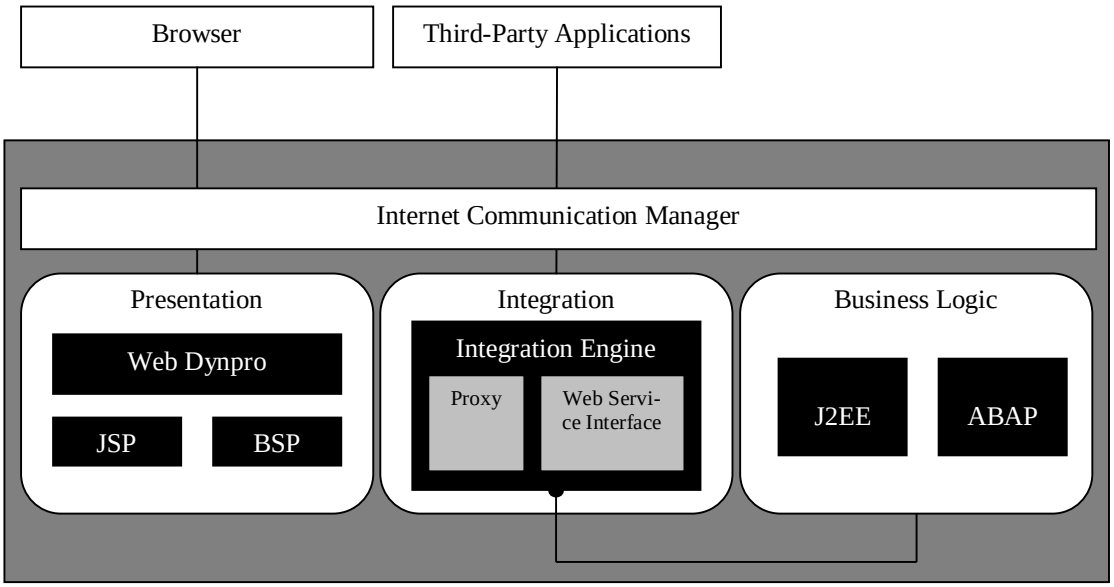


Figure 17: SAP Web Application Server architecture

Internet Communication Manager

The Internet Communication Manager (ICM) is connected to the presentation and integration component. It manages and establishes network connections. Client requests coming via different kinds of user interfaces (e.g. http requests) are transferred to the Web Dynpro runtime environment contained in the presentation component, whereas web service requests are transferred to the integration engine within the integration component. Computational results of a request are returned to the client and in addition, stored into a cache directory to quickly reply to any forthcoming requests. This cache directory stores and maintains static content as well as dynamic and active content. Using this technology improves the performance and scalability of web applications. Obsolescence of contents is avoided by active caching. Detecting application-specific events active caching allows the checking of the validity of cached data and to update the contents if needed.

Presentation

Presentations are created using Java Server Pages (JSP) or Business Server Pages (BSP) technology. Web Dynpro provides tools to graphically create web user interfaces. The abstract description is based on XML and can be transformed into JSP/BSP source code using the respective generators.
Upon client requests, the JSP Engine converts Java Server Pages into a servlet. This servlet is then loaded and executed by the servlet engine. Servlets can produce HTML, XML or WML code which is then sent to the client as a response.
If clients request a BSP resource, the BSP engine creates a BSP object and submits a predefined sequence of events to the corresponding event handler. BSP objects also create HTML, XML or WML documents which are returned to the client. Both technologies, JSP and BSP, allow the embedding of JavaScript into the HTML description.

Business Logic

Business logic can be programmed in Java (J2EE) or in ABAP. Both kinds of code, J2EE and ABAP, are compiled into intermediate platform independent code, which can be executed on the respective virtual machines. Using Enterprise Java Beans (EJB), business logic is mapped into session or entity beans and then deployed to the J2EE environment of WebAS. The advantage of this approach is that business logic can be deployed into any J2EE based application server.
Business logic in ABAP can also be created as business objects or is executed directly by the ABAP interpreter. Client applications can also be realized as a combination of both technologies, ABAP and J2EE.

Integration

The Integration Engine provides web service access to the implemented business logic. These web services are described via the Web Service Description Language (WSDL) and stored within the Integration Engine. According to this WSDL description, a client may produce SOAP messages to call a web service executing the respec-

tive business logic. Using the Integration Engine, the WebAS itself can also act as a consumer requesting a web service from another provider. Based upon the requested WSDL description, a proxy is created within the Integration Engine. This proxy can be used by the business logic to integrate certain functionalities. For instance it would be possible to include e-mail web services within a purchase order processing application.

Infrastructure

The SAP Web Application Server (WebAS) is the middleware system running in a three-tier client-server infrastructure as shown in Figure 18. Applications programmed in ABAP or J2EE are executed on one or more application servers while accessing a common database. Clients can connect to WebAS either via browser or by using SAPGUI (SAP Graphical User Interface), which is available for various platforms.

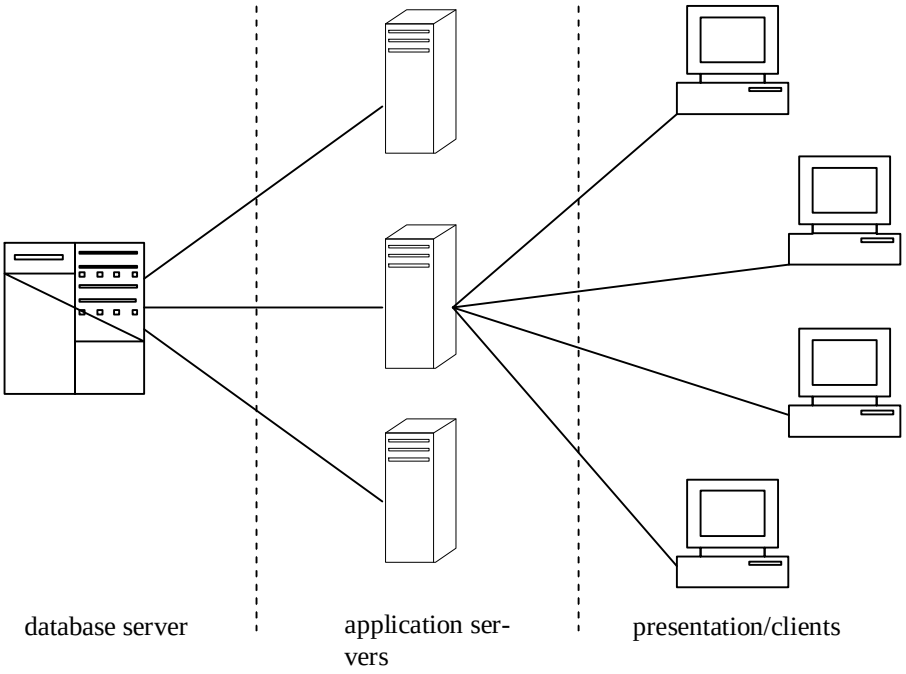


Figure 18: Three-Tier Client-Server Infrastructure

2.2.7.4 Expectations and Roadmap

So far SAP WebAS is not ready for the Grid. From the SAP perspective most important is the execution of WebAS on highly reliable clusters rather than the classical Grid scenario of many heterogeneous unreliable computing nodes. A further key issue on the implementation roadmap is to provide scalable amounts of computing resources, i.e. dynamic allocation, de-allocation and migration of jobs. The concept of virtual organizations shall be exploited to isolate users (possibly competitors) from each other. This aspect is especially important with respect to hosting scenarios. Table 3 compares the properties of current Grid approaches with the additional expectations for XtreemOS from the view of multi-tier business solutions with SAP WebAS as the middleware component.

Table 3: Comparison of existing Grid approaches with the additional expectations for XtreemOS regarding multi-tier business solutions.

Properties	Existing Grid Approaches	Additional Expectations for XtreemOS
application landscape	– automated batch processing – file-based	Efficient deployment and execution of the whole stack including: – user applications – middleware – database

Size	targeting the world-wide thing	– many administrative domains within a single administrative super-domain (hosting) – Dynamic VOs across administrative domains
job characteristics	– stateless batch jobs – short deployment times (secs) – limited execution times – highly mobile	– stateful sessions, services and jobs – long deployment times (mins, hrs) – possibly unlimited execution time – restricted mobility (licensing, sessions)
data characteristics	– relatively flat data structures – no transactional data	– large amounts of highly structured data – transactional data with semantics (Online Transaction Processing)
performance requirements	– high throughput and high resource utilization	– good response time for all interactive requests high resource utilization
service level agreements	performance isolation of individual applications	need for performance isolation across applications, middleware and DB
legacy applications	do not care about legacy	legacy support is a must with no need to adapt the source code to Grid OS (XtreemOS)
main drivers	– get the compute power – get it done at all/quicker	– improve administrative flexibility – get it done efficiently (cost vs. customer-specific quality)

2.2.8 DBE (T6)

2.2.8.1 Introduction

The major role of the Digital Business Ecosystem (DBE) is, in simple terms, to support service discovery and execution for SMEs (in relation to the Information Level Exchange [HS2002], the DBE is located at level III) with a unique approach that focuses on the evolutionary/ecosystem approach. The DBE hence assumes that services already exist and are either provided by a human being or supported by an information system in an environment that it is not under the control of the DBE; i.e. it is outside its range. In biological terms, we can say that the DBE is not involved in the creation of individuals, but instead is supporting and helping their interactions. The various software artefacts that the project will deliver are conceived and realized with the goal of helping evolution and integration but not of *developing* services. In this sense, we have to clarify that the DBE environment devoted to service creation takes care of describing it and generating all the structural code needed to manage it, but does not implement how the request is going to be fulfilled; this is delegated to the SME back office. The most correct definition still resides in its name: digital business ecosystem.

2.2.8.2 Current status

The whole DBE platform has been under development since the beginning of 2004. Still under development, early implementations at the end of 2004, have allowed partners in the project to use the platform. This means that the platform has been running for nearly two years but it is still under development. The first "stable" implementation will be released at the end of this year.
The DBE platform runs on Java and it is platform independent. It has been tested with Linux, MacOSX and Windows.

2.2.8.3 Detailed overview

Architecture

A global separation of concerns in the DBE can be identified by:

- Service Factory (SF);
- Service Execution Environment (ExE);
- Evolutionary Environment (EvE).

In the figure below we have on top, the layer representing the real business network that is made of people and tangible goods. The middle layer represents the two environments that are visible to the SME users; Service

Factory Environment (SFE) and the Service Execution Environment (ExE). Here it is evident that the data flow and messages are direct from the SFE to the ExE. The lower layer represents the Evolutionary Environment that has to implement the automatic evolutionary feature of the project; this environment is transparent and not visible to the user community.

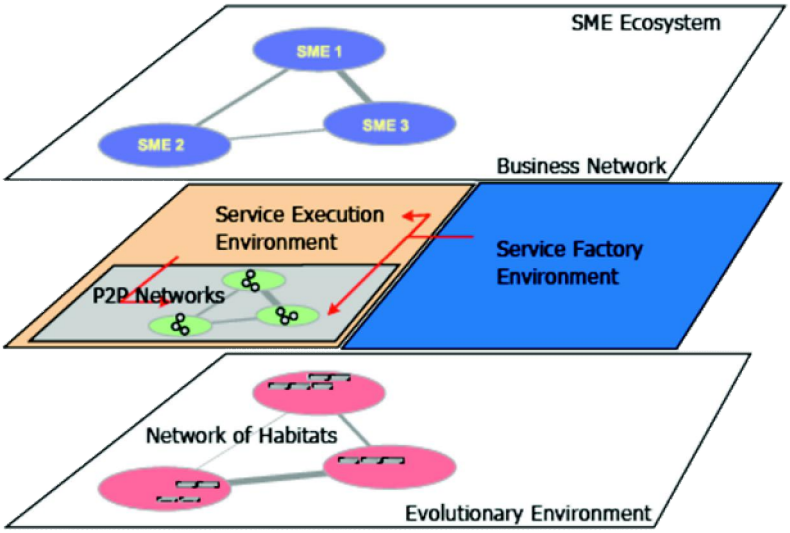


Figure 19: DBE Environments, adapted from Salzburg Technical University

The Service Factory

The SF is the place where the SMEs *creates offers* that can be related to services or products. The creation process requires the description of the many facets of the offer, which span from the business motivation of the firm to the software interface parameters that enable it. Rather then being just an extended development environment that supports coding and testing with a programming language, the SF specifically supports the complex and articulated effort of describing the offers from a business perspective which, in the current Service Oriented Architecture (SOA), are only marginally treated. The SF, in contrast, widens the horizon by taking into account the parts that are strategic to real business transactions, i.e. business location, business motivation, business process, events, processes and so forth. The current effort in IT is mostly aimed at describing services from a computing perspective, sometimes enriched with semantic specifications, and it is only marginally interested in describing it from a purely business viewpoint. The availability of an extended business description allows the realization of intelligent search mechanisms that aim at automating the B2B discovery, negotiation and consumption transactions. Enabling business and semantic discovery with pure Web Service technology is awkward, analogous to evaluating suppliers by inspecting and evaluating the trucks used to deliver the products. The SF does not only provide a set of visual tools for modeling and developing, it aims also at giving seamless support in the entire path needed to describe the offer that can be a product or a service (often these two terms converge). This path is made up of mostly sequential phases that require different actors with different objectives and skills. These phases start from pure business modeling down to platform code development, and the SF aims to support these transitions through semiautomatic model and code generation.

The Execution Environment

The DBE Service Execution Environment (ExE) is where actual services live. They are registered, deployed, searched, retrieved and consumed. This parallel world is sometimes referred to as the *runtime* of the DBE. As a consequence, the ExE is the gate to access the DBE services.
In fact, the ExE is the actual DBE Network. The DBE P2P Network is created on top of all the instances of the ExE. Each single instance of the ExE contributes with part of its resources to the whole DBE P2P system. This set of ExE instances behave as a unique system with its own identification service (distributed identity), its own persistency service (distributed storage), etc... As the ExE is the DBE application bus (nervous system), it is where all results regarding network behaviour, network topologies, stability and symmetries, self-healing, self-organization, evolutionary environment, etc... have to be applied.

Even if, from a technical point of view, the ExE can support the entire implementation and behaviour of the services, it is best described as a substrate, a soil where SME services get connected and interact. This statement is in line with DBE as being at Level-3 in the ILE (reference DBE Architecture Requirement). The DBE is not supposed to replace the back-end infrastructure systems of SMEs, but rather to act as a wide smart Internet-enabled bus that enables services to interact, evolve and integrate. Moreover the ExE provides transparent integration with the Evolutionary Environment and with basic services like accounting, security, logging, payment and so forth. In the figure below the relationship between the two environments is shown.

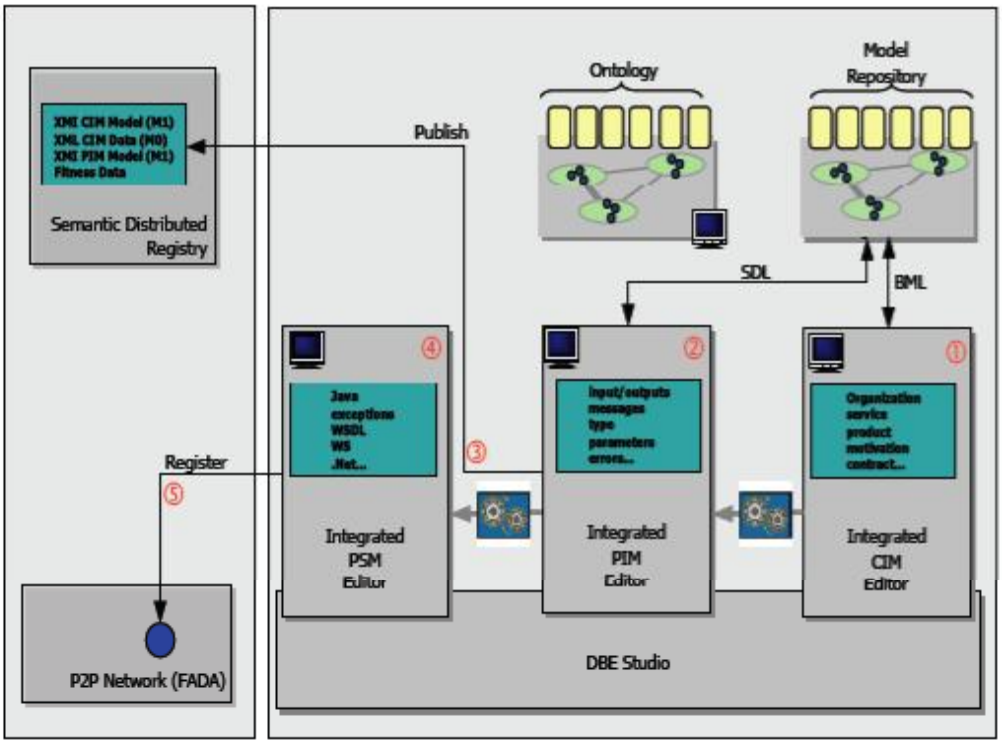


Figure 20: ExE and SFE interoperability

Evolutionary Environment

The EvE (Evolutionary Environment) is transparent with respect to the other two environments. In order to allow the ExE and the SF not to be aware of the existence of the EvE and hence not to hardcode any EvE aware code an event based publishing and subscribing message protocol has been implemented. Essentially the EvE will receive usage information from the ExE and send back the created service chains.
In more detail, the EvE has some more dependencies because it has to access the core-services running in the ExE in order to gather more information about the services. What is important to highlight is that the EvE only deals with the CIM (Computer Independent Model) part of services and it does not try to solve the PIM (Program Independent Model) mapping. This will be addressed by the manual composer. This means that the only meaningful piece of information is the CIM model; the manual composer will try to compose services also from the computing perspective, i.e. the by examining the PIM. The approach is to create several instances of EvE, one for each habitat. This mechanism will reduce the network traffic and optimize the distributed storage mechanism.

Nervous System, FADA

The middleware in the DBE that is in charge of storing and distributing the service proxies is called the DBE Nervous System. It is transparent to users, by means of the Servent (an application which is SERVer and cliENT at the same time), and the coordinated interaction among different infrastructure services provides the illusion of a single centralize service registry.
FADA (Federated Advanced Directory Architecture) is the name of SUN's implementation technology, it is based on SUN's Jini concept and it implements the Nervous System. The main features provided by FADA technology is the ability to be a decentralized storage area which maintains a link (lease) with the registrar process, in this way the service proxies are always in-sync with the process that provides access to the service. In addition

FADA is able to work on the Internet and be tolerant to some extent when there are network and hardware failures.

2.2.8.4 Expectations and Roadmap

We want to solve two main problems by porting our application to the XtreemOS framework: allowing intensive data calculations and granting service availability.
The Evolutionary Environment collects data from the network and processes it using Genetic Algorithms. It has been calculated that the Evolutionary Environment needs a high population and many iterations in order to get reasonable good results. Calculation never stops and results have to be changed with each remote call performed by one of the neighbourhood. We expect the XtreemOS framework will allow us to minimize the calculation time using Grid technology. On the other hand, we want to keep the whole environment and individual services always running and available. We want to use replication techniques in order to grant the service availability. First steps are to replicate services in different nodes, detect when a service is down in order to start another instance and detect when the CPU usage is high in order to start a new instance for the next requests.
DBE will make use of the basic XtreemOS operations, mainly related to Job Submission and Monitoring:

- The Genetic Algorithms will be rewritten to take advantage of the **Grid computation**. They will just submit jobs to different machines. In this case the basic functionality will be enough.
- We also want to keep the system always available. This task will use the **monitoring** system provided with XtreemOS (CPU Load, Memory available, etc).
- Once we can get information about the status of the resources we would be able to make decisions. These decisions will use other XtreemOS capabilities as **job migration** or **job replication** (virtual nodes).

The integration with XtreemOS will begin once the XtreemOS API is available and ported to Java. These are the steps needed in order to achieve full integration:

- Deploy services as XtreemOS Jobs (ask for location, submit job). This will spread our services within the Grid.
- Allow service monitoring in order to detect intensive CPU usage, problems with communication, etc.
- Create a most robust monitoring system, able to take decisions depending on monitoring feedback (ie. be able to throw more processes, migrate a service, choose faster service when a new request is received, etc).
- Transform the Evolutionary Environment in a Master-Slave parallel process able to run jobs in different computers and collect results.

2.2.9 Tifon (TID)

2.2.9.1 Introduction

This document presents a description of the first application TID will provide to run on XtreemOS. The description will detail the application's main features at the moment of writing. The application, called Tifon, is a PC videotelephony client which supports the main standard VoIP and instant messaging protocols, especially the Session Initiation Protocol (SIP). For the purposes of XtreemOS we will use a stripped-down version of the application, which uses only the instant messaging and presence features.

2.2.9.2 Current status

Right now, Tifon is developed in Java for the x86 platform (in particular, on J2SE 1.4). It uses SWING libraries for the graphical user interface. On an x86 PC the application is currently running and it’s almost error free. Of course, as the main goal of its inclusion was to demonstrate applications running in a mobile device, it needs to be ported to Java for the ARM architecture. It also needs to be adapted to take advantage of XtreemOS services, mainly in order to access the authorization and user information functionalities offered by the operating system (currently, Tifon uses LDAP for user authentication).

2.2.9.3 Detailed overview

Features

In this section, we will emphasize those features that are important for the XtreemOS version of Tifon; that is, instant messaging and presence (IM&P) ones.

Message exchange

Tifon, in its current development state, allows users to exchange messages through a server (a server acting as a SIP-PROXY element). Sending messages requires the user to previously connect to a server which broadly translates the contact information (which are SIP URIs) into IP addresses. The following screenshot shows the conversation window raised to chat with other users.

Figure 21: Basic message interface

Presence

Presence features give information about each contact's current state in the user's addressbook, and allows the user to be seen by others as *online*, *away*, *busy* or *offline*. Icons for each state are shown in the following figure.

Figure 22: Presence icons

Interfaces

The network interface and the user interface are the main ones in Tifon. The former consists of a Java implementation of the communications protocols that allow the application to work properly (in fact, the SIP protocol); the latter allows users to perform all necessary actions: message exchange, configuration, contact administration, etc. This section will be dedicated to describe Tifon's user interface. This interface has already been simplified to use only instant messaging and presence functionalities, so that everything related to audio/video communications won't clutter the interface. Of course, the final graphical interface can (and will) change during the adaptation to mobile devices, but no major conceptual changes are expected.

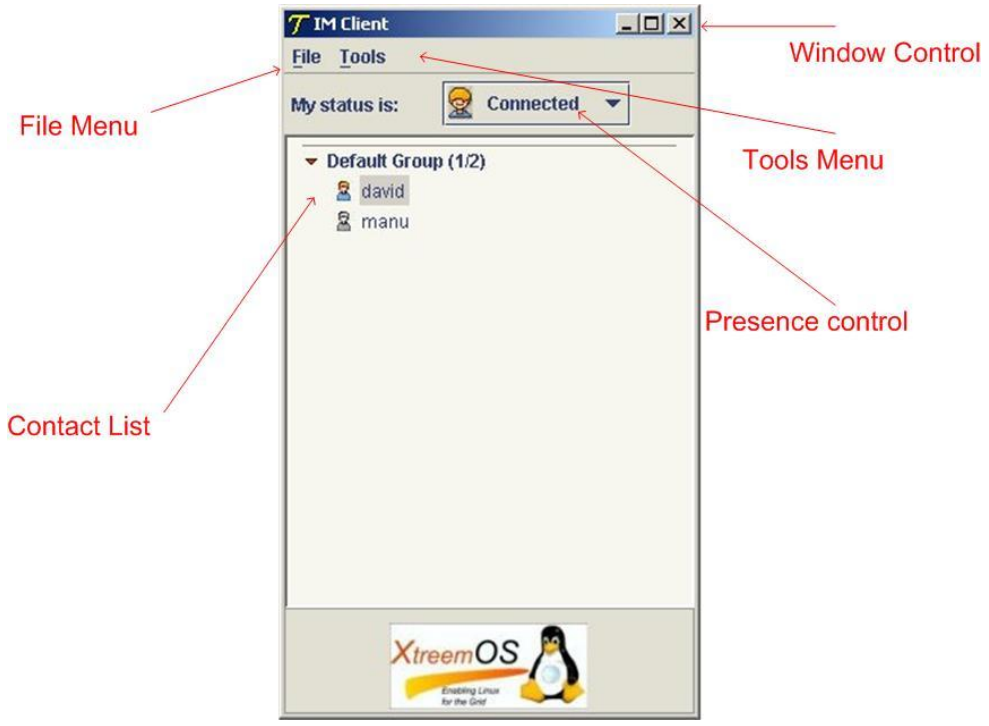


Figure 23: Tifon main window

The main elements in the Graphical User Interface, as shown in previous image, are:

1. File Menu: lets users logout.
2. Tools Menu: accesses configuration and contact administration
3. Contact List: here is the place where we can see the presence status of users in our contact list. We can also start exchanging messages with those users by double clicking their name in the contact list. Also contact administration can be done in this area, by adding, deleting and moving contacts from the different groups.
4. Presence control: here we can establish our status, which the other users will see in their Tifon instances.
5. Window control: actions related to application window.

Architecture

Due to the removal of the functionality not needed for XtreemOS (such as audio and video communication, access to users' blogs, LDAP-based user authentication, etc), Tifon architecture has been deeply simplified compared to the original one. The components related to these functionalities have also been removed from Tifon.
Current Tifon architecture consists of two main components:

- Communications: it implements all the tasks involving network interactions, like user registry, message exchange, presence information management… It's the core of all versions of Tifon on the different platforms.
- Graphical interface: on top of the communications layer/component, it lets the user interact with other users, manage his contacts, see other users' presence status through a friendly interface… etc. It could be slightly different for the various hardware architectures (platforms) to which Tifon will be ported: PCs, PDAs, Smart Phones…

2.2.9.4 Expectations and Roadmap

So far, Tifon is NOT ready for Grid environments.
Due to the nature of the application, which is neither CPU nor storage-intensive, we expect Tifon to use only the user management and VO management features offered by XtreemOS, to provide the identification and authentication of users, as well as information on users location (mostly, the IP address). Also, presence information about the users could be extracted from XtreemOS, if that kind of data is available.

The current roadmap for adapting Tifon to XtreemOS environment consists of the following steps:

1. Remove unneeded functionality from original Tifon (Audio/vide communication, blogs…)

2. Adaptation to Mobile Devices (Changes in the GUI, underlying JVM…)
3. Adaptation to XtreemOS
    a. Changes in authentication mechanisms (Tifon users are XtreemOS users and authentication will be done using OS APIs)
    b. Changes in access to user information (User information is managed and accessed through XtreemOS)
    c. Changes in presence management (If presence information is not provided by XtreemOS, Tifon could implement a P2P presence information system)

2.2.10 JobMA (TID)

2.2.10.1 Introduction

JobMA is a job management and monitoring application designed for mobile devices. It will be specifically developed for XtreemOS, therefore a very detailed description can't be provided at this point.
The goal of the application is to manage and monitor Grid jobs execution (launch, current state, final results, execution success, etc.) from mobile devices using a friendly and intuitive interface. Thus, heavy jobs which access a number of heterogeneous and geographically dispersed resources will be completely manageable from a mobile device.
This application will take advantage of XtreemOS features such as user authentication, virtual organizations (VOs) management, resource allocation, etc.

2.2.10.2 Current status

Right now, JobMA is under development, since it is an application specifically designed for XtreemOS. It will be implemented in Java for Mobile Devices, and it will use XtreemOS' job management capabilities.

2.2.10.3 Detailed overview

General features

JobMA will provide an intuitive access to XtreemOS job management capabilities (it will directly use the process management APIs and services provided by the OS), including:

- Monitor
- Launch
- Stop (pause)
- Resume
- Cancel
- View final state
- View results

JobMA intends to be a graphical interface to manage Grid jobs. Therefore, it strongly depends on the capabilities provided by XtreemOS to do so. Job information will be obtained by requests to XtreemOS, which will provide mechanisms to make this information available from any node.

Users

From now on, the user who starts the application will be called the owner of the application.
Only global Grid users will be allowed to run the application. The user will need to be authenticated to XtreemOS in order to access a job's information according to the policies established by the virtual organizations he belongs to.
Once the user has been authenticated, scheduled jobs information will be requested to XtreemOS and presented via a user-friendly graphical interface. Initially, it will only be possible to monitor jobs running in the user's VO context. However, as mentioned before, user's access to job information will be determined by VO security policies.

Monitoring

The owner will be able to select which parameters he wants to see among a set of job parameters. These parameters could be:

- The owner of the job (a global user).
- Launch time.
- Actual job status: active, stopped, cancelled, and finished.
- Input/output files location, etc.

- Node/nodes where the job is currently running

The VO policies will be able to restrict the parameters shown to users different from the owner.

Input/output

Users will be able to launch jobs from this application, including input/output files selection. Once the job has finished, graphical access to output files will be provided.

Configuration issues

It is expected that at least one configuration file will be required to store parameters such as the graphical interface's appearance, presented information, XtreemOS services URIs, etc. This file could be stored in a different node from the one where the application was started: XtreemOS Grid facilities will give access to it wherever it is stored (using the Grid Filesystem). Thus, the user will be able to access his jobs in a customized way, regardless of the node where he is connecting from.

Likely features

In this section, we will note down some features which are of less importance but could enrich the application.

- Number of nodes the job is splitted into.
- Mechanism to exactly define jobs (job workflows). For example, a simple XML file (like in GT 4).

These features will depend on the final set of capabilities provided by XtreemOS.

2.2.10.4 Expectations and Roadmap

JobMA is expected to use mainly all job monitoring and managing capabilities provided by XtreemOS on mobile devices, including:

- Basic job information: command, arguments, date launched, owner, status, input/output files...
- Advanced job information: workflow-related information, nodes hosting the job, estimated job runtime...
- Job management facilities: start, stop, resume, cancel...

Obviously, performing all these actions will also put to use XtreemOS security and authentication methods.

The roadmap for this application's development and use in XtreemOS is totally dependant on the progress made by the project itself, specially in the process management area. So, JobMA's advancements will closely follow the ones made by XtreemOS.

2.2.11 Wissenheim (UDUS)

2.2.11.1 Introduction

"Wissenheim" is a virtual presence application, implementing a multi-user interactive 3D-world. Users are represented by avatars and perceive a rich multi- and hypermedia experience. This includes animated teaching material, meeting opportunities, games, fun, and customized personal objects & spaces.

2.2.11.2 Current status

Currently, "Wissenheim" runs on top of Plurix, a proprietary operating system developed at the University of Ulm. Wissenheim is written in Java (sticking to Java 1.0 language specification). Plurix comes with its own runtime. Only a few dozen of the traditional Java classes and packages are used. Heavyweight packages like AWT or Java Swing are not supported. The additional Plurix runtime itself is rather small, respecting the goal of keeping things lean and understandable.
A first port of Wissenheim to Linux has been made which enables users to explore the world of Wissenheim on a single node.

2.2.11.3 Detailed overview

Shared Scene Graph

The central data structure in Wissenheim is the scene graph, describing the virtual world, all objects, and the avatars. Instead of using a traditional Client/Server approach, we use a peer-to-peer solution based on the Plurix DSM. The scene graph is stored in the transactional DSM and registered in the global name service. Thus each participating station can transparently access and modify the scene graph.

This means if a node needs to change some data belonging to the scene graph (for instance the position of an avatar which is controlled by this node) it can immediately write the new and actualized data into the common scene graph without first sending the appropriate modification request to the server and then waiting until the update response will be received.
The consistency of the objects in the common scene graph is guaranteed by a transactional consistency protocol. Possible accessing conflicts, when two or more nodes wish to update the same data at the same time, are resolved completely transparently by the operating system.

Figure 24: Wissenheim interface

The availability of a consistent distributed heap storage facility also frees the programmer from the necessity of distinguishing between local and shared data. Every object will appear in the common scene graph only once and by default it is shareable.

Demand-Driven Display Technique

The demand-driven display approach of Wissenheim is the consequence of combining the direct accessibility of the common scene graph by all nodes and the transactional consistency offered by the underlying operating system. A node reads and writes the desired data from the scene graph without placing locks and without special consideration. No explicit server instance to control a certain part of the scene graph is needed and therefore no update messages are sent, broadcasted, received and interpreted just to realize perhaps that the update message was only relevant for another node. Using the demand-driven approach, a node accesses the data only when it is going to use it. In Client/Server based implementations this filtering is typically done by the Server component called area of interest management.
The shared memory concept let the nodes work without explicit replication and maintenance of local copies of the scene graph. Even during the rendering process, when a node is about to send for example the shape of an object to the graphics adapter, it accesses the data directly from the shared scene graph. The transactional consistency ensures that every access into the scene graph will always return the most actual data - no further consistency issues need to be considered by the application.
Another interesting side effect is the automatic filtering of outdated data. If for example a slower node has a lower drawing frame rate, it will always receive the newest data. If for example an avatar is moved, the slower node may not receive/fetch each avatar position, but will always get the most recent one.

2.2.11.4 Expectations and Roadmap

So far Wissenheim is not ready to run under UNIX or Kerrighed. The first step is to port Wissenheim to Linux and run it on a single node. This will require bridging the Plurix interfaces and runtime to the Linux world. For single node operation, shared memory is not required but some parts of the Plurix runtime (e.g. garbage collection and object relocation manager) will be ported and may be adapted where necessary.
We are very interested in evaluating Wissenheim using GOM (WP3.4) and other XtreemOS services in the Grid. Right now we think of separate worlds per virtual organization. Furthermore, most of the virtual world objects are normally read only, so not that critical. But of course we will learn more when studying scalability and latency issues of shared scene graphs in the XtreemOS Grid testbed. During porting to the Grid world we might also change the overall structure of Wissenheim (e.g. the Demand Driven approach) to integrate better into the Grid structure.

2.2.12 Galeb (XLAB)

2.2.12.1 Introduction

Galeb is an econometric finance modelling framework for nonlinear time series analysis and prediction. Using genetic programming, Galeb constructs an analytical model (a function) that minimizes the mean square error for a given time series. Given enough time and a time series that can be modelled by an elementary function, Galeb will find the exact solution.
The computational core is written in C++, based on the GaLib library **[Wall96]**, and exposed as a Globus Toolkit 4 Grid service **[Foster05]**. A simple distributor starts multiple independent copies on the available nodes, collects the results and selects the best constructed model among them. The application is typically time constrained, thus using more nodes can greatly improve the precision of the results.

2.2.12.2 Current Status

Galeb is currently in the development phase. As explained above, the current implementation uses a custom-design job distributor on top of the GT4 Grid service infrastructure. The security aspects of Galeb are also being developed to prevent unauthorized parties accessing input, as well as output, data **[GTK4Sec]**. It has not yet been decided whether it is also necessary to prevent them from seeing that some computation is being run and on which computers.

2.2.12.3 Detailed Overview

Modelling Core

Symbolic regression is used to solve a problem of modelling a nonlinear econometric relation (monetary exchange equation), where the structure of a model is optimized with genetic programming - a computerized random search optimization algorithm that assembles equations until it identifies the fittest one. Similar time constrained econometric modelling applications are quite common in computational finance **[Koza95]**.
The issue is deciding what type of function most appropriately fits the data, not merely computing the numerical coefficients after the type of function for the model has already been chosen. In other words, the problem is both the discovery of the correct functional form that fits the data and the discovery of the appropriate numeric coefficients. The set of functions used includes four arithmetic operators (addition, subtraction, multiplication, division) and five unary mathematical functions (log, exp, sin, cos, sqrt). The set of terminals includes the inputs appropriate to the problem domain and various constants.
The search space is a hyperspace of all possible compositions of functions that can be recursively composed of the available functions and terminals. The model should satisfy the following requirements:

- Simplicity: simpler functions (i.e. the functions with less operands) are preferred, and
- Accuracy: the sum of squared errors between predicted and measured values of the predicted variable should be minimal.

Typical running time to obtain a good solution is a few minutes. In order to speed-up the search, the user can also provide initial approximation of the target function as an input parameter to Galeb, in addition to the time series data.
Galeb engine is a wrapper class extending GaLib library, which is a freely available (even for commercial purposes), object-oriented C++ library. It implements a set of genetic algorithms and corresponding data structures, stopping criteria and selection schemes. It works on various operating systems (Windows, Unix/Linux, MacOS) and is very easy to use: the user only has to choose one of the data structures provided by GaLib, write a generator of solution candidates and a cost function for measuring accuracy of particular candidate. After the genetic algorithm is started, all the execution details are managed by the GaLib: generating new solution candidates, creating populations, performing crossover and mutation, etc.

Input and output

The Galeb time series modeller is a command-line utility that is given an input text file and returns the results in an output text file.
The time series is given in the input file as a tab-separated table, where each column corresponds to one variable and each line (except for the header) represents a sample of the data. Variables of type string, double and integer are supported, but string variables are not used in the model. Each variable also has a role, which can be “domain” (independent variable), “codomain” (the dependent variable being modelled) or “other” (not to be used in the model).

The output file contains the sum of squared error of the obtained model (first line), followed by the model itself, represented by a tree (the rest of the file). The tree consists of internal nodes (operators like addition, logarithm etc) and leaves (constants and independent variables).

Grid Service Implementation

Galeb is written in C++ as a command-line utility. The former was packaged as a library, which is called by the Grid service **[Czajkowski04]** written in Java using the Java Native Interface, as shown in the figure below. The WSDL description of the Grid service is given in the appendix.

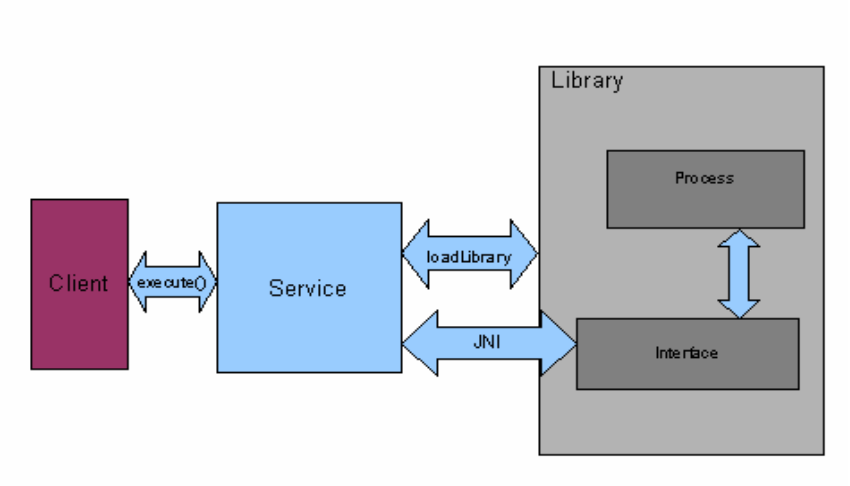


Figure 25: Galeb Software Architecture

The Grid service is stateless **[OASIS06]**. It doesn't need to store history, resource properties and a resource manager to preserve the state because all it does is pass on information for computation. The input dataset can be quite large and varies drastically, so there is no need to preserve it. All the input parameters can and should (for better results) be changed with every rerun of the algorithm, so they should not be stored in an internal state information.

Load Distribution

Genetic algorithms include making random choices and selecting the best result from the obtained result set. Therefore, the simplest distribution method is sending the same job to multiple nodes, which will almost certainly obtain different results.

For job distribution, we use a simple distributor that reads the addresses of available hosts with running service from a configuration file, enqueues each subsequent request and distributes them on a FCFS basis (see figure below). Files with input parameters are transferred via GridFTP to each host. The distributor starts a new thread for each request which then submits its data to the Grid service.

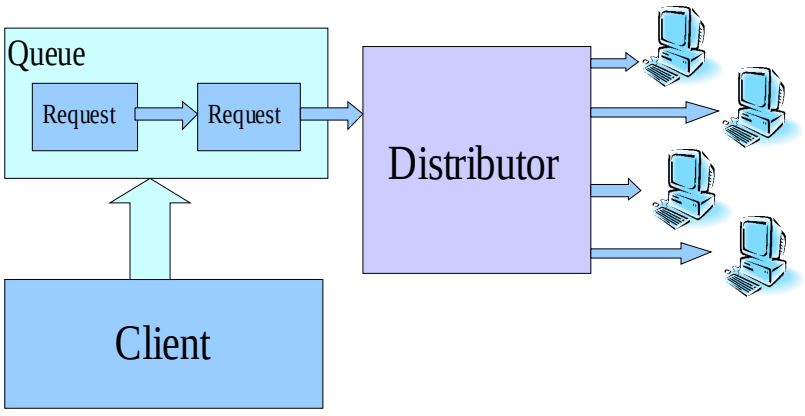


Figure 26: Galeb Load Distribution

Upon finishing the calculation, the Grid service methods return the names of files with results. Those are transferred by GridFTP back to the distributor. The distributor evaluates them and returns the best result – the one with the smallest RMSE (root mean squared error) compared to the original data.

2.2.12.4 Expectations and Roadmap

In the first implementation of Galeb for XtreemOS we will simply replace the currently used GT4 calls with the equivalent XtreemOS-API calls to produce a running application as soon as possible. Then, Galeb for XtreemOS will be modified to employ advanced XtreemOS features.

In particular, instead of using our own distribution mechanisms on top of the operating system, XtreemOS's global Grid scheduler will be employed to distribute the processes to the appropriate VO nodes automatically. Data transfer will also be ported from the existing GridFTP-based implementation to one base on GridFS.

3 Use Cases

In this section, we describe use cases illustrating to the XtreemOS developers what application users expect from this new operating system. First we introduce the methodology and the template used to express these use cases. It leads to a common and uniform way to formulate them. The template was created to be rich enough to take into account all the specifics of covered cases.

Thereafter the general and application-specific use cases themselves are presented. The latter deal with scenarios specific to each of the proposed applications. The former exemplifies the general usage of XtreemOS by illustrating typical scenarios.

The following template was used to present the use cases.

Number	use case name: describes a completable goal, should be sufficient for the end user to understand what the use case is about.
Summary	A quick overview of the use case, to understand what the use case is about. It should include the goal and principal actor.
Actors, Roles and Goals	Identifies all the actors in the scenario, their roles (i.e. what they do in the scenario) and their goals (i.e. what they want to achieve by participating in the scenario), e.g.: **User**: role of actor 1 • **Role**: the role of actor 1 • **Goal**: the goal of actor 1 ... (any other actors in the same format)
Preconditions	Conveys any conditions that must be true when a user initiates a use case. They are not however the triggers that initiate a use case.
Triggers	Describes the starting condition(s) which cause a use case to be initiated. Can be external, temporal or internal.
Basic course of events	Conveys a primary scenario, or the typical course of events, described as a set of numbered steps. **Nominal GUC Run.1 # 1 :** 1. Description of Step 1 2. Description of step 2 3. …
Alternative paths	Describes secondary paths, or alternative scenarios which are variations on the main theme. Exceptions, or what happens when things go wrong, may also be described. The alternative paths make use of the numbering of the basic course of events to show at which point they differ from the basic scenario, and if appropriate where they rejoin **Alternative GUC Run.1 # 1 :** Brief description of the alternative 1. Description of Step 1 2. Description of step 2 3. … **Alternative GUC Run.1 # 2 :** Brief description of the alternative 1. Description of Step 1 2. Description of step 2 3. … …
Postconditions	Summarizes the state of affairs after the scenario is complete.
Notes	Any important information that doesn't fit the structure of the template.
Author, date	name (institution), date

Because of the immaturity of some applications and lack of specificity of some general use cases, it was not always possible to conform to the template. Fields could thus be added or omitted on author's judgment.

3.1 General XtreemOS Use Case Scenarios

The general use cases (GUCs) serve as a methodology to present the XtreemOS, its concepts, functionality and the global vision about how XtreemOS will typically be used, what actors will be involved etc. They cover a significant part of the requirements also presented in this document, but they are not intended to summarize all the requirements. There is also no one-to-one correspondence between the GUCs and individual requirements.

The goal is also to show how the different parts of XtreemOS interact between them and to discover any troubles or conflicts in a normal execution of XtreemOS. Many technical aspects not covered by the application use cases are also incorporated in the GUCs.

In addition to the GUCs described in this document, the concepts and functionality of general Grid systems are well described by some use cases resulting from other Grid projects. Grid Economy Use Cases (http://www.Gridforum.org/Public_Comment_Docs/Documents/Sep-2005/draft-ggf-gesa-usecases-01-9.pdf) focus on the economy point of view:

- computational provider use case,
- application service provider,
- software application provider,
- brokering service provider,
- computational reseller use case economy use cases.

OGSA Use Cases (http://www.Gridforum.org/documents/GFD.29.pdf) illustrate several scenarios of using Grid systems, for example:

- commercial data center,
- severe storm modeling,
- online media and entertainment,
- Grid workflow,
- Grid resource reseller,
- virtual organization Grid portal.

Akogrimo Use Cases (http://www.akogrimo.org/download/Deliverables/D5.3.1_Final.pdf) focus on the attributes that the architecture must have, i.e. interoperability, scalability, reliability, availability, performance, and security. Several scenarios are defined that can be used for evaluating how the selected architecture copes with each attribute. These scenarios are quite similar in scope with the XtreemOS GUCs presented below. Some of them can later be used for XtreemOS evaluation, for example:

- mobile user authenticates against the VO and it is able to contact administration functions (e.g. to create a new VO member),
- a new service is available in the VO: some services could be updated and this work could affect the availability of involved services (e.g. because they need to be restarted).

3.1.1 User roles

XtreemOS by itself defines just three user roles: global Grid users, VO administrators and local resource administrators.

A **global Grid user** is a person, corporation or equivalent real-world entity who makes use of a computer system. A user in XtreemOS context is the one who accesses resources and services provided by a single node or a set of nodes which may be affiliated with a Virtual Organization. A global user does not need to have local accounts on the nodes he wishes to use.

A **VO administrator** is a global Grid user authorized to manage VO membership, policies etc. Each VO must have at least one VO administrator.

A **local resource administrator** is a local user, i.e. user with a local administrative account on a node, that is authorized to manage permissions and policies that grant or deny individual VOs or global Grid users certain types of access to the resource. Each resource must have at least one local resource administrator. The local admin does not have to be a member of any VO and need not even have a global Grid account.

XtreemOS must provide capabilities to define much **more complex role hierarchies** in individual VOs. The hierarchies can be described using hierarchical VO subgroups and VO-specific user role. Internal VO user policies are then expressed in terms of such hierarchies.

Additionally, **certificate authorities** (CA) and **real-world entities** (persons, organizations etc) are not part of an XtreemOS Grid, but are involved in creation of global Grid users.

3.1.2 Administration Use Cases

This section deals with 10 general use cases related to administration of users, VOs and resources. These are:

- GUC ADM.1: registering as a global Grid user.
- GUC ADM.2: creating a VO.
- GUC ADM.3: adding a user to a VO (similar is adding a VO admin).
- GUC ADM.4: creating a VO subgroup.
- GUC ADM.5: adding a user to a VO subgroup.
- GUC ADM.6: deleting a user from a VO (similar are deleting a user from a subgroup and deleting a VO admin. The latter must not be the only admin of the VO, because a VO cannot have zero admins).
- GUC ADM.7: changing policies of a resource.
- GUC ADM.8: setting resource access policies within VO.
- GUC ADM.9: moving or copying users from one VO to another.
- GUC ADM.10: user logging into the Grid in the context of a VO.

Note that there are still some open questions regarding some of the use cases. They are given in the “Notes” field of each detailed use case description.

The diagram below shows the relationships between the use cases and roles. Note that those use cases in the diagram that are very similar to some other use case are not mentioned in the above list nor described in detail.

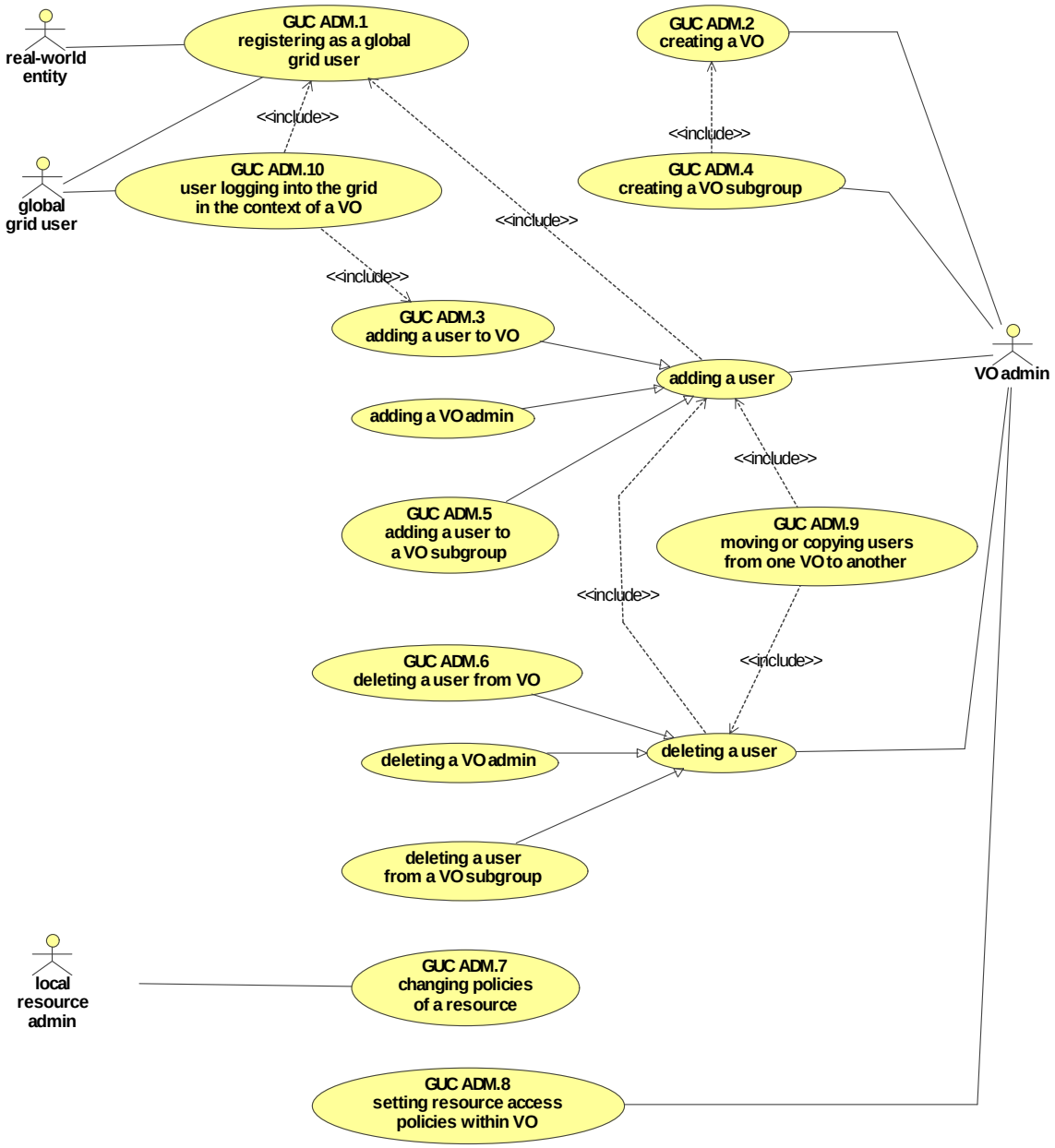


Figure 27: Overview administration use cases

GUC ADM.1	Registering as a global Grid user
Summary	A real-world entity registers to become a global Grid user, i.e. she/he acquires credentials (a certificate) that will be used to prove her/his identity on login.
Actors, Roles and Goals	**User**: a real-world entity trying to register • **Role**: real-world entity (at start), global Grid user (on success) • **Goal**: to acquire credentials of a new global Grid user **CA**: a certificate authority • **Role**: CA • **Goal**: to issue a certificate if and only if the user complies with the CA's policy
Preconditions	• The user must have a local account on the node where the certificate will be stored. • The user must select a common name (CN) used for the certificate.
Triggers	User request/command
Basic course of events	**Nominal GUC ADM.1 # 1 :** 1. The user requests a certificate from a CA following the CA's protocol. 2. The CA issues a certificate. 3. The certificate is stored on user's node. 4. The user receives success notification.
Alternative paths	**Alternative GUC ADM.1 # 1 :** CA requests the user to prove his/her identity.

	1.1. After step 1, CA requests the user to prove her/his identity. 1.2. The user proves the identity following the CA's prescribed protocol. 1.3. Continue with step 2. **Alternative GUC ADM.1 # 2 :** CA denies the request. 2. The CA denies the request (either the request contradicts the CA policy or the user failed to prove her/his identity). 3. The user receives failure notification.
Postconditions	The real-world entity possesses the certificate that allows her/him to log into XtreemOS. However, she/he cannot yet do anything (use any resource etc) after logging on, and is not a member of any VO.
Notes	This is not necessarily a part of XtreemOS, but is important for the user. XtreemOS must define the acceptable certificate formats. It must also enable VOs to define additional requirements that a certificate must satisfy to be usable as VO member credentials.
Author, date	Marjan Sterk (XLAB), November 3rd 2006.

GUC ADM.2	Creating a VO
Summary	A global Grid user creates a new VO and automatically becomes its admin.
Actors, Roles and Goals	**User**: the user creating the VO • **Role**: global Grid user (at start), VO admin (on success) • **Goal**: to become the sole administrator of a new VO
Preconditions	• The user must be logged into XtreemOS. • The user must be allowed to create new VOs. Details depend on whether the VO data is stored on a local machine, in a global VO database etc.
Triggers	User request/command
Basic course of events	**Nominal GUC ADM.2 # 1 :** 1. The user requests VO creation and provides any data necessary (VO name, the place to store VO data, the list of CAs the VO will trust etc). 2. The system checks whether the user is allowed to create the requested VO. 3. The VO data are stored in the provided place. 4. The user is registered as the only admin of the new VO. 5. The user receives success notification.
Alternative paths	**Alternative GUC ADM.2 # 1 :** user does not have permission to create the VO. 3. The system concludes that the user is not allowed to create the VO. 4. The user receives failure notification. **Alternative GUC ADM.2 # 2 :** storing VO data fails 3. Storing the VO data fails. 4. The user receives failure notification.
Postconditions	• The new VO's data are stored in the specified place. • The user is the only admin of the new VO.
Notes	An admin of a VO admin need not be its member, so the user creating a VO is not automatically made its member. Of course, being a VO admin, she/he can add her/himself to the VO members later. Allowing anyone to create a VO would open a security hole – a malicious user could create more VOs than the system can handle.
Author, date	Marjan Sterk (XLAB), November 3rd 2006.

GUC ADM.3	Adding a user to VO
Summary	A VO admin adds a global Grid user to the VO members list.
Actors, Roles and Goals	**Admin**: the VO admin making the action • **Role**: VO admin • **Goal**: to add a user to the VO
Preconditions	• The admin must be logged into XtreemOS. • The VO admin must authorize the user to join the VO.
Triggers	Admin's request/command
Basic course of events	**Nominal GUC ADM.3 # 1 :** 1. The admin provides the global user ID, ID of user's CA and any other, VO dependant data.

	2. The system checks whether the CA is trusted. 3. The user is added to the VO member list. 4. The admin receives success notification. 4.1. Optionally, the user also receives success notification.
Alternative paths	**Alternative GUC ADM.3 # 1 :** the user's CA is not trusted 2.1. The CA trust check fails. 2.2. The admin receives failure notification. 2.3. Optionally, the user also receives failure notification. **Alternative GUC ADM.3 # 2 :** storing VO data fails 3. Storing the VO data fails. 4. The admin receives failure notification. 4.1. Optionally, the user also receives failure notification. **Alternative GUC ADM.3 # 3 :** the user is already a member of VO 3. The admin receives notification.
Postconditions	• The user's ID and other data is stored in the VO database. • Once logged into XtreemOS, the user can use any resources that are accessible to all members of the VO.
Notes	Adding a new VO admin to the VO follows the same steps. The global user ID is probably the CN of the user certificate.
Author, date	Marjan Sterk (XLAB), November 3rd 2006.

GUC ADM.4	Creating a VO subgroup
Summary	A VO admin creates an (initially empty) subgroup of VO users.
Actors, Roles and Goals	**User**: the VO admin creating the subgroup • **Role**: VO admin • **Goal**: to create a new, initially empty VO subgroup
Preconditions	The admin must be logged into XtreemOS.
Triggers	Admin's request/command
Basic course of events	**Nominal GUC ADM.4 # 1 :** 1. The admin requests VO subgroup creation and provides any data necessary (subgroup name etc). • If the new subgroup is to be a sub-subgroup of an existing subgroup, the name of the latter must also be provided by the admin. 2. The subgroup is created and its data stored together with the VO data. 3. The admin receives success notification.
Alternative paths	**Alternative GUC ADM.2 # 1 :** storing VO data fails 2. Storing the VO data fails. 3. The admin receives failure notification.
Postconditions	• The new VO subgroup data are stored in the specified place. • The subgroup contains no users.
Notes	It remains to be defined whether a subgroup can also have its own admin that is not a VO admin.
Author, date	Marjan Sterk (XLAB), November 6th 2006.

GUC ADM.5	Adding a user to VO subgroup
Summary	A VO admin adds a VO user to the subgroup members list.
Actors, Roles and Goals	**Admin**: the VO admin making the action • **Role**: VO admin • **Goal**: to add a user to the subgroup
Preconditions	• The admin must be logged into XtreemOS. • The user must be a member of the VO. • The VO admin must authorize the user to join the subgroup.
Triggers	Admin's request/command
Basic course of events	**Nominal GUC ADM.3 # 1 :** 1. The admin provides the subgroup name/ID, global user ID and potentially any other data. 2. The system checks whether the user is a member of the VO.

	3. The user is added to the subgroup member list. 4. The admin receives success notification. 4.1. Optionally, the user also receives success notification.
Alternative paths	**Alternative GUC ADM.3 # 1 :** the user is not a member of the VO 2.1. The VO membership check fails. 2.2. The admin receives failure notification. 2.3. Optionally, the user also receives failure notification. **Alternative GUC ADM.3 # 3 :** the user is already a member of the subgroup 3. The admin receives notification.
Postconditions	• The user's ID and other data is stored in the subgroup member list. • Once logged into XtreemOS, the user can use any resources that are accessible to all the members of the subgroup.
Notes	
Author, date	Marjan Sterk (XLAB), November 6th 2006.

GUC ADM.6	Deleting a user from a VO
Summary	A VO admin deletes a user from the VO members list.
Actors, Roles and Goals	**Admin**: the VO admin making the action • **Role**: VO admin • **Goal**: to delete the user from the subgroup
Preconditions	• The admin must be logged into XtreemOS. • The user must be a member of the VO.
Triggers	Admin's request/command
Basic course of events	**Nominal GUC ADM.3 # 1 :** 1. The admin provides the global user ID. 2. The system checks whether the user is a member of the VO. 3. All user's applications are terminated. 4. The user is removed from the VO member list. 5. The admin receives success notification. 4.1.Optionally, the user also receives notification.
Alternative paths	**Alternative GUC ADM.3 # 1 :** the user is not a member of the VO 2.1. The VO membership check fails. 2.2. The admin receives failure notification. 2.3. Optionally, the user also receives failure notification.
Postconditions	• No applications are running any more on behalf of the deleted user. • The user's ID and other data is removed from the subgroup member list. • The user cannot start new applications in the context of the VO he was deleted from. He also cannot use any the resources on behalf of this VO.
Notes	Termination of an application can be avoided by changing the owner of the application before deleting the user. Obviously, the new owner must belong to the same VO and must be authorized to run the application and use any necessary resources. Similar to GUC ADM.6 are also: • deleting a user from a subgroup (precondition: user must be a member of the sub-group; postcondition: user is still a member of the VO, but not of the subgroup), • deleting a VO admin (precondition: the user being deleted must be a VO admin, BUT not its only admin (VO cannot have zero admins); postcondition: the user is no more a VO admin (but is still a VO member if he was a member before)). It remains to be defined whether a user can also delete her/himself from a VO.
Author, date	Marjan Sterk (XLAB), November 6th 2006.

GUC ADM.7	Changing policies of a resource
Summary	The admin of a local resource changes the access policies for VOs, VO subgroups or individual users.
Actors, Roles and Goals	**Local admin**: the local resource's admin performing the action • **Role**: local admin • **Goal**: change access policies to grant or deny certain types of access to VOs, VO

	subgroups or individual users
Preconditions	• A relation of trust exists between the VO admin and the resource owner/local admin (this trust can be based on SLAs, personal friendship...- it is out of the scope of the use case). • The local admin must know which individual users, VOs or subgroups are using his resources.
Triggers	Local admin's request/command.
Basic course of events	**Nominal GUC ADM.7 # 1:** 1. The local admin selects the local resource. 2. The local admin provides the necessary pieces of information to identify the subject (VO, individual user) to which new policies will apply. 3. Optionally, VO admin must validate the inclusion/change of the new resource (this validation can be part of the VO policies) 4. The local admin changes access policies. 5. The local admin receives success notification and changes take effect. Depending on the resource configuration and the VO notificaction policy, the user (VO, VO subgroup or individual user) may also receive a notification of the policy changes.
Alternative paths	**Alternative GUC ADM.7 # 1:** failure of access policy changes 4. The local admin receives failure notification and changes don't take effect. No user receives failure notification.
Postconditions	Next time the user accesses this resource, new policy will apply.
Notes	If the management interface is a Command Line Interface (CLI), steps 1-3 can be merged into a single step.
Author, date	TID team, November 7th 2006.

GUC ADM.8	Setting resource access policies within a VO
Summary	A VO admin limits the access rights of individual users or subgroups.
Actors, Roles and Goals	**Admin**: the VO admin performing the action • **Role**: VO admin • **Goal**: to limit the access rights of individual users or subgroups
Preconditions	The admin must be logged into XtreemOS.
Triggers	Admin's request/command.
Basic course of events	**Nominal GUC ADM.8 # 1:** 1. The VO admin selects a resource (or group of resources) within the VO. 2. The VO admin provides necessary data to identify the user/subgroup (user ID, sub-group name, etc.). 3. The admin changes the access rights of the user or subgroup with respect to the re-source. 4. The admin receives success notification and settings take effect. Depending on the VO notification policy, the user or subgroup may also receive a notification of the changes concerning him.
Alternative paths	**Alternative GUC ADM.8 # 1:** failure of access policy set 4. The admin receives failure notification and settings don't take effect. No user receives failure notification.
Postconditions	Next time the user accesses this resource, new policy will apply.
Notes	If the management interface is a Command Line Interface (CLI), steps 1-3 can be merged into a single step.
Author, date	TID team, November 7th 2006.

GUC ADM.9	Moving or copying users from one VO to another
Summary	A VO admin wants to move or copy users between two VOs.
Actors, Roles and Goals	**Admin**: the VO admin performing the action. • **Role**: VO admin • **Goal**: to copy or move users between two VOs **User**: the user being moved or copied • **Role**: VO user • **Goal**: to belong to both VOs that share the same admin (copy) or pass from one VO to another one (move).
Preconditions	• The admin must be logged into XtreemOS. • The VO admin must authorize the user to join the destination VO.

	• The VO admin is a VO admin of both involved VOs. • The user must be a member of the source VO.
Triggers	Admin's request/command
Basic course of events	**Nominal GUC ADM.9 # 1 :** 1. The system checks whether the admin has proper credentials in both involved VOs. 2. The system checks whether the user CA is trusted by the destination VO. 3. The system copies or moves the global Grid user from the source VO members list to the destination VO membership list. 4. The admin receives a success notification.
Alternative paths	**Alternative GUC ADM.9 # 1 :** the admin is not an admin in both VOs 4. The admin receives a failure notification.
Postconditions	The user can log into the destination VO and access resources associated to that VO, according to an associated policy.
Notes	Apart from membership lists, other information could be copied/moved from the source VO to the destination VO.
Author, date	TID team, November 7th 2006.

GUC ADM.10	User logging into the Grid in the context of a VO
Summary	A Grid user tries to log into a particular VO.
Actors, Roles and Goals	**User**: Grid user trying to access a VO • **Role**: global Grid user • **Goal**: to access to a VO
Preconditions	• User is a global Grid user with a certificate acquired by GUC ADM.1. • VO exists. • User is a VO member of the VO she/he is trying to log into.
Triggers	User request/command. User specifies the VO she/he wants to log into.
Basic course of events	**Nominal GUC ADM.10 # 1 :** 1. The system checks the validity of the user's certificate. 2. The system checks whether the User's Global UID is in the VO membership list. 3. The user receives success notification. (Probably including user credentials to access the VO)
Alternative paths	**Alternative GUC ADM.10 # 1 :** the user certificate is not valid (e.g. outdated) 2. The user receives failure notification. **Alternative GUC ADM.10 # 1 :** the user is not a member of the VO 3. The user receives failure notification.
Postconditions	If the login process succeeded, the user has access to VO resources, according to the VO policy. The system can be configured so that a GridFS volume automatically becomes a home directory.
Notes	The global Grid user in this GUC has to specify the VO she/he wants to log into. There would be a special case, when (and if) the user certificate includes information about a principal or main VO (in the same way as linux users belong to several groups but have a main group). In that hypothetical case, the user wouldn't have to indicate a VO if she/he wants to log into the main VO.
Author, date	TID team, November 7th 2006.

3.1.3 Running Application Use Cases

This section deals with 5 general use cases related to running application on XTREEMOS. These are:

- GUC RUN.1: submitting a job.
- GUC RUN.2: monitoring a job.
- GUC RUN.3: cancelling a job on user request.
- GUC RUN.4: submitting, monitoring, cancelling a workflow.
- GUC RUN.5: using virtual robust nodes implemented as replicated nodes.

Note that the workflow manager will not be a part of XtreemOS, but rather a third-party tool on top of XtreemOS.

The diagram below shows the relationships between the use cases and roles. Note that those use cases in the diagram that are very similar to some other use case are not mentioned in the above list nor described in detail.

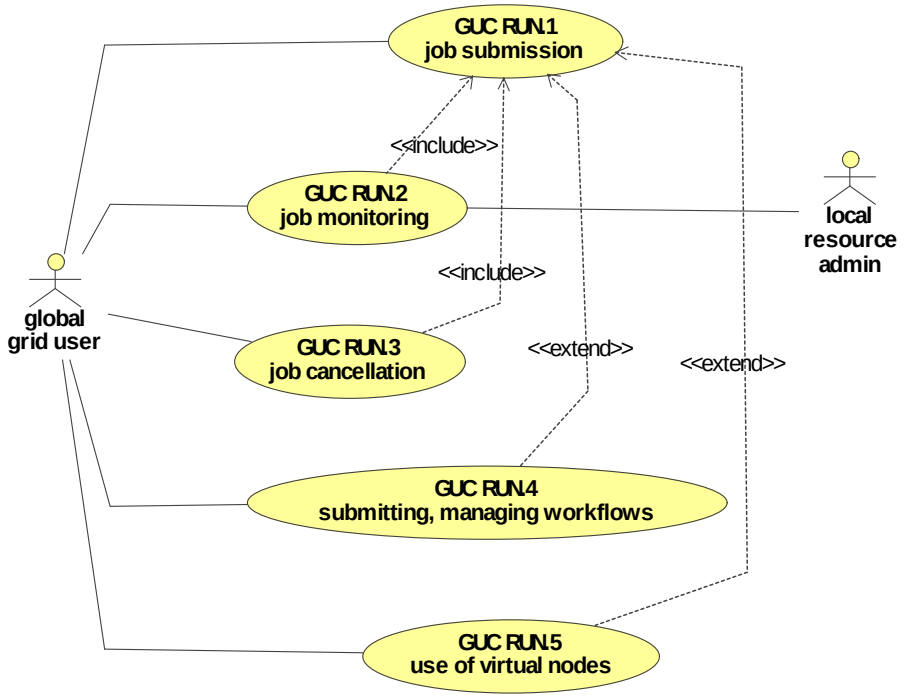


Figure 28: Overview use cases for running applications

GUC RUN.1	Job Submission
Summary	How to execute an application on XtreemOS .
Actors, Roles and Goals	**The end user** submitting the job**:** • **Role**: global Grid user • **Goal**: wants to run its application on the system
Preconditions	• The application to be submitted is ready to be run • Its parameters and requirements in term of resources are clearly identified. • All input files reside on the GridFS. • The application has been "packaged" for XtreemOS in order to be submitted. In a "job card", all information related to the application have been precisely formulated. For example : o The path to the application executable o The specific requirements of the application in terms of type of CPU supported, expected bandwidth, amount of disk needed, geographical restriction, maximum and minimum number of nodes required, etc o The eventual need to run in an MPI-compatible environment as well as precising further some typical MPI characteristics : ▪ The number of nodes required, as a number or a range ▪ The need to have the nodes picked out in a set of resource connected with high bandwidth network (on a single cluster for example) ▪ The need to have access to a parallel high performance file system.… o The procedure chosen for notification of the end of the job or any other issue : by mail and/or text reporting in a separate file o Optionally the expected duration or memory consumption of the Job o The need to have a transparent checkpoint/restart service o The ability for the application to be migrated during its run
Triggers	A job can be submitted either by an end user, either by another application (in particular a workflow manager presented in GUC.RUN.4)
Basic course of events	**Nominal GUC RUN.1 # 1 :** 1. The submitter is authenticated by the system 2. He submits his jobs to XtreemOS 3. He is notified of the end of his jobs (by the job submission utility) 4. He checks the output results of the file

Alternative paths	**Alternative GUC RUN.1 # 1 :** Job rejected. 1. Job is rejected by XtreemOS because the submitter has not the rights to submit such an application or to use such resources. 2. The submitter is notified with a clear message stating clearly the cause of the rejection of his query. **Alternative GUC RUN.1 # 2 :** Job delayed 1. Not enough resources are available to run properly the application. 2. The submitter is notified with a clear message stating clearly the cause of the delayed submission. 3. He may choose to cancel his job or to wait until available resources match the ones required. **Alternative GUC RUN.1 # 3 :** Job consuming more resources than allowed 1. During application runtime the resource usage overpasses the allowed usage. 2. The submitter is notified with a message clearly stating the potential menace for the application of an eventual shortage of resources not in line with the ones that has been demanded. 3. He may choose to go on, to cancel its job or to ask for the migration of the job asking this time for better dimensioned resource. **Alternative GUC RUN.1 # 4 :** Job experiencing a resource failure. 1. Application runs fine but one of the resources it uses fails. 2. The submitter is notified with a message clearly stating the crash that occurred and the consequence for the application. He may choose to cancel his job. 3. If it has been declared so in the job card, the application is automatically checkpointed/restarted and rescheduled.
Postconditions	• The application is not using resources of XtreemOS anymore. • Eventually output file(s) was (were) produced and stored on GridFS.
Notes	
Author, date	Samuel KORTAS (EDF), November 2nd 2006.

GUC RUN.2	Job Monitoring
Summary	How to monitor an application running on XtreemOS .
Actors, Roles and Goals	**End user** that wants to monitor jobs: • **Role**: global Grid user • **Goal**: wants to get information about a running application, either his own or another application he is allowed to monitor. **Local admin** that authorizes and prevents XtreemOS to access a given resource: • **Role**: local resource admin • **Goal**: wants to get information about the use of the resource made available to at least one VO
Preconditions	• The application has been submitted and is scheduled to run, is currently running, or has just run on XtreemOS.
Triggers	A job can be monitored either by an end user, either by another application (in particular a workflow manager presented in GUC.RUN.4)
Basic course of events	**Nominal GUC RUN.2 # 1 :** Job monitored by end user 1. The user is authenticated by the system. 2. He asks XtreemOS for a list of running jobs that he is allowed to monitor, which depends on VO policies and whether he is a VO admin. 3. XtreemOS provides the information tagging each job with an ID, owner and current status : scheduled/running/stopped/migrating/complete. 4. Using the job ID, the submitter can access additional information : o Estimate of the time of the beginning of the run if the job is in queue. o Remaining time before its end.

	o Detailed resource used by the job (CPU Time, File System and bandwidth use, number and IDs of nodes allocated, RAM used…) o Date of important Events for the job : time of submission, beginning of the run, eventual failure of resources, eventual migration, time of the end of the job or its cancellation. 5. These information stay available as long as the job is running on XtreemOS. At the end of the run, they are gathered in a file transmitted to the submitter. **Nominal GUC RUN.2 # 2 :** Job monitored by the local admin 1. The local admin is authenticated by the system. 2. He asks XtreemOS for a list of the jobs using a given resource he agreed to make available to at least one VO. 3. XtreemOS provides the list information tagging each job with an ID and its current status: scheduled/running/stopped/migrating/complete. 4. Using the job ID, and the resource ID, the local admin can access additional information : o Estimate of the time of the beginning of the run if the job is in queue o Detailed resource currently used by the job on the considered resource (CPU Time, File System space use, number of nodes allocated, RAM used), remaining time before its end. o Eventually capability of the job to migrate or restart after a checkpoint. o Eventually bandwidth used in the case of an MPI Application. The same kind of information can be obtained regarding the whole use of the Resource. Knowing these information helps the local admin when he decides to stop sharing the resources with XtreemOS VOs. If a job is scheduled to end in a few seconds, he may be inclined to wait till that event.
Alternative paths	**Alternative GUC RUN.2 # 1 :** Permission denied. 1. For a given reason, the local admin or the end user have no access to that kind of information : they are informed by the System.
Postconditions	
Notes	
Author, date	Samuel KORTAS (EDF), November 3rd 2006.

GUC RUN.3	Job Cancellation
Summary	How to cancel an application running on XtreemOS .
Actors, Roles and Goals	**End user** on XtreemOS: • **Role**: global Grid user • **Goal**: wants to cancel an application running on the system
Preconditions	• The application has been submitted and is scheduled to run, is currently running, or has just run on XtreemOS.
Triggers	A job can be cancelled either by an end user, either by another application (in particular a workflow manager presented in GUC.RUN.4).
Basic course of events	**Nominal GUC RUN.3 # 1 :** 1. The user is authenticated by the system 2. He asks XtreemOS for a list of running jobs he is allowed to cancel, which depends on VO policies and whether he is a VO admin. 3. XtreemOS provides the information tagging each job with an ID and its current status : scheduled/running/stopped/migrating/complete a. Using the job ID, the user posts a cancellation request of this job. 4. Job is cancelled and all the resources it used are released. 5. The user is notified of the end of his job 6. He receives the eventual output results as well as a cancellation report
Alternative paths	**Alternative GUC RUN.3 # 1 :** Permission denied. 1. For a given reason, the user has no the privilege to cancel the job: he is informed so by the System.
Postconditions	
Notes	
Author, date	Samuel KORTAS (EDF), November 3rd 2006.

GUC RUN.4	Submitting, Managing workflows
Summary	How to launch, monitor and cancel a workflow on XtreemOS .
Actors, Roles and Goals	**End user** of the workflow: • **Role**: global Grid user • **Goal**: wants to manage a workflow on XtreemOS **Workflow Manager** (not actually an actor but rather an application) • **Role:** middleware application that handles workflows • **Goal:** give a transparent control of a workflow to the user
Preconditions	• The application has been divided into several jobs to be run on XtreemOS. • Their interaction and coupling is described by a workflow. • There is a workflow manager in XtreemOS.
Triggers	A workflow can be submitted either by an end user, either by another application.
Basic course of events	**Nominal GUC RUN.4 # 1 :** run of a workflow on XtreemOS 1. The submitter is authenticated by the system. 2. He submits to the Workflow Manager his workflow. 3. The workflow manager submits to XtreemOS each job involved in the workflow according to the plan settled by the end user (see GUC RUN.1). 4. The workflow manager continuously interacts with XtreemOS, waiting for the end of running jobs, processing their results and submitting other jobs according to the workflow defined by the user. 5. The submitter is notified of the end of his workflow. 6. He receives the eventual output results as well as a running report.
Alternative paths	**Alternative GUC RUN.4 # 1 :** the end user monitors the workflow 1. A workflow has been submitted and is running as described in the nominal case. 2. The user asks the Workflow Manager information about the workflow currently running. 3. The Workflow Manager automatically asks XtreemOS information about running jobs (see GUC RUN.3) and consolidates along with the information it can gather about non running job in the workflow. 4. It presents the information consolidated to the end user and gives him a global vision of the workflow. **Alternative GUC RUN.4 # 2 :** the end user cancels the workflow 1. A workflow has been submitted and is running as described in the nominal case. 2. The user asks the Workflow Manager to cancel the workflow currently running. 3. The Workflow Manager automatically forwards the cancellation request to XtreemOS for currently running jobs (see GUC RUN.3), expects an acknowledgement from their cancellation and notifies the end User of the effective cancellation of the whole Workflow **Alternative GUC RUN.4 # 3 :** Permission denied. 1. For a given reason, the Work Flow Manager has no privilege to run a job. 2. It notifies the end user of the encountered problem 3. It eventually cancels the remaining running jobs that were part of the workflow
Postconditions	
Notes	The workflow manager is not a part of XtreemOS, but rather a third-party tool on top of XtreemOS.
Author, date	Samuel KORTAS (EDF), November 3rd 2006.

GUC RUN.5	Use of virtual nodes
Summary	How an application uses redundant nodes seen virtually as one in order to limit the consequences of a node failure on application that does not provide a checkpoint/restart feature.
Actors, Roles and Goals	**The end user** submitting the job: • **Role**: global Grid user • **Goal**: wants to runs its application on XtreemOS
Preconditions	All the preconditions of GUC RUN.1. Additionally, it must be ensured that the application does not use any resources that would be shared among the two (or more) instances that are run with the same input.
Triggers	A job can be submitted either by an end user, either by another application (in particular a workflow manager presented in GUC.RUN.4)
Basic course of events	**Nominal GUC RUN.5 # 1 :** run of an application using virtual nodes on XtreemOS 1. The submitter is authenticated by the system. 2. He submits his jobs to XtreemOS. 3. Process or services requiring redundancy are running on virtual nodes. This is transparent for the application. 4. Once the application has ended, the End user is notified of the end of his jobs. 5. He checks the output results of the file
Alternative paths	**Alternative GUC RUN.5 # 1 :** Virtual nodes are not available. 1. No virtual resources are available to run properly the application. 2. The submitter is notified with a clear message stating clearly the cause of the delayed submission. 3. He may choose to run on regular nodes, to cancel his job or to wait until available resource matches the ones required. **Alternative GUC RUN.5 # 1 :** Job experiencing a resource failure. 1. Application runs fine but one of the resources its uses fails 2. If it concerns a virtual node, it has no impact for the application. The submitter is notified with a message warning that some problem occurred but was solved using the replication capability of virtual nodes. As soon as possible, additional resource is added to the virtual node in order to keep the same level of redundancy. 3. If another regular node is concerned, the submitter is notified with a message clearly stating the crash that occurred and the consequence for the application. He may choose to cancel his job. If it has been declared so in the job card, the application is automatically checkpointed/restarted and rescheduled.
Postconditions	• The application is not using resources of XtreemOS anymore. • Eventually output file(s) was (were) produced and delivered to the submitter.
Notes	
Author, date	Samuel KORTAS (EDF), November 3rd 2006.

3.1.4 GridFS-related Use Cases

This section deals with 10 general use cases related to file management on the GridFS. These are:

- GUC GFS.1: creating a directory on GridFS.
- GUC GFS.2: copying files.
- GUC GFS.3: deleting files.
- GUC GFS.4: managing concurrent access to files.
- GUC GFS.5: changing access permissions.
- GUC GFS.6: changing "distributed-ness" and replication parameters of files.
- GUC GFS.7: monitoring.
    - Monitoring performance aspects: number of files transferred back and forth, getting information about bandwidth loads...
    - monitoring other aspects: file numbers and sizes, free space, replication parameters...
- GUC GFS.8: exporting a file or directory.
- GUC GFS.9: un-exporting a file or directory.

In general, all basic POSIX system calls and commands related to file and directory management should work as is on GridFS mounted volumes. It has not yet been decided whether full POSIX ACLs will be supported. The last two use cases are not planned to be supported by the core GridFS services, but they can be implemented on top of the latter.

The diagram below shows the relationships between the use cases and roles. Note that those use cases in the diagram that are very similar to some other use case are not mentioned in the above list nor described in detail.

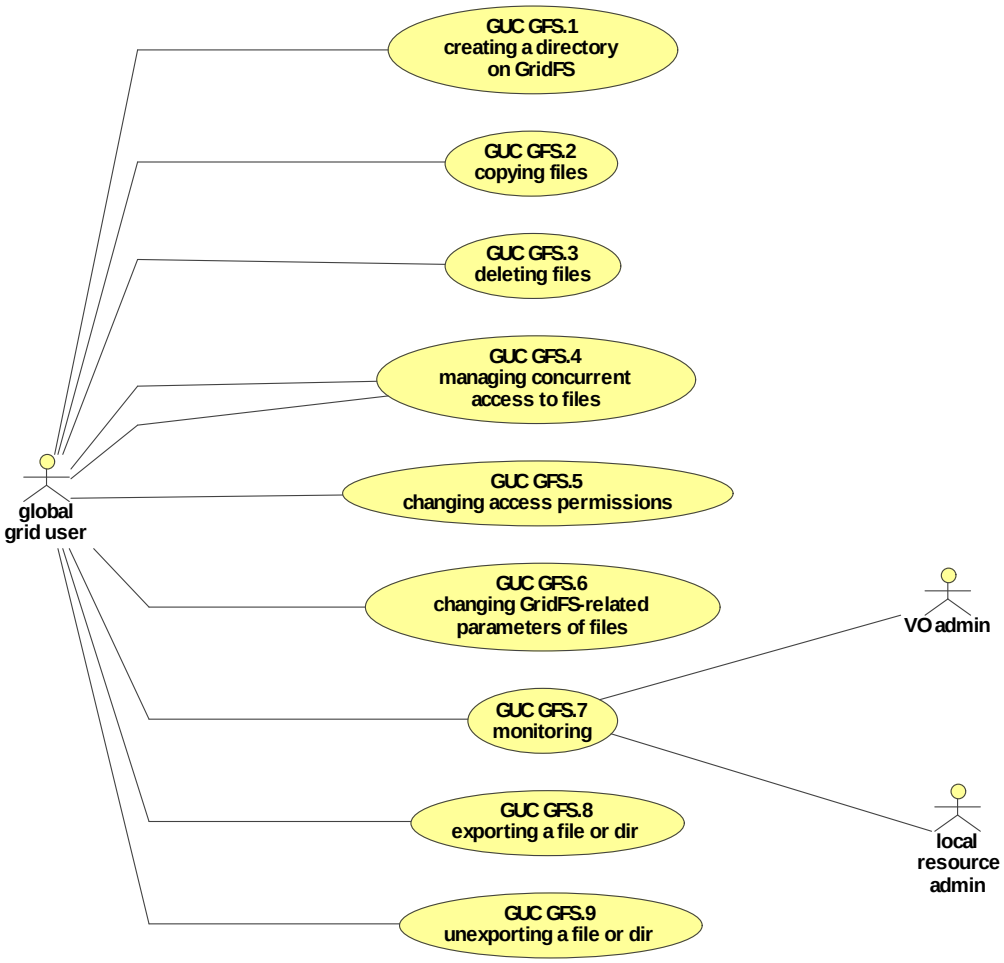


Figure 29: Overview of GridFS use cases

GUC GFS.1	Creating a directory on GridFS
Summary	Creating a directory inside a GridFS volume.
Actors, Roles and Goals	**User**: a global Grid user • **Role**: a global Grid user • **Goal**: to manage his or her GridFS files and directories **Application**: an application • **Role**: an application running under the credentials of a global Grid user • **Goal**: to manage the user's files and directories
Preconditions	• The user must have initiated a Grid session. • The system has the corresponding GridFS volume mounted on the host.
Triggers	1. User request/command or system call
Basic course of events	**Nominal GUC GFS.1 # 1 :** 1. The user requests the creation of a directory inside a GridFS volume by invoking the /bin/mkdir command and specifying the path of the directory as its parameter. 2. The operating system creates the specified directory on the GridFS. 3. The command returns successfully. **Nominal GUC GFS.1 # 2 :** 1. The application requests the creation of a directory inside a GridFS volume by invok-

	ing the mkdir system call and specifying the path of the directory as its parameter. 2. The operating system creates the specified directory on the GridFS. 3. The command returns successfully.
Alternative paths	**Alternative GUC GFS.1 # 1 :** The system call fails. 2. The operating system fails to create the directory (bad permissions, bad path, ...). 3. The mkdir system call fails and sets errno accordingly. 4. The /bin/mkdir command fails and shows the corresponding error message. **Alternative GUC GFS.1 # 2 :** The system call fails. 2. The operating system fails to create the directory (bad permissions, bad path, ...). 3. The mkdir system call fails and sets errno accordingly.
Postconditions	The directory specified in step 1 has been created.
Notes	In general, all basic POSIX system calls and commands related to file and directory management should work as is on GridFS mounted volumes. This implies, for example, that the creating user is the owner of the newly created directory.
Author, date	Josep M. Perez (BSC), November 6th 2006.

GUC GFS.2	Copying a file
Summary	Copying a file to or from a GridFS volume.
Actors, Roles and Goals	**User**: a global Grid user • **Role**: a global Grid user • **Goal**: to manage his or her GridFS files and directories **Application**: an application • **Role**: an application running under the credentials of a global Grid user • **Goal**: to manage the user's files and directories
Preconditions	• The user must have initiated a Grid session. • The system has the corresponding GridFS volume mounted on the host.
Triggers	2. User request/command
Basic course of events	**Nominal GUC GFS.2 # 1 :** 1. The user requests copying a file from or to a GridFS volume by invoking the /bin/cp command and specifying the path of the source file and the path of the target directory or file as its parameters. 2. The operating system copies the file following the POSIX conventions. 3. The command returns successfully. **Nominal GUC GFS.2 # 2 :** 1. The application requests opening the input file by calling the standard POSIX open system call with the proper parameters. 2. The operating system returns a file descriptor for the input file. 3. The application requests opening the output file by calling the standard POSIX open system call with the proper parameters. 4. The operating system returns a file descriptor for the output file. 5. The application reads a chunk of the input file by invoking the read system call. 6. The operating system returns the size of the data read. 1. If the read operation failed, then the application continues at step 10. 7. The application writes the chunk into the output file by invoking the write system call. 8. The operating system returns the size of the chunk that has been written. 9. The application goes back to step 5. 10. The application closes the output file by invoking the close system call on the output file descriptor. 11. The system call returns with a success value. 12. The application closes the input and output files by invoking the close system call on the input and output file descriptors. 13. The system call returns with a success value.
Alternative paths	**Alternative GUC GFS.2 # 1 :** The system call fails. 2. The operating system fails to open the file (bad permissions, bad source path, bad target path, ...). 3. The standard /bin/cp UNIX command fails with the corresponding error message. **Alternative GUC GFS.2 # 2 :** The open system call fails. 2/4. The operating system fails to open the file (bad permissions, bad source path, bad tar-

	get path, ...). 3/5. The open system call returns with error and sets errno accordingly.
Postconditions	The target file specified in step 1 has been created in the target path with the same contents as the source file and permissions as specified by POSIX.
Notes	In general, all basic POSIX system calls and commands related to file and directory management should work as is on GridFS mounted volumes.
Author, date	Josep M. Perez (BSC), November 6th 2006.

GUC GFS.3	Deleting a file
Summary	Deleting a file from a GridFS volume.
Actors, Roles and Goals	**User**: a global Grid user • **Role**: a global Grid user • **Goal**: to manage his or her GridFS files and directories **Application**: an application • **Role**: an application running under the credentials of a global Grid user • **Goal**: to manage the user's files and directories
Preconditions	• The user must have initiated a Grid session. • The system has the corresponding GridFS volume mounted on the host.
Triggers	3. User request/command
Basic course of events	**Nominal GUC GFS.3 # 1 :** 1. The user requests removing a file from a GridFS volume by invoking the /bin/rm command and specifying the path of the file as its parameter. 2. The operating system erases the file from the file system. 3. The command returns successfully. **Nominal GUC GFS.3 # 2 :** 1. The application requests removing a file from a GridFS volume by invoking the unlink system call and specifying the path of the file as its parameter. 2. The operating system erases the file from the file system and returns successfully.
Alternative paths	**Alternative GUC GFS.3 # 1 :** The system call fails. 1. The operating system fails to remove the file (bad permissions, bad path, ...). 2. The standard /bin/cp UNIX command fails with the corresponding error message. **Alternative GUC GFS.3 # 1 :** The system call fails. 2. The operating system fails to remove the file (bad permissions, bad path, ...). 3. The operating system returns a failure value and sets errno accordingly.
Postconditions	The target file specified in step 1 has been removed from the file system.
Notes	In general, all basic POSIX system calls and commands related to file and directory management should work as is on GridFS mounted volumes.
Author, date	Josep M. Perez (BSC), November 6th 2006.

GUC GFS.4	Concurrent access to files
Summary	This use case shows the effects of concurrent access to a file.
Actors, Roles and Goals	**Application1**: an application • **Role**: an application running under the credentials of a global Grid user • **Goal**: to manage the user's files and directories **Application2**: an application • **Role**: an application running under the credentials of a global Grid user • **Goal**: to manage the user's files and directories
Preconditions	• The user must have initiated a Grid session. • The file F contains the data D0. • The system has the corresponding GridFS volume mounted on the host.
Triggers	4. User request/command
Basic course of events	**Nominal GUC GFS.4 # 1 :** 1. Application1 opens the F file for reading by using the open system call. 2. The operating system returns a file descriptor for the file. 3. Application2 opens the same file for writing by invoking the open system call. 4. The system call returns a file descriptor for that file. 5. Application2 writes to F the data D1 by using the write system call. 6. The system call returns successfully.

	7. Application1 reads from F by invoking the read system call. 8. The system call returns successfully and retrieves D1. 9. Application1 closes F by invoking the close system call. 10. The system call returns successfully. 11. Application2 closes F by invoking the close system call. 12. The system call returns successfully.
Alternative paths	**Alternative GUC GFS.4 # 1 :** Non POSIX writes. 3. Application2 opens the F file for writing by invoking the open system call but specifies an open flag that indicates relaxed POSIX compliance. 8. The system call returns successfully and retrieves D1 or D0. **Alternative GUC GFS.4 # 2 :** The open system call fails. 1/3. The operating system fails to open the file (bad permissions, bad source path, bad target path, ...) 2/4. The open system call returns with error and sets errno accordingly.
Postconditions	File F contains D1.
Notes	In general, all basic POSIX system calls and commands related to file and directory management should work as is on GridFS mounted volumes.
Author, date	Josep M. Perez (BSC), November 6th 2006.

GUC GFS.5	Changing file or directory access permissions
Summary	Changing the file and directory Access Control Lists on a GridFS volume.
Actors, Roles and Goals	**User**: a global Grid user • **Role**: a global Grid user • **Goal**: to manage his or her GridFS files and directories **Application**: an application • **Role**: an application running under the credentials of a global Grid user • **Goal**: to manage the user's files and directories
Preconditions	• The user must have initiated a Grid session. • The file must exist. • The user must have the proper file permissions. • The system has the corresponding GridFS volume mounted on the host.
Triggers	5. User request/command
Basic course of events	**Nominal GUC GFS.5 # 1 :** 1. The user requests changing the ACL's of a file or a directory by invoking the /bin/setfacl command from POSIX 1003.2c and specifying the path of the file/directory and the ACLs as its parameters. 2. The operating system performs the specified ACL changes on the file or directory of the GridFS. 3. The command returns successfully. **Nominal GUC GFS.5 # 2 :** 1. The application requests changing the ACL's of a file or a directory by invoking a combination of the ACL system calls from POSIX 1003.1e on the path of the file or directory. 2. The operating system performs the specified ACL changes on the file or directory of the GridFS and returns successfully for every system call.
Alternative paths	**Alternative GUC GFS.5 # 1 :** System call failure 2. The operating system fails on a system call (bad permissions, bad path, ...) 3. The system call returns with error and sets errno accordingly.
Postconditions	The file or directory has had the ACLs updated as specified by POSIX 1003.1e.
Notes	In general, all basic POSIX system calls and commands related to file and directory management should work as is on GridFS mounted volumes.
Author, date	Josep M. Perez (BSC), November 6th 2006.

GUC GFS.6	Changing "distributed-ness" and replication parameters of files
Summary	An application might want to change the distribution parameters and striping policies of files in the file system to meet specific security criteria or to increase performance.
Actors, Roles and Goals	**User** that is using the file system: • **Role**: global Grid user

	• **Goal**: change the distribution parameter and striping policies **Grid** that is providing the file system: • **Role**: Grid • **Goal**: distributing and striping files over the Grid, providing transparent access
Preconditions	GridFS is used.
Triggers	User process actively wants to change parameters.
Basic course of events	**Nominal GUC GFS.6 # 1 :** 1. User process asks FS to change specific file parameters like distribution or striping. 2. FS acknowledges or denies request. **Nominal GUC GFS.6 # 2 :** 1. User process asks FS about the current parameters of a file. 2. FS returns the current policies of the file. **Nominal GUC GFS.6 # 3 :** 1. User process asks FS to change specific file parameters like distribution or striping. 2. FS acknowledges or sends a list of possible parameters User process changes request to meet the possible parameters.
Alternative paths	**Alternative GUC GFS.6 # 1 :** user not allowed to change the parameters or parameter values invalid 1. FS informs the user about the error.
Postconditions	
Notes	
Author, date	Michael Sonnenfroh (Uni Düsseldorf), 7.11.2006

GUC GFS.7	Monitoring file system
Summary	A user might want to monitor the file system and the performance of file system deamons by many aspects: amount of data transferred, % of max bandwidth usage, number of files, space disk used, free disk space, replica parameters, … This data may be provided has either instantaneous value or as mean values over some predefined period (last 5 or 15 minutes)
Actors, Roles and Goals	**User 1** that is using the file system: • **Role**: global Grid user • **Goal**: request some information about its file system usage **User 2** that is providing part of the file system: • **Role**: local provider • **Goal**: monitor its local disks usage **User 3** that is managing a VO: • **Role**: Grid • **Goal**: monitoring disk space allocated to the VO & distribution and striping of the VO's files over the Grid.
Preconditions	GridFS is used.
Triggers	User request / command
Basic course of events	**Nominal GUC GFS.7 # 1 :** 1. User command asks FS to get information on its usage of the filesystem. 2. FS acknowledges and returns the requested information or denies request. **Nominal GUC GFS.7 # 2 :** 1. User command asks FS usage of local filesystem 2. FS acknowledges and returns the requested information or denies request. **Nominal GUC GFS.7 # 3 :** 1. User process asks FS about filesystem usage of a specific VO 2. FS acknowledges and returns the requested information or denies request.

Alternative paths	**Alternative GUC GFS.7 # 1 :** user not allowed to request the parameters or parameter values invalid 1. FS informs the user about the error.
Postconditions	
Notes	This use case covers monitoring both the file system and the file system daemons because there use case steps, preconditions etc are the same for both cases.
Author, date	Guillaume ALLEON (EADS), 15.11.2006

GUC GFS.8	Exporting a file or a directory
Summary	Exporting a file or a whole directory recursively to a GridFS volume.
Actors, Roles and Goals	**User**: a global Grid user • **Role**: a global Grid user • **Goal**: to manage his or her GridFS files and directories
Preconditions	• The user must have initiated a Grid session. • The local file or directory that is to be shared exist. • The user must have access to the local directory or file. • The user must have the proper file permissions. • The system has the corresponding GridFS volume mounted on the host.
Triggers	2. User request/command
Basic course of events	**Nominal GUC GFS.8 # 1 :** 1. The user requests exporting a file or a whole directory to a GridFS volume by invoking a specific command and specifying the path of the file or directory and the GridFS volume. 2. The operating system performs all required operations so that the files and directories can be accessed through the GridFS volume. 3. The command returns successfully.
Alternative paths	**Alternative GUC GFS.8 # 1 :** System call failure 2. The operating system fails on a system call (bad permissions, bad path, bad volume, ...) 3. The system call returns with error and sets errno accordingly. 4. The command fails and shows the corresponding error message.
Postconditions	The file or directory and its contents are now accessible through the specified GridFS volume. Depending on the GridFS policies and parameters, the files can later be physically moved, replicated etc.
Notes	This use case is not planned to be supported by the core GridFS services, but can be implemented on top of the latter. It remains to be defined which global Grid users have read/write access to the newly exported file. One proposal is that the exporting user is the only one able to access the file. He must then explicitly allow other global Grid users (or whole VOs) to access it.
Author, date	Josep M. Perez (BSC), November 6th 2006.

GUC GFS.9	Unexporting a file or a directory
Summary	Updating local copy of a GridFS file (or a whole directory recursively) and removing it from the GridFS volume.
Actors, Roles and Goals	**User**: a global Grid user • **Role**: a global Grid user • **Goal**: to manage his or her GridFS files and directories
Preconditions	• The user must have initiated a Grid session. • The file or directory must exist. • The user must have the proper file permissions. • The system has the corresponding GridFS volume mounted on the host.
Triggers	3. User request/command
Basic course of events	**Nominal GUC GFS.9 # 1 :** 1. The user requests unexporting a file or a whole directory from a GridFS volume by invoking a specific command and specifying the local path of the file or directory and the GridFS volume. 2. The operating system performs all required operations so that the files and directories

	are updated locally and they cannot be accessed through the GridFS volume. 3. The command returns successfully.
Alternative paths	**Alternative GUC GFS.9 # 1 :** System call failure 2. The operating system fails on a system call (bad permissions, bad path, bad volume, ...) 3. The system call returns with error and sets errno accordingly. 4. The command fails and shows the corresponding error message.
Postconditions	The local file or directory and its contents have been updated locally and have been removed from the GridFS volume.
Notes	This use case is not planned to be supported by the core GridFS services, but can be implemented on top of the latter.
Author, date	Josep M. Perez (BSC), November 6th 2006.

3.1.5 Grid Monitoring Use Cases

This section deals with 5 general use cases related to monitoring application on XTREEMOS. These are:

- GUC MON.1 : getting (aggregated over all users) information on node loads, amounts of network transfer, number of node failures etc
- GUC MON.2: getting per-resource information used for accounting
- GUC MON.3: getting non-aggregated tracing information.
- GUC MON.4: Dealing with security attacks
- GUC MON.5: monitoring Workflows.

The monitoring system will be perfectly extensible, that is, events and properties to be monitored can be added and removed at any time by the user or other servers. In addition these new events do not need to be pre-established; they can be completely new and not though in advance.

The diagram below shows the relationships between the use cases and roles. Note that those use cases in the diagram that are very similar to some other use case are not mentioned in the above list nor described in detail.

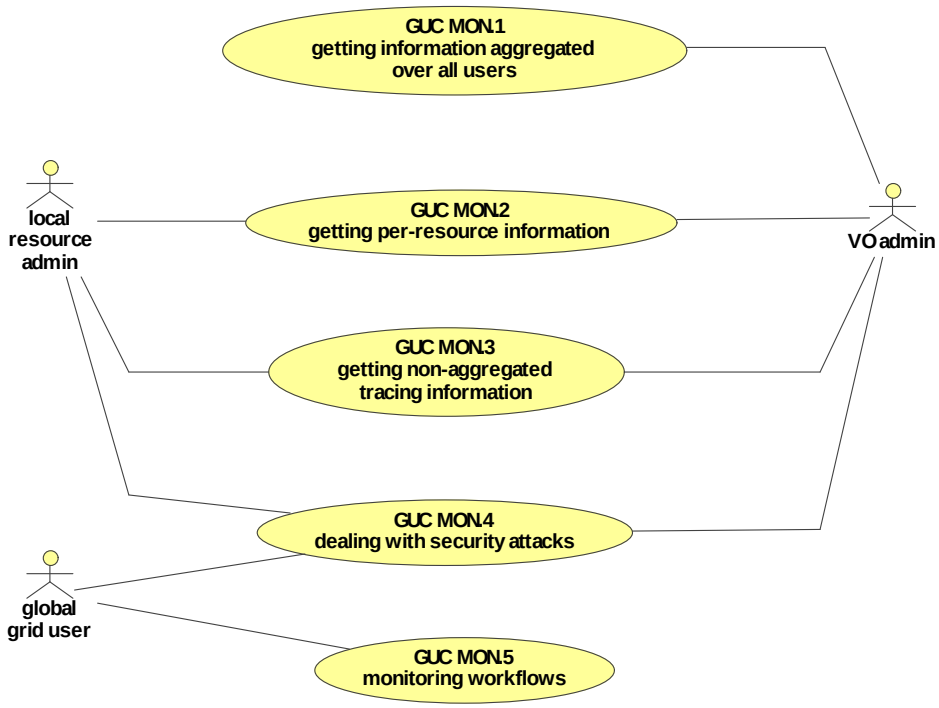


Figure 30: Overview of Grid monitoring use cases

GUC MON.1	Getting information aggregated over all users of a VO
Summary	It must be possible to record the usage of the Grid resources on a per VO basis.
Actors, Roles and Goals	**User**: (human or software) • **Role**: VO Administrator

	• **Goal**: Get accounting information about the usage of the Grid resources by a specific VO
Preconditions	• There is a collection of jobs running on the GRID. o It is possible to get information for each running job ▪ Start time ▪ End time ▪ Memory consumption ▪ CPU consumption ▪ Storage consumption ▪ Owner o The system knows members of a VO • There is an accounting process (or a collection) where jobs will submit their accounting information
Triggers	--
Basic course of events	**Nominal GUC MON.1 # 1 :** • Each job collects its own accounting information if it run by somebody part of the specific VO • Jobs submit information to the accounting process each X seconds • The account process manage data and store it in a proper way to be reviewed later
Alternative paths	**Alternative GUC MON.1 # 1 :** Collecting information (snapshot) • VO admin submit a request in order to get accounting information • The system checks if the user is allowed to get this information (permissions) • The system collects information about all processes running by an specific Grid User or VO **Alternative GUC MON.2 # 2 :** Collecting information (by date) • See Alternative GUC MON.1 # 1 • The system collects stored accounting information for an specific VO • The actual and stored information is aggregated.
Postconditions	• Accounting data is saved in the system • Allowed user and administrators can get information of the actual or/and past resource usage.
Notes	--
Author, date	Guillaume ALLEON (EADS), November 15th

GUC MON.2	Getting per-resource information used for accounting
Summary	It must be possible to record the usage of a resource by a user at any given time. A way to implement / use special cost models must also be provided.
Actors, Roles and Goals	**User**: (human or software) • **Role**: Grid User, VO Administrator, Local administrator • **Goal**: Get accounting information about the usage of the resources by an specific user or VO **Job:** 1. **Role:** Task to be done 2. **Goal:** finish computation **Accounting process** 1. **Role**: Core process 2. **Goal**: Get notification of jobs usage and store data for further analysis
Preconditions	• There is a collection of jobs running on the GRID. o It is possible to get information for each running job ▪ Start time ▪ End time ▪ Memory consumption ▪ CPU consumption ▪ Storage consumption ▪ Owner o The system knows members of a VO

	• There is an accounting process (or a collection) where jobs will submit their accounting information. This accounting process is owned by the system itself, not by the user.
Triggers	--
Basic course of events	**Nominal GUC MON.2 # 1 :** • Each job collects its own accounting information • Jobs submit information to the accounting process each X seconds Default time can be set by the administrator of the system. It can be a fix period of time (ie. slots of 10 seconds) or it can simply reacts when a new events, like increasing amount of memory or storage by 1 Mb, occurs. • The account process manage data and store it in a proper way to be reviewed later.
Alternative paths	**Alternative GUC MON.2 # 1 :** Collecting information (snapshot) • User submit a request in order to get accounting information • The system checks if the user is allowed to get this information (permissions) • The system collects information about all processes running by an specific Grid User or VO **Alternative GUC MON.2 # 2 :** Collecting information (by date) • See Alternative GUC MON.2 # 1 • The system collects stored accounting information for an specific User or VO • The actual and stored information is aggregated.
Postconditions	• Accounting data is saved in the system • Allowed user and administrators can get information of the actual or/and past resource usage.
Notes	--
Author, date	Javier Noguera (T6), November 14th

GUC MON.3	Getting non-aggregated tracing information
Summary	Tracing should 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, so a mechanism should be in place to set these.
Actors, Roles and Goals	**User**: (human or software) 1. **Role**: Grid User 2. **Goal**: Get information about the used resource usage **Application:** 1. **Role:** Task to be done 2. **Goal:** finish computation
Preconditions	• The user wants to know the resource usage • System will send events to the user when: o The application start/stop/pause/resume o The application open/close a file or socket o The application tries to allocate/create new jobs o The application create new threads • New jobs are associated automatically to the main tracing process and can then be traced
Triggers	• A callback mechanism is created between the job and the main user process. • A new process in the user space is created in order to collect all information from the jobs. User can then ask to this process to show tracing information.
Basic course of events	**Nominal GUC MON.3 # 1 :** • User submit an application in a special way to indicate to the system he want tracing information (usually adding tracing configuration) • The application sends information to the user when a traceable event occurs • User can ask for actual resource consumption
Alternative paths	**Alternative GUC MON.3 # 1 :** Runtime tracing • User create a listener able to get tracing events in runtime • User can associate a listener to an specific job an then get real-time information of

	that job resource usage
Postconditions	User can get information about resource usage of his applications.
Notes	--
Author, date	Javier Noguera (T6), November 14th

GUC MON.4	Dealing with security attacks
Summary	Detection, alert distribution, hindering of attack propagation and discovery of attack source. Attackers
Actors, Roles and Goals	**Admin (human/software)**: Site and VO administrators • **Roles**: Security administration roles • **Goal**: Maintaining VOs and XtreemOS resources. Configuration and application of security policies. **User:** VO/local user • Roles: VO member; registered Grid user; Local User • **Goal:** Perform tasks and business processes using VO and XtreemOS resources. **Target:** VO resources • **Roles:** VO resource, local system • **Goal:** to deliver services with a agreed on service quality. **Attacker**: Local user, VO user, external user (Assumption administrators/super-users are trustworthy) • **Roles:** Legitimate User, Illegitimate User • **Goal:** To violate security policies and to derogate service quality attributes.
Preconditions	• Nodes are capable of monitoring resource utilization and requests to resources. • Nodes are capable of forwarding this information to a collector and detector.
Triggers	An attack pattern is detected by at least one node acting as a network monitor in XtreemOS resource or in a VO
Basic course of events	**Nominal GUC MON.4 # 1 :** • Users performing tasks and implementations of processes. • Appearance of requests to service, programming interfaces and resources in untypical quantity, order or with untypical arguments. • Derogated of service quality attributes.
Alternative paths	**Alternative GUC MON.4 #1 :** Unknown Attack Signature This path occurs if an attack signature was unknown and the attack has been successful on one node. The alert should indicate that the node is unavailable until further notice **Alternative AUC MON.4 #2 :** User action required Although it is undesirable to introduce users into the process of dealing with security attacks, it is still necessary at some points to alert them to take appropriate actions. For example, a user may need to update their OS or application in order to alleviate a known vulnerability. **Alternative AUC MON.4 #3 :** Attack-target hosts critical services If the attack-target hosts critical services such as a VO membership database, the removal of this node will disrupt a wide set of VOs and Grid users. It must be possible to migrate at least some users and VOs to an alternative provider of critical services, with minimal downtime. (there may be more alternative paths)
Postconditions	The post conditions are ordered with respect to the most ideal case: • Data about resource utilization, quantity and atypical messages is available as a log. • Distributed logs can be integrated. • The authenticity and integrity of a log is guaranteed. • Authorized principals have access to a log only.
Notes	See http://www.seifried.org/security/index.php/Linux_Security for useful details concerning

	support for attack detection and recovery in Linux. http://osg-docdb.openscienceGrid.org/0000/000019/002/OSG_incident_handling_v1.0.pdf provides a few references and related work on incident handling in the Grid
Author, date	Philip ROBINSON (SAP), November 6th 2006

GUC MON.5	Workflow Monitoring.
Summary	Monitoring the precedence constraints related execution of individual tasks as part of applications on Grid resources. To this end, the application owners as well as the resource providers must be able to monitor the past and current resource allocation. However, some information is only available for a restricted set of individuals.
Actors, Roles and Goals	**Application Owner** • **Role**: Monitors the progress of the overall application; is interested in the resource consumption from the past and at the moment; requires estimates for the application completion time and for the completion time of individual tasks; is interested in the plan resource allocation for the individual tasks; provides a directed acyclic graph to the workflow engine • **Goal**: tries to control the whole workflow with human being implicit knowledge; in cases of hard failures of in the presence of hard deadlines the application owner might modify the resource requirement specifications **Workflow Engine** • **Role**: acts as a user from the system point of view; coordinates the different tasks automatically; uses a directed acyclic graph (DAG) to describe the overall application; reacts automatically on resource failures • **Goal**: tries to start the individual task corresponding to the user objectives (in most cases the minimization of the application completion time can be assumed); must be able to detect failures (resource as well as software failures); reacts on failures by restarting tasks; only hard failures must be resolved by the application owner **Host Monitoring** • **Role**: controls the state of all nodes within the system; is able to provide past and current data about the usage • **Goal**: controls the different nodes in the Grid; in some cases (only if the application owner has specified it) the host monitoring can restart individual jobs in cases of failures **Network Monitoring** • **Role**: monitors the current network traffic; estimates the future network usage • **Goal**: provision of precise network information about the current network and about the estimated network usage in the future **Data Replica Manager** • **Role**: duplicates data if necessary; decides when to delete which data set • **Goal**: tries to provide quick data access for all jobs within the system; might decide to replicate data; provides a versioning of data **Job Manager/Scheduler** • **Role**: controls and monitors the execution of individual tasks; provides estimates about the planned execution of waiting jobs • **Goal**: tries to start jobs as soon as possible regarding their priority and their constraints **Administrator (Site and VO)** • **Role**: provides the necessary resources (CPU, network, storage etc.) • **Goal**: wants to control the resource usage by application, by workflow and by users individually; want to adapt the charges for execution corresponding to the resource usage and the relationship to the application owner
Preconditions	• The application owner is able to describe the workflow with an DAG • The communication / data exchange within the system is secure • The application owner has the permissions to execute the whole workflow • The necessary input data are available. • All managers/monitors are up and running
Triggers	A user or application owner submits a workflow application to the workflow engine that runs on the systems

Basic course of events	**Nominal GUC MON.5 # 1 :** • The user submits a DAG to the workflow engine • The workflow engine analyzes the DAG and derives which subtasks can already be started • The workflow engine requests estimates from the job manager in order to estimate the whole application flow and the execution times/intervals of all sub tasks. This ends in a first coarse schedule of the whole workflow • All jobs without further constraints are sent to the job manager • The job manager incorporates the replica manager, the network monitoring, and the host monitoring tool during the decision process where to start the individual jobs. For all jobs the job manager must provide an estimate of their start and completion times. • The replica manager moves data (in case that is necessary) • The host manager starts the individual jobs on the various nodes • The job manager and the workflow engine get the information about the (planned) job execution • During the execution, the network monitoring, data replica manager, and host monitoring are updated with "fresh" data frequently • After all jobs of the whole workflow have been executed, the whole application finishes, • The output data are copied by the replication manager to the predefined directories.
Alternative paths	**Alternative GUC MON.5 # 1 :** Node failure In this case, a node failure during the execution of an individual job is assumed 1. The host monitoring must recognise the node failure 2. The node failure is reported to the job manager and to the administrator 3. If the job is independent and can be restarted (must be specified in advance) the job manager is able to restart the job on an alternative node (by incorporating the data replica manager) 4. The workflow engine must be notified by the node monitoring system as an individual failure might result in further delays for all other jobs 5. The workflow engine analyzes the dependencies and updates the data of the job manager 6. The job manager updates all job specifics and reports the result to the workflow engine 7. The workflow engine informs the application user (by call-back or by email) about the delay **Alternative GUC MON.5 # 2 :** Permanent failure of individual jobs This failure might result from software errors 4. The host monitoring informs the job manager about the completion of a job 5. The job manager analyzes the state of the job and realizes the abortion 6. If a restart and relocation was not successful, the job manager informs the workflow engine 7. The workflow engine reports the permanent failure to the application owner 8. The application owner must resolve this problem manually by exchanging the sources for the individual job or be re-defining the overall workflow
Postconditions	• The whole workflow was executed • The output data are stored in the predefined directories • All intermediate data are deleted at all nodes • The user is informed about the finished workflow application by call-back or email • No sub-task of the overall workflow is running anymore on any node • Some data replica that are not needed anymore are deleted • The host monitoring, replica monitoring, network monitoring, job monitoring have updated their list of running jobs and are able to detail the resource consumption in the past • The "billing" for the resource usage has been done
Notes	The workflow manager is not a part of XtreemOS, but rather a third-party tool on top of XtreemOS.
Author, date	Carsten Franke (SAP), 7th November

3.2 Application-specific Use Cases

3.2.1 SPECweb2005

AUC SPEC-WEB.1	Setting up the application
Summary	The application needs to be set up manually over a network before it can be run. The use case for the steps to do that are described here.
Actors, Roles and Goals	**User**: Administrator • **Role**: Person (initiator) • **Goal**: Provide the application for use in the system **User**: End-user • **Role**: Person • **Goal**: Make use of the application.
Preconditions	• Privileged access to a network with several systems. • A Java Virtual Machine (1.4.1 or greater recommended) on each system in the network. • PHP (4.3.X or newer) or JSP capabilities required for the system running the Web Server.
Triggers	Manually by administrator wishing to provide use of application to users.
Basic course of events	**Nominal AUC SPECWEB.1 # 1** 1. For each system that is part of the test setup , run the following command: 2. "**java –jar setup.jar**" 3. From the list of options presented, select to install a client
Alternative paths	**Alternative AUC SPECWEB.1 # 2 :** Prime Client setup 1. Select "Prime client" option from the setup app. 2. Configure .config files with respect to the setup **Alternative AUC SPECWEB.1 # 1 :** Web Server setup 1. Select "Web Server" option from the setup app. 2. Depending on the Web Server, copy and setup either php or jsp files to it. 3. Enter wafgen directory and run wafgen according to its README 4. Verify correct installation on Web Server **Alternative AUC SPECWEB.1 # 2 :** BeSim setup 1. Select "BeSim" option from the setup app. 2. Enter the besim directory of the install directory and make the code 3. Install this on a prepared Web Server. 4. Use the verification scripts to check that everything is correct.
Postconditions	There must be at least one BeSim system setup in the network. There must be a Primary Client selected for the network. All clients must be able to contact the SUT and the SUT must be able to access the BeSim machine.
Notes	The machine running the Web Server is known as the SUT (system under test).
Author, date	Kevin Hogan (BSC), November 7th 2006

AUC SPEC-WEB.2	Launching the application
Summary	This describes how the application is launched.
Actors, Roles and Goals	**User**: End-user • **Role**: Person • **Goal**: Test a Web Server
Preconditions	• Network must be set up with a correct and valid installation of the application before it can be run. • The user must have access to the different systems being used on the network.
Triggers	User wishing to test or benchmark a Web Server.

Basic course of events	**Nominal AUC App.1 # 1 :** 1. On each client machine the following command is run: "**java -Xms512m -Xmx512m -jar specwebclient.jar**" 2. Finally to start the test, the primary client needs to be run. This starts the test according to its configuration files and should be invoked using the following command: "**java -Xms512m -Xmx512m -jar specweb.jar**"
Alternative paths	None
Postconditions	Benchmark obtained or test completed.
Notes	None
Author, date	Kevin Hogan (BSC), November 7th 2006

3.2.2 GSfastDNAml (BSC)

AUC GSDNA.1	Deploying the application
Summary	This is an administrator level use case that describes the process that must be performed in order to be able to run the application on the Grid.
Actors, Roles and Goals	**User**: Application end user • **Role**: Person (initiator) • **Goal**: take benefit of being able to run the application **User**: Administrator • **Role**: Person • **Goal**: install the application on the Grid
Preconditions	A pair of executables must exist which correspond to the master and worker of the application.
Triggers	The end user wishes to use the application.
Basic course of events	**Nominal AUC GSDNA.1 # 1 :** 1. The end user asks the administrator to install the application. 2. The administrator exports the executable files to the Grid filesystem. 3. The administrator initiates a session on the Grid. 4. The administrator gives proper access rights to the aforementioned files. 5. The administrator terminates the Grid session. 6. The administrator notifies the application end user.
Alternative paths	None
Postconditions	The application binaries can be accessed form the Grid filesystem and have the proper access permissions for the end user.
Notes	None
Author, date	Josep M. Perez (BSC), November 6th 2006

AUC GSDNA.2	Running the application
Summary	This is a user level use case that describes the process an end user must perform in order to obtain the maximum likelihood phylogenetic tree of a set of taxa.
Actors, Roles and Goals	**User**: Application end user • **Role**: Person (initiator) • **Goal**: obtain the maximum likelihood phylogenetic tree of a set of taxa.
Preconditions	The end user must have read access to a set of files containing taxa written in Phylip format.
Triggers	The end user wishes to obtain the maximum likelihood phylogenetic tree of the aforementioned set of taxa.

Basic course of events	**Nominal AUC GSDNA.2 # 1 :** 1 The end user gathers all the files corresponding to the taxa and creates a file containing the taxa that will be evaluated. 2 The end user exports the taxa file to the Grid file system. 3 The end user initiates a Grid session. 4 The end user starts the GSfastDNAml application from the command line and passes as an input file the one generated in the previous step. 5 The end user waits for the application to finish. 6 The end user terminates the Grid session. 7 The end user removes the input file from the Grid file system. 8 The end user imports the output files.
Alternative paths	**Alternative AUC GSDNA.2 # 1 :** Invalid input file 5. If the end user generated an invalid input file, then the application will finish with an error. **Alternative AUC GSDNA.2 # 2 :** Execution with monitoring 5. The end user wishes to monitor the application execution. 6. The end user starts the GRID superscalar monitor. 7. The end user surveys the application execution while it is running.
Postconditions	The user obtains two files containing the outputs of the execution.
Notes	None
Author, date	Josep M. Perez (BSC), November 3rd 2006

AUC GSDNA.3	Starting a GRID superscalar task
Summary	This is a low level use case that shows the interaction between the GRID superscalar run time and XtreemOS services when starting an application task (a phylogenetic tree evaluation).
Actors, Roles and Goals	**User**: GRID superscalar runtime (initiator) • **Role**: Application library • **Goal**: to manage the execution of the tasks that compose the application **User**: XtreemOS filesystem service • **Role**: OS service • **Goal**: to allow jobs to access files transparently across the Grid **User**: XtreemOS resource discovery service • **Role**: OS service • **Goal**: to allow applications to query the Grid nodes that are available **User**: XtreemOS execution service • **Role**: OS service • **Goal**: to start jobs on Grid nodes
Preconditions	The application is running under the proper credentials.
Triggers	The GRID superscalar runtime has decided that one GRID superscalar task is ready for execution.
Basic course of events	**Nominal AUC GSDNA.3 # 1 :** 1. The runtime requests the discovery service for a node where it can run a GRID superscalar task. 2. The discovery service returns a node reference to the runtime. 3. The runtime generates the task's input files on the Grid. 4. The runtime requests the execution service for the execution of a job, specifying that it • has some input files (the input files in the previous step) • has as an executable, the worker executable • has a series of parameters for the executable • has a series of files that must be placed in the Grid filesystem • must be notified of the job's finalisation by means of a callback 5. The execution service replies that the request was successful.

Alternative paths	**Alternative AUC GSDNA.3 # 1 :** No resource available 2. The discovery service returns an exception indicating that there are no resources available. 3. The use case finishes with a "no free resources" exception. **Alternative AUC GSDNA.3 # 2 :** Job error 5. The execution service returns an exception indicating that the job could not be submitted for execution. 6. The use case finishes with a fatal exception.
Postconditions	The Grid filesystem contains the files generated in step 3 and the job from step 4 is running or waiting to be executed on the Grid.
Notes	None
Author, date	Josep M. Perez (BSC), November 3rd 2006

AUC GSDNA.4	End of job notification
Summary	This is a low level use case that shows the interaction between the GRID superscalar run time and XtreemOS services when a GRID superscalar task finishes.
Actors, Roles and Goals	**User**: GRID superscalar runtime • **Role**: Application library • **Goal**: to manage the execution of the tasks that compose the application **User**: XtreemOS execution service (initiator) • **Role**: OS service • **Goal**: to start jobs on Grid nodes
Preconditions	A GRID superscalar task has been sent for execution by following the "Starting a GRID superscalar task" use case and it has just finished.
Triggers	The GRID superscalar task was running and has just finished.
Basic course of events	**Nominal AUC GSDNA.4 # 1 :** 1. The execution service notifies the runtime that a job has finished by performing a callback. 2. The runtime parses the outputs file if necessary. 3. The runtime updates its internal data structures.
Alternative paths	**Alternative AUC GSDNA.4 # 1 :** The job has failed 1. The execution service notifies the runtime that a job has failed by performing a callback. 2. The runtime updates its internal data structures.
Postconditions	The runtime data structures have been updated so that the corresponding task is marked as finished.
Notes	None
Author, date	Josep M. Perez (BSC), November 3rd 2006

AUC GSDNA.5	Application finalisation
Summary	This is a low level use case that shows the interaction between the GRID superscalar run time and XtreemOS services when the application finishes.
Actors, Roles and Goals	**User**: GRID superscalar runtime (initiator) • **Role**: Application library • **Goal**: to manage the execution of the tasks that compose the application **User**: XtreemOS filesystem service • **Role**: OS service • **Goal**: to allow jobs to access files transparently across the Grid **User**: XtreemOS resource discovery service • **Role**: OS service • **Goal**: to allow applications to query the Grid nodes that are available **User**: XtreemOS execution service • **Role**: OS service • **Goal**: to start jobs on Grid nodes
Preconditions	All the application's GRID superscalar tasks have been finished.
Triggers	The "End of job notification" use case has been executed and the runtime data structures indicates that all the application's tasks have already finished.

Basic course of events	**Nominal AUC GSDNA.5 # 1 :** 1. The runtime gathers the list of temporary files. 2. The runtime requests the filesystem service to remove the temporary files from the Grid filesystem.
Alternative paths	None
Postconditions	The temporary files no longer exist on any filesystem.
Notes	None
Author, date	Josep M. Perez (BSC), November 3rd 2006

3.2.3 Elfipole (EADS)

AUC ELF.1	Integration of Elfipole
Summary	Before a workflow based on Elfipole components can be run, it has to be designed
Actors, Roles and Goals	**User**: Integrator • **Role**: Person • **Goal**: set up the basic Grid services for enabling elfipole components.
Preconditions	• User has its applications (anadel, elfipole, …) ready to be run (i.e. compiled) on XtreemOS • The submission parameters intervals are clearly identified : o Number of nodes required to run the simulation : ideal, maximum and minimum o Parameter passed as argument to the executable o input and expected output files
Triggers	The process is triggered manually
Basic course of events	**Nominal AUC ELF.1 # 1 :** 1. The User is authentificated by the system and has the credentials to create a workflow 2. Appropriate services (job services) are created and configured for each of the Elfipole components 3. Appropriate services (data services) are created and configured for each data transfer between job services 4. The workflow is saved and published with appropriate access rights in a directory
Alternative paths	**Alternative AUC ELF.1 # 2 :** The design is experiencing failure 1. All job services have to be created prior to any data service 2. The repository is not available
Postconditions	The different Grid services are configured so that an Elfipole workflow can be executed
Notes	
Author, date	Guillaume Alleon, 13/11/2006

AUC ELF.2	Submission, Run and Monitoring of Elfipole
Summary	The workflow has already been designed, and it can be run on the XtreemOS platform
Actors, Roles and Goals	**User**: Application End User • **Role**: Person • **Goal**: runs its workflow of applications on XtreemOS using all the Resources available.
Preconditions	• User has its applications (anadel, elfipole, …) ready to be run (i.e. compiled) on XtreemOS • The workflow has been designed for XtreemOS Grid services
Triggers	The workflow is manually started or through a job

Basic course of events	1. The User is authentified by the system. 2. He submits his workflow to XtreemOS. 3. Asked Resources are allocated on XtreemOS. 4. Input Files are transmitted as well as the executable of the application. 5. The application starts taking as an argument the parameters given by the End User during the job submission 6. Processes are dynamically deployed on the Grid. Some of them collect or distribute work to the other and need to be run on virtual nodes in order to prevent the application from crashing if one of the nodes fails. 7. The application runs and ends. Eventually a final collection of data is made by the collector nodes which produce result output files. These are moved back to the End User. 8. The End user is notified of the end of his job 9. All Resources are freed, and any file or executable from the run of the application on XtreemOS is deleted 10. After having logged out from XtreemOS, the End User can locally check the output results produced by the run
Alternative paths	**Alternative AUC ELF.2 # 1 :** Connection with the End User is made by the application Same scenario as in the nominal case but the application also needs to connect to the user to ask him for interactive inputs (anadel is a typical example) **Alternative AUC ELF.2 # 2 :** Application is experiencing Resource failure Same scenario as in the nominal case but a resource failure occurs while the application is running: 1. While job is running, a node fails or a file system collapses o If asked, in the job card submitted, MPI critical applications will only be run on virtual nodes and will not notice any change in the Resource thanks to redundancy of nodes or file systems…. 2. These Events are traced by XtreemOS and gathered in a report file which is sent to the End user at the end of the run.
Postconditions	The application is not using resources of XtreemOS anymore
Notes	
Author, date	Guillaume Alleon, 13/11/2006

3.2.4 HLA (EADS)

AUC HLA.1	Integration of simulation components
Summary	Prior to their utilization the different components need to be published
Actors, Roles and Goals	**User**: Integrator • **Role**: Person • **Goal**: set up the basic Grid services for enabling simulations to be run.
Preconditions	• User has its applications components ready to be run on XtreemOS. • The submission parameters intervals are clearly identified : o Number of nodes required to run the simulation; ideal, maximum and minimum o Parameter passed as argument to the executable o Input and expected output files
Triggers	The process is triggered manually
Basic course of events	**Nominal AUC HLA.1 # 1 :** 1. The User is authentificated by the system and has the credentials to create a workflow. 2. Appropriate services (job services) are created and configured for each of the HLA com-

	ponents 3. Appropriate executions rights are set according to VO's for each components by their owners
Alternative paths	
Postconditions	The different Grid services are configured so that basic components can be executed
Notes	
Author, date	Guillaume Alleon, 13/11/2006

AUC HLA.2	Submission, Running and Monitoring of HLA simulation
Summary	The HLA simulation can be designed based on each component and then run
Actors, Roles and Goals	**User**: Application End User • **Role**: Person • **Goal**: runs its simulation on available XtreemOS resources
Preconditions	• User has all the expected components ready to be run (i.e. deployed) on XtreemOS • The simulation has been designed (configured)
Triggers	The simulation is manually started or started through a job
Basic course of events	1. The User is authentified by the system 2. He submits his simulations to XtreemOS. 3. Asked Resources are allocated on XtreemOS according to the components requirements and rights are given by the integrator 4. The application runs and ends. Data is exchanged between the components through HLA. Eventually a final collection of data is made by the collector nodes, which produce result output files. These are moved back to the End User. 5. The End user is notified of the end of his simulation 6. All Resources are freed 7. After having logged out from XtreemOS, the End user can locally check the output results produced by the run
Alternative paths	
Postconditions	The application is not using resources of XtreemOS anymore
Notes	
Author, date	Guillaume Alleon, 13/11/2006

3.2.5 Simulations (EDF)

AUC SIMU.1	MODERATO and SIMEON
Summary	Submission, Run and Monitoring of two Gridified EDF applications : MODERATO and SIMEON.
Actors, Roles and Goals	**User**: Application End User • **Role**: Person • **Goal**: run its Griddified application on XtreemOS using all the resources available.
Preconditions	User has its application ready to be run on XtreemOS The application is packaged for an XtreemOS submission with required Resources clearly identified : Number of nodes required to run the simulation : ideal, maximum and minimum Parameter passed as argument to the executable input and expected output files
Triggers	A job can be submitted either by an end user, or by another application.

Basic course of events	**Nominal AUC SIMU.1 # 1 :** The End User is authenticated by the system He submits his jobs to XtreemOS. The required Resources are granted on XtreemOS. Input Files are transmitted as well as the executable of the application. The application starts by taking as an argument the parameters given by the End User in the job submission Processes are dynamically deployed on the Grid. Some of them collect or distribute work to the other and need to be run on virtual nodes in order to prevent the application from crashing if one of the nodes fails. The application runs and ends. Eventually a final collection of data is made by the collector nodes which produce result output files. These are moved back to the End User. The End user is notified of the end of his job All Resources are freed, and any file or executable from the run of the application on XtreemOS is deleted After having logged out from XtreemOS, the End user can locally check the output results produced by the run
Alternative paths	**Alternative AUC SIMU.1 # 1 :** Number of nodes allocated for the job changes Same scenario as in the nominal case with varying amount of nodes made available to run the application during the job life: while job is running, new nodes can be used by the application Either, the application dynamically submits work to the existing available resources. This list of resources is regularly updated by XtreemOS. That way, use of additional resource is transparent Or, the application has registered to a resource announcing service and receives a signal when a new resource is available. This new distribution is only processed if the signal is issued. These Events are traced by XtreemOS and gathered in a report file sent to the End user at the end of the run. **Alternative AUC SIMU.1 # 2 :** Application is experiencing Resource failure Same scenario as in the nominal case with a resource failure occurring while the application is running: while job is running, a node fails or a filesystem collapses If it is a collector or a distributor node hosted by an XtreemOS virtual node, a redundant node replaces the missing Resource in a transparent way for the application and for the user. If it is a regular node, the application – targeted for Grid – copes with the deletion of this process and respawns the lost work on another node These Events are traced by XtreemOS and gathered in a report file sent to the End user at the end of the run.
Postconditions	The application is not using resources in XtreemOS anymore
Notes	None
Author, date	Samuel KORTAS (EDF), November 5th 2006

AUC SIMU.2	ZEPHYR
Summary	Submission, Run and quasi-real time Monitoring of an EDF Application that can also be run in MPI environment : ZEPHYR
Actors, Roles and Goals	**User**: Application End User • **Role**: Person • **Goal**: runs ZEPHYR on XtreemOS; eventually on several processors in an MPI environment. He also wants to inspect the results produced on the fly.

Preconditions	• User has its application ready to be run on XtreemOS • The application is packaged for an XtreemOS submission with required Resources clearly identified : o Number of nodes required to run the simulation : ideal, maximum and minimum o Parameter passed as argument to the executable o input and expected output files
Triggers	A job can be submitted either by an end user, or by another application.
Basic course of events	**Nominal AUC SIMU.2 # 1 :** 1. The End User is authenticated by the system 2. He submits his jobs to XtreemOS. 3. The required Resources are granted by XtreemOS. 4. Input Files are transmitted as well as the executable of the application. 5. The application starts by taking as an argument the parameters given by the End User in the job submission 6. Processes are dynamically deployed on the Grid. Their number remains constant during the run. The application does not know how to handle a node failure, cope with a resource shortage or take benefit of additional resources. 7. Periodically, the End User needs to connect to the running job and check, in quasi-real time, if the data produced is that which he expected. The application and XtreemOS have to make this connection possible. 8. The application runs and ends.The output files are moved back to the End User. 9. The End user is notified of the end of his job 10. All Resources are freed, and any files or executable used during the run of the application on XtreemOS is deleted 11. After having logged out from XtreemOS, the End user can locally check the output results produced by the run
Alternative paths	**Alternative AUC SIMU.2 # 1 :** Connexion with the End User is made at application initiative. 1. Same scenario as in the nominal case but application also needs to be connected to the user to ask him for further directions: 2. Application and XtreemOS have to make this connection possible, either by a call back or a notification through the system.. **Alternative AUC SIMU.2 # 2 :** Application is experiencing Resource failure 1. Same scenario as in the nominal case with a resource failure occurring while the application is running: 2. while job is running, a node fails or a file system collapses 3. If asked in the job card submitted, the MPI critical application will only be run on a virtual node and will not notice any change in the Resource thanks to redundancy of nodes or filesystem…. 4. These Events are traced by XtreemOS and gathered in a report file sent to the End user at the end of the run.
Postconditions	The application is not using resources in XtreemOS anymore
Notes	None
Author, date	Samuel KORTAS (EDF), November 5th 2006

3.2.6 Secure Remote Computing (EDF)

AUC SRC.1	Secure Remote Computing
Summary	Application run on XtreemOS in a Secure Environment. Submission, Running, Cancellation and monitoring of the application by an End User.
Actors, Roles and Goals	**User**: Application End User • **Role**: Person • **Goal**: run their application on XtreemOS in a secure environment which is as transparent as possible
Preconditions	• User has its application ready to be run on XtreemOS • The application is packaged for an XtreemOS submission : o resources needed are clearly identified, o input and expected output files are mentioned, o a secure running environment is required
Triggers	A job can be submitted either by an end user, or by another application.
Basic course of events	**Nominal AUC SRC.1 # 1 :** 1. The End User is authenticated by the system 2. He submits his job to XtreemOS. 3. The required Resources are granted on XtreemOS. The connection and access to these resources are secure 4. Input Files are transmitted, as well as the executable of the application. 5. The application starts. *Access to the application, to any of the files it uses or produces is only granted to the End user.* 6. The application runs and ends. Output files are moved back to the End User in a secure way 7. The End user is notified of the end of his job 8. All Resources are freed, and any file or executable from the run of the application on XtreemOS is deleted 9. After having logged out from XtreemOS, the End user can locally check the output results produced by the run

Alternative paths	**Alternative AUC SRC.1 # 1 :** Remote access while Application runs
	
	Same scenario as in the nominal case with the difference that :
	• while job is running, the End User sometimes interacts with the running application :
	  • He may download some output files, or dynamically change some input
	  • He may visualize results remotely
	  • He may cancel the application
	• While job is running, the application may request additional features:
	  • A call back to another application may be issued
	  • Some additional Resources may be needed
	• All these actions are performed in a *Secured environment.*
	
	Alternative AUC SRC.1 # 2 : No secure environment is available when submitting job
	
	Same scenario as in the nominal case with the difference that no secure access to resource can be granted when submitting the job, or when a remote access is requested, while the application is running.
	• The request is kept in the submitting queue in case a secure environment may be scheduled afterwards.
	• In the opposite case, if no secured environment is planed to be available, the job is rejected by XtreemOS and an explicit error message is sent to the End User.
	
	Alternative AUC SRC.1 # 2 : No secure environment is available when the running application requires access
	
	Same scenario as in AUC SRC.1#1 with the difference that no Secure access to resource can be granted when a remote access is requested while the application is running.
	• XtreemOS rejects the demand and an explicit error message is sent to the End User or the application.
	• The application either stops in error, or handles the event by postponing its request
	
	Alternative AUC SRC.1 # 3 : A security attack is detected
	
	Any of the previous scenarios is being run and a security attack is detected
	• XtreemOS stops in priority of the execution of the job that required a secure environment
	• This can happen automatically upon a security alert, or at the explicit demand of an XtreemOS administrator or a Resource Manager, whose resources provided to XtreemOS have been compromised.

Postconditions	The application is not using resources of XtreemOS anymore
Notes	None
Author, date	Samuel KORTAS (EDF), November 5th 2006

3.2.7 WebAS (SAP)

AUC SAP.1	Sales and Distribution (SD): The Sales and Distribution scenario covers important actions of the application and is also used for benchmarking purposes
Summary	This use case covers the steps of an order process and consists of several transactions.
Actors, Roles and Goals	The use case requires a valid application user to perform the transactions. **User**: role of actor 1 • **Role**: The user acts in a role which permits them to perform the transactions • **Goal**: The goal of the user is to input the order in the system and pass the necessary transactions successfully
Preconditions	Each benchmark user has his or her own master data, such as material, vendor, or customer master data to avoid data-locking situations. The user must be: 1. Logged on 2. Have the MAIN SCREEN dialog menu opened 3. The system must be able to accept user input

Triggers	The process is triggered manually or by a batch job
Basic course of events	The following steps are necessary to perform the sales and distribution use case: **Nominal AUC SAP.1 # 1 :** 1. Create an order with five line items. 2. Create a delivery for this order. 3. Display the customer order. 4. Change the delivery and post goods issue. 5. List 40 orders for one sold-to party. 6. Create an invoice. User interaction steps 2 - 16 are repeated n times (15 user interaction steps --> min. 150 sec. duration). One run (user interaction steps 2 - 16) corresponds to the selling of five items.
Alternative paths	In order to penetrate the application and measure its performance the scenario is repeated multiple times.
Postconditions	None
Notes	None
Author, date	Adolf HOHL (SAP), November 7th 2006

AUC SAP.2	ESS Portal Benchmark (EP): The EP portal benchmark covers important actions of the application and is also used for benchmarking purposes
Summary	This use case covers the steps of the ESS Portal Benchmark.
Actors, Roles and Goals	The use case requires a valid application user to perform the transactions. **User**: role of actor 1 • **Role**: The user acts in a role which permits them to perform the transactions • **Goal**: The goal of the user is to input the order in the system and pass the necessary transactions successfully
Preconditions	The user must be logged on
Triggers	The process is triggered manually or by a batch job
Basic course of events	The following steps are necessary to perform EP benchmark: **Nominal AUC SAP.2 # 1 :** User Interaction Steps of the EP-ESS Benchmark 1. Log on to the portal - Welcome Page (includes 3 iViews) 2. Choose Working Time -> Record Working Time 3. Choose Leave Request 4. Choose Leave Request Overview 5. Choose Personal Informaion -> Personal Data t 6. Choose Address 7. Choose Bank Information 8. Choose Beneits and Payment > Paycheck Inquiry 9. Log off from portal Seven different business transactions are launched per loop (step 2-8) . Every loop is repeated n times. The user think time is 10 seconds; thus one loop takes at least 70 seconds.
Alternative paths	In order to penetrate the application and measure its performance, the scenario is repeated multiple times.
Postconditions	None
Notes	None
Author, date	Adolf HOHL (SAP), November 7th 2006

AUC SAP.3	Business Warehaus Benchmark (BW): The BW scenario covers important actions of the application and is also used for benchmarking purposes
Summary	This use case covers the steps of the BW Benchmark.
Actors, Roles and Goals	The use case requires the *simuser* application user to perform the transactions. **User**: role of actor 1 • **Role**: The user acts in a role which permits them to perform the transactions • **Goal**: The goal of the user is to run the penetration test suite.
Preconditions	The user must be logged on
Triggers	The process is triggered manually or by a batch job
Basic course of events	The following steps are necessary to perform EP benchmark: **Nominal AUC SAP.3 # 1 :** 1. Log on as <simuser> to your benchmark driver. 2. Go to <SIMDIR> and create a directory following the specifications in the user environment (parameter MESS. The default is: MESS=<SIMDIR>/mess.<SAPSYSTEMNAME>_bench.) 3. Go to <SIMDIR>/mess.<SAPSYSTEMNAME>_bench and create a directory depending on the benchmark scenario. 4. Go to <SIMDIR>/mess.<SAPSYSTEMNAME>_bench/<benchmark> and create a simulation directory sim<usernumber> . This directory will be reusable, so you don't have to create a simulation directory for every run, but for every number of users you are going to run. 5. Go to <SIMDIR>/mess.<SAPSYSTEMNAME>_bench/<benchmark>/sim<usernumber> and run: mkapl.pl <scenario> <host> <DBhost> <usernumber> <loopnumber> <extended evaluation> > apl (to create the parameter file) Users are started with a delay of 1 second to avoid the situation where all users are running the same transaction at the same time. You can overwrite this default value by setting "userdelay n" in the apl file. This will set the delay to n seconds between the start of two users.
Alternative paths	In order to penetrate the application and measure its performance the scenario can be repeated multiple times.
Postconditions	None
Notes	None
Author, date	Adolf HOHL (SAP), November 7th 2006

3.2.8 DBE (T6)

AUC DBE.1	Deploy a Service
Summary	A service is deployed in the Execution Environment (ExE)
Actors, Roles and Goals	**User:** SME • **Role:** owner • **Goal:** register the service and make it visible to others
Preconditions	ExE is running
Triggers	None
Basic course of events	**Nominal AUC DBE.1 #1 :** 1. Zipped service is sent using HTTP protocol 2. Service is unzipped 3. Classes are loaded and service is started 4. A new job is created and the Job is submitted to a free node 5. Service information is uploaded to the P2P Network 6. A copy of the Zipped service is stored for future deployments
Alternative paths	**Alternative AUC DBE.1 #1 :** If there are no free nodes 1. Keep it running in the master ExE node
Postconditions	Service is registered in the P2P network and can be used.
Notes	None
Author, date	Javier Noguera, November 8th 2006

AUC DBE.2	A service is down
Summary	The ExE detects one of the deployed services is down, it has to be re-deployed.
Actors, Roles and Goals	User: ExE • Role: owner • Goal: register the service and make it visible to others
Preconditions	A service was deployed and now is down. A signal has been received from XtreemOS platform
Triggers	Email or signal.
Basic course of events	**Nominal AUC DBE.2 #1 :** 1. Recover Zipped service, stored when the user deployed it 2. Redo AUC DBE.1
Alternative paths	None
Postconditions	Service is registered in the P2P network and can be used
Notes	None
Author, date	Javier Noguera, November 8th 2006

AUC DBE.3	Duplicate a Service
Summary	The ExE detects one job is consuming CPU and/or Resources and needs to replicate in order to provide High Availability
Actors, Roles and Goals	User: ExE • Role: owner • Goal: duplicate an already running service to allow HA
Preconditions	A service is running out of resources. A signal has been received from XtreemOS platform.
Triggers	Email or signal.
Basic course of events	**Nominal AUC DBE.4 #1 :** 1. Recover Zipped service, stored when the user deployed it 2. A new job is created and the Job is submitted to a free node 3. New request will be balanced between similar services
Alternative paths	None
Postconditions	A new Service is registered in the P2P network and can be used
Notes	None
Author, date	Javier Noguera, November 8th 2006

AUC DBE.4	Undeploy a Service
Summary	An already running service is undeployed. It can be done by User or if it was replicated (AUC DBE.3)
Actors, Roles and Goals	User: SME, ExE • Role: owner • Goal: unregister the service and make it unavailable
Preconditions	ExE is running and service is resisted
Triggers	None
Basic course of events	**Nominal AUC DBE.4 #1 :** 1. Stop service 2. Stop job 3. Unregister from P2P 4. delete the copy of the Zipped file
Alternative paths	None
Postconditions	Service is not available
Notes	None
Author, date	Javier Noguera, November 8th 2006

AUC DBE.5	Process a request
Summary	A service gets a request. It has to process it
Actors, Roles and Goals	User: SME/ExE • Role: customer • Goal: return a result
Preconditions	The service needs to know its current IP (depending on the machine he is running).
Triggers	The ExE will triggers other services in order to store information.
Basic course of events	**Nominal AUC DBE.5 #1 :** 1. The HTTP request is transformed into an invocation (method + parameters) 2. Use service interface and XtreemOS API to perform the invocation
Alternative paths	**Alternative AUC DBE.5 #1 :** It is not possibility to access to the job location 1. The HTTP request is redirected to the computer where the service is running (the destination IP must be known by the ExE when the service is deployed). 2. The service tranforms the request into an invocation (method + parameters) and executes it. **Alternative AUC DBE.5 #2 :** XtreemOS API is not enough location 1. Use rendezvous methods
Postconditions	The result is returned to the service
Notes	None
Author, date	Javier Noguera, November 8th 2006

3.2.9 Tifon (TID)

This section describes the use cases proposed for Tifon running in a mobile device with XtreemOS installed. All the use cases assume that Tifon has been successfully installed and started.

AUC TIF.1	Connection
Summary	This use case describes how the user logs in to the Tifon application in order to communicate with other users. XtreemOS authentication methods will be used.
Actors, Roles and Goals	**User**: an end user who wants to start Tifon. • **Role**: the end user of Tifon. • **Goal**: login to Tifon in order to communicate with other XtreemOS users.
Preconditions	Tifon started and user still not logged in. User can see the connection window.
Triggers	None
Basic course of events	**Nominal AUC TIF.1 # 1:** 1. User fills in the User ID used for unique identification in XtreemOS. 2. User fills in the password used by XtreemOS to authenticate him. Other authentication methods supported by XtreemOS, apart from password, could be used. 3. User fills out other optional fields (for example, a nickname). 4. User clicks on "Connect" to authenticate user and login to Tifon.
Alternative paths	None
Postconditions	If the user logged in correctly, he will be able to communicate with other users connected in the same way, and see their state if presence information is available.
Notes	None
Author, date	TID team, October 31st 2006

AUC TIF.2	Add contact
Summary	This use case describes how the users can add contacts to their local addressbook, in order to communicate with them later. This will also enable them to see the presence status of those contacts, if that information is available.
Actors, Roles and Goals	**User**: an end user who has started Tifon. • **Role**: the end user of Tifon. • **Goal**: adding another user to his addressbook, in order to communicate with him later on, and possibly to see if that user is present or not.
Preconditions	Tifon started and connected. User knows the other user's data.
Triggers	The user right clicks on 'Contacts' → 'Add contact'

Basic course of events	**Nominal AUC TIF.2 # 1:** 1. A dialog window rises. 2. User fills out the data about the user. 3. User clicks on 'OK'
Alternative paths	None
Postconditions	If the user entered the correct information, now he will be able to communicate with the user he just added, provided that he is also running another instance of Tifon.
Notes	The data needed by the addressbook will at least be the other user's global ID, or any other unique ID that Tifon finds useful.
Author, date	TID team, October 31st 2006

AUC TIF.3	Message exchange
Summary	This use case describes how to send an instant message to another user.
Actors, Roles and Goals	**Sender**: an end user who has started Tifon, has logged in, and wants to send a message to another user. • **Role**: the end user of Tifon. • **Goal**: send an instant message to another user. **Receiver**: an end user who has started Tifon and logged in. • **Role**: the end user of Tifon. • **Goal**: since he has started Tifon, his goal is communicating with somebody.
Preconditions	Tifon started and connected. Sender must know the receiver's unique ID, probably already added as a contact in AUC TIF.2
Triggers	The sender double-clicks on a user from his addressbook
Basic course of events	**Nominal AUC TIF.3 # 1:** 1. A conversation window rises. 2. Sender types the text message and clicks 'Send' button to send the message. 3. Receiver receives the message.
Alternative paths	None
Postconditions	None
Notes	None
Author, date	TID team, October 31st 2006

AUC TIF.4	Notice other users' presence state
Summary	This use case describes how we can see other users' state in Tifon in order to see if they are online, away, busy or offline. Basically, depending on this state, we will be able to send them messages or not.
Actors, Roles and Goals	**Active user**: an end user who has started Tifon and logged in. • **Role**: the end user of Tifon. • **Goal**: see the state of other users. **Passive users**: end users who have or have not started Tifon (presence information will differ). • **Role**: the end user of Tifon. • **Goal**: share their presence state
Preconditions	Active user logged in. One or more users added to the addressbook.
Triggers	The active user simply looks at Tifon's graphic interface
Basic course of events	**Nominal AUC TIF.4 # 1:** 1. Tifon gets destination user's presence information from XtreemOS if that information is available. 2. The graphic interface shows the passive users' state.
Alternative paths	**Alternative AUC TIF.4 # 1:** XtreemOS does not provides user's presence information. • Tifon gets destination user's presence information from a presence server or a Peer to Peer presence network.
Postconditions	None
Notes	None
Author, date	TID team, October 31st 2006

AUC TIF.5	Change own presence state
Summary	This use case describes how we can change our state in Tifon in order to appear in other users' application as online, away, busy or offline. Basically, depending on this state, other users will be able to communicate with us or not.
Actors, Roles and Goals	**Active user**: an end user who has started Tifon and is connected to a presence server (or equivalent). • **Role**: the end user of Tifon. • **Goal**: to change his current presence. **Passive user**: end users who have started Tifon and are subscribed to Active user's presence. • **Role**: the end user of Tifon. • **Goal**: since he has started Tifon, his goal is to communicate with somebody, or at least to know somebody's presence state
Preconditions	Tifon started and connected. Active user added to passive user's addressbook.
Triggers	The active user selects a new state from the state selection (or presence control) menu.
Basic course of events	**Nominal AUC TIF.5 # 1:** 1. Change in presence state is provided to proper XtreemOS service in order to be published to users subscribed to that presence information. 2. The other users' Tifon interface show active user in his new state.
Alternative paths	**Alternative AUC TIF.5 # 1:** XtreemOS does not provide/store user presence information. • Change in presence state is sent to a presence server or distributed through a Peer to Peer presence network.
Postconditions	None
Notes	None
Author, date	TID team, October 31st 2006

3.2.10 JobMA (TID)

AUC JMA.1	Application start
Summary	This use case describes how the user can start JobMA.
Actors, Roles and Goals	**User**: an end user who has started the application. • **Role**: the end user of JobMA. • **Goal**: managing and monitoring XtreemOS jobs.
Preconditions	None.
Triggers	The user types the command to start JobMA from the command line.
Basic course of events	**Nominal AUC JMA.1 # 1:** 1. Authentication window rises. 1. The user enters his credentials (User ID/password, Certificate Path...) to be authenticated by XtreemOS mechanisms. This will act as an XtreemOS "login" to the VO. 2. If these pieces of data are correct, JobMA starts and the main panel is shown.
Alternative paths	None.
Postconditions	The user is immediately able to manage jobs under VO policies.
Notes	None.
Author, date	TID team, November 6th 2006

AUC JMA.2	Job launch
Summary	This use case describes how the user can launch a generic job within XtreemOS using JobMA.
Actors, Roles and Goals	**User**: an end user who has started the application. • **Role**: the end user of JobMA. • **Goal**: launching a job to be executed on XtreemOS.
Preconditions	User has adequate data to start the job (input files, job workflow definition, etc.). User has the right permissions in the VO to launch the job.
Triggers	Select job launch from menu.
Basic course of events	**Nominal AUC JMA.2 # 1:** 1. Raise job launch window from menu.

	2. Fill in the parameters to define the job. 3. Launch the job (call to XtreemOS job launch functionality).
Alternative paths	If XtreemOS allows defining jobs from XML files, the following could be an alternative path. **Alternative AUC JMA.2 # 1:** 1. Build a job definition file (i.e. in XML format). 2. Load that file into JobMA. 3. Launch the job (call to XtreemOS job launch functionality).
Postconditions	Once the job is submitted, XtreemOS will enqueue it until required resources are available.
Notes	None.
Author, date	TID team, November 2nd 2006

Before describing this use case, it is necessary to point out that an active job is any job that has been launched for execution on XtreemOS. Thus, its properties can be accessed through XtreemOS functionalities.

AUC JMA.3	Job monitoring
Summary	This use case describes how the user can monitor the status of a job running on XtreemOS using JobMA.
Actors, Roles and Goals	**User**: an end user who has started the application. • **Role**: the end user of JobMA. • **Goal**: monitoring a job that is being executed on XtreemOS.
Preconditions	User has the right permissions to monitor the job. User is interested in a job and wants to know some of its characteristics.
Triggers	Active jobs are shown on the main panel.
Basic course of events	Once the user can see active jobs in the main panel, he may just fulfil a visual tracking of the job or ask for a more complete set of properties about the selected job. This second option is described below. **Nominal AUC JMA.3 # 1:** 1. Select the job to monitor from the main panel. 2. Raise properties window to view as many properties as XtreemOS functionalities can provide.
Alternative paths	None.
Postconditions	Use can close the properties window or update it to track job execution.
Notes	None.
Author, date	TID team, November 2nd 2006

AUC JMA.4	Job stop (pause)
Summary	This use case describes how the user can stop a job that is running on XtreemOS. Note that stop doesn't mean cancel, since a stopped job's execution can be resumed later.
Actors, Roles and Goals	**User**: an end user who has started the application. • **Role**: the end user of JobMA. • **Goal**: stopping (pausing) a job that is running on XtreemOS.
Preconditions	The user has the right permissions to stop the job.
Triggers	Running jobs are shown in the main panel.
Basic course of events	**Nominal AUC JMA.4 # 1:** 1. Select the running job to stop from the main panel. 2. Stop the job (call to XtreemOS job stop functionality).
Alternative paths	None.
Postconditions	If the job was successfully stopped, it will appear as "stopped" or similar.
Notes	None.
Author, date	TID team, November 2nd 2006

AUC JMA.5	Job's execution resume
Summary	This use case describes how the user can resume a stopped job.
Actors, Roles and Goals	**User**: an end user who has started the application. • **Role**: the end user of JobMA. • **Goal**: resuming a job that was previously stopped.
Preconditions	The user has the right permissions to resume the job's execution.
Triggers	Stopped jobs are shown on the main panel.
Basic course of events	**Nominal AUC JMA.5 # 1:** 1. Select the stopped job from the main panel. 2. Resume job's execution (call to XtreemOS job execution's resume functionality).
Alternative paths	None
Postconditions	If the job was successfully resumed, it will appear as "running" or similar.
Notes	None.
Author, date	TID team, November 2nd 2006

AUC JMA.6	Job cancel
Summary	This use case describes how the user can cancel a job, thus losing any intermediate results that had been obtained up to that moment. A job may be cancelled from any state but, obviously not if it's already cancelled.
Actors, Roles and Goals	**User**: an end user who has started the application. • **Role**: the end user of JobMA. • **Goal**: cancelling an existing job.
Preconditions	The user has the right permissions to cancel the job.
Triggers	Active jobs are shown on the main panel.
Basic course of events	**Nominal AUC JMA.6 # 1:** 3. Select the existing job from the main panel. 4. Cancel job's execution (call XtreemOS job cancel functionality).
Alternative paths	None.
Postconditions	If the job was successfully cancelled, it will appear as "cancelled" or similar. Later, the job should disappear from the main panel.
Notes	None.
Author, date	TID team, November 2nd 2006

AUC JMA.7	View job final state
Summary	This use case describes how the user can see if the job he launched has finished successfully or not, using JobMA.
Actors, Roles and Goals	**User**: an end user who has started the application. • **Role**: the end user of JobMA. • **Goal**: visualizing the final status of a job.
Preconditions	The user has the right permissions to view the job's final state.
Triggers	Finished jobs are shown on the main panel.
Basic course of events	**Nominal AUC JMA.7 # 1:** 1. Select the finished job from the main panel. 2. View the job's final state from menu. JobMA will show the relevant information about the final status of the job (at least, a boolean value marking whether the job finished successfully or not).
Alternative paths	None.
Postconditions	None.
Notes	The amount of time that the final state of the job must be kept will be determined by VO policies.
Author, date	TID team, November 2nd 2006

AUC JMA.8	View job results
Summary	This use case describes how the user can see any data generated by a finished job, using JobMA.
Actors, Roles	**User**: an end user who has started the application.

and Goals	• **Role**: the end user of JobMA. • **Goal**: visualizing the final results of a job.
Preconditions	The user has the right permissions to view the job's final results.
Triggers	Finished jobs are shown on the main panel.
Basic course of events	**Nominal AUC JMA.8 # 1:** 1. Select the finished job from the main panel. 2. View the job's final results from menu. JobMA will raise a window to explore the job's final results (probably stored in a file).
Alternative paths	None.
Postconditions	None.
Notes	None.
Author, date	TID team, November 2nd 2006

3.2.11Wissenheim (UDUS)

AUC WISS.1	Virtual Collaboration
Summary	A Virtual meeting should take place and the chair of the meeting initializes the virtual meeting room in the Grid.
Actors, Roles and Goals	**Chair**: • **Role**: creates a virtual room, manages the meeting • **Goal**: hold a meeting **Participant**: • **Role**: joins the virtual meeting by entering the virtual room • **Goal**: attend a meeting **Grid**: • **Role**: Host the virtual room process • **Goal**: provides transparent access for all participants
Preconditions	• Every user is connected to the Grid via a XtreemOS enabled computer and meets the requirements of the Wissenheim graphical subsystem.
Triggers	Meeting
Basic course of events	**Nominal AUC WISS.1 :** 1. Chair creates virtual room, injects the virtual room process into the Grid. 2. Participants enter the meeting by connecting to the virtual room. 3. Virtual room process relocates itself automatically to provide the best performance for all participants using the information provided by the XtreemOS (latency, throughput etc.)
Alternative paths	**Alternative AUC WISS.1 :** Process crashes due to an exceptions, segmentation fault, etc. 1. Checkpointing facilities of XtreemOS recovers last saved state and restarts the process 2. Participants' processes are unaware of failure, only feedback to the user should be a higher latency or short stalling until XtreemOS has recovered the process from the failure. **Alternative AUC WISS.1 :** Node hosting the room process crashes 1. Checkpointing facilities of XtreemOS recover last saved state and restarts the process on a different node without user interaction. 2. Participants' processes are unaware of failure, only feedback to the user should be a higher latency or short stalling until XtreemOS has recovered from the failure.
Postconditions	As soon as the meeting is over, the chair closes the virtual room by killing the process.
Notes	None
Author, date	Michael Sonnenfroh (University of Düsseldorf), 03.11.2006

AUC WISS.2	Virtual Worlds scenario
Summary	The Wissenheim project should consist of multiple persistent worlds accessible to users. Worlds are a means of partitioning the load and the content.
Actors, Roles and Goals	**Participant**: • **Role**: enters and interacts with a world • **Goal**: interactive teaching content, fun and leisure **World:** • **Role**: process managing the participants in a world • **Goal**: provides location independent access via XtreemOS
Preconditions	– Every user is connected to the Grid via a XtreemOS enabled computer and meets the requirements of the Wissenheim graphical subsystem.
Triggers	User connects to a virtual world.
Basic course of events	**Nominal AUC WISS.2 :** 1. User is connecting to the world. 2. World process might migrate to a location nearer to the user to optimize latency. 3. World process might be migrated to a user computer for cost efficiency. Reducing bandwidth needed for hosting the world.
Alternative paths	**Alternative AUC WISS.2 :** Failure of the node hosting the virtual world process 1. Checkpointing facilities of XtreemOS recovers last saved state and restarts the process 2. Participants' processes are unaware of failure, only feedback to the user should be a higher latency or short stalling until XtreemOS has recovered from the failure. **Alternative AUC WISS.2 :** Node hosting the room process crashes 1. Checkpointing facilities of XtreemOS recover last saved state and restarts the process on a different node without user interaction. 2. Participants' processes are unaware of failure, only feedback to the user should be a higher latency or short stalling until XtreemOS has recovered from the failure.
Postconditions	World process may survive beyond the user processes, all data must be persistent.
Notes	None
Author, date	Michael Sonnenfroh (University of Düsseldorf), 03.11.2006

3.2.12Galeb (XLAB)

As Galeb is a command line driven, non-interactive utility, there is a single use case, in which a user uses the Galeb framework to construct an analytical model of a time series.

AUC Galeb.1	Constructing an analytical model of a time series
Summary	The user provides a time series, for which the application builds an analytical model.
Actors, Roles and Goals	**User**: the end user of the application • **Role**: end user • **Goal**: to construct a model for his time series
Preconditions	• The user must prepare input (the time series) in the form of a text file. • The user must be logged in to the Grid and must be authorized to access the data and to run the application. (In the current, Globus-based version, the user must first provide authentication and authorization information for the Galeb Web Service container running on the master node by running the Grid-proxy-init command of GT4 and supplying it with a proper user certificate.)
Triggers	User command
Basic course of events	**Nominal AUC Galeb.1:** 1. The user starts the application from the command line, passing it the name of the input file and a number of configuration parameters, e.g. `java xlab.Grid.GAlebExecute.GAlebExecuter in inputData.txt out model.txt popsize 150 pmut 0.1 num_repetitions 5` 2. The application starts and distributes the computation to the allocated nodes. 3. During runtime, the user is notified about progress, nodes used etc. 4. The best analytical model found is written to a text file. 5. The application ends.

Alternative paths	**Alternative AUC Galeb.1 # 1 :** input file or parameter invalid 2. After step 1, an error message is shown. 3. Application fails. **Alternative AUC Galeb.1 # 2 :** no computer available for processing 2. Error message is shown. 3. Application fails.
Postconditions	The output text file contains an analytical model.
Notes	None
Author, date	Marjan Sterk and Jaka Mocnik (XLAB), November 6th 2006.

4 Requirements

This section presents the application requirements for XtreemOS. These requirements are structured as follows:

- General requirements
- Virtual organization support in XtreemOS
- Checkpointing and restart
- Federation management
- XtreemOS interfaces
- Higly available and scalable Grid services
- Data management
- Security in virtual organizations
- Support for mobiles devices

Although this structure is closely related 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. The requirements are presented by their requirement number (Rx), a requirement title and a detailed description of the respective requirement. Furthermore, we indicate from which positions in the questionnaire each requirement has been derived. Two rankings per requirement are provided:

- Importance: "Obligatory" indicates that the respective requirement must be implemented to ensure that the applications can be executed, whereas "Optional" means that the requirement is useful but the applications can still be executed without the requested functionality.
- 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 during which phases of the project lifetime the requested functionality should be provided. Note that we do not intend to specify a kind of graph that describes the suggested implementation order as this would require additional knowledge about the dependencies within the different work packages of SP2 and SP3. Thus, we only intend to provide hints from the application perspective on the different XtreemOS aspects.

4.1 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 (lSPEC-WEB, HLA, SIMEON) are both whereas SRC, TIFON 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).

Questions referenced: Q2.1
Importance: **Obligatory**
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 31.

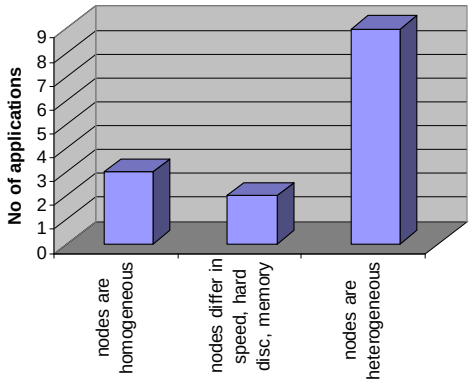


Figure 31: Hardware diversity

Some applications are programmed to be platform-independent and do therefore support heterogeneous architec-

tures. Certain applications like ELFIPOLE or ZEPHYR benefit from homogeneous architectures to facilitate runtime prediction.
Questions referenced: Q2.5
Importance: **Obligatory**
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.

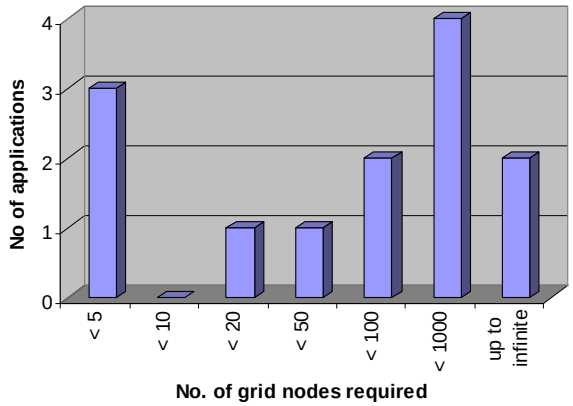


Figure 32: Number of Grid nodes required by the applications

Figure 32 demonstrates that 5 applications require relatively "small" Grids whereas the majority of applications may also demand Grids with up to thousands of nodes.
Questions referenced: Q2.6
Importance: **Obligatory**
Implementation order: **3.5**

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 are 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 R41 for a similar mechanism on a 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.
Questions referenced: Q2.7
Importance: **Obligatory**
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 following requirement). This capability to dynamically react to a changing number of Grid nodes is required by about half of the applications.

Questions referenced: Q2.8
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.

Questions referenced: Q2.8
Importance: **Obligatory**
Implementation order: **3.7 (corresponds to previous requirement)**

R7 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.

Questions referenced: Q2.9
Importance: **Obligatory**
Implementation order: **8.1**

R8 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.

Questions referenced: Q2.10
Importance: **Obligatory**
Implementation order: **4.6**

R9 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.

Questions referenced: Q2.11
Importance: **Obligatory**
Implementation order: **6.0**

R10 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. One application (HLA) also exhibits a near real-time requirement. Reliable connectivity is also a crucial requirement by various applications, which may be tremendously affected by network failures (see Figure 33).

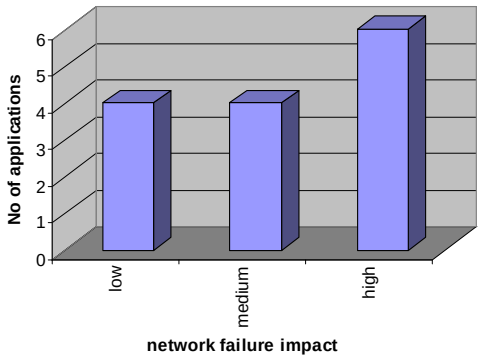


Figure 33: Expected impact of network failures

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 to be considered by XtreemOS

Questions referenced: Q2.12, Q2.13, Q2.14, Q2.15
Importance: **Obligatory**
Implementation order: **3.4**

R11 XtreemOS must support IPv6

The XtreemOS must support IPv6.
Questions referenced: Q2.16
Importance: **Obligatory**
Implementation order: **7.5**

R12 XtreemOS must support 64 bit architectures

Since almost no pure 32 bit systems are sold anymore in today's HPC market it is essential that XtreemOS supports x86_64 architectures. Furthermore several upcoming releases of applications will only be executable on 64-bit processors.
Questions referenced: Discussion
Importance: **Obligatory**
Implementation order: **2.4**

R13 XtreemOS must support virtual machines

This requirement is actually twofold. First, the XtreemOS hardware layer should be able to cope with virtualized hardware, e.g. XEN Hypervisor or VMWare. Second, XtreemOS should provide virtual hosts/machines for VO support.
Questions referenced: Discussion
Importance: **Obligatory**
Implementation order: **6.0**

R14 XtreemOS shall support multicast

It is required that XtreemOS supports multicast across clusters.
Questions referenced: Q2.17
Importance: **Optional**
Implementation order: **8.0**

R15 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 34.

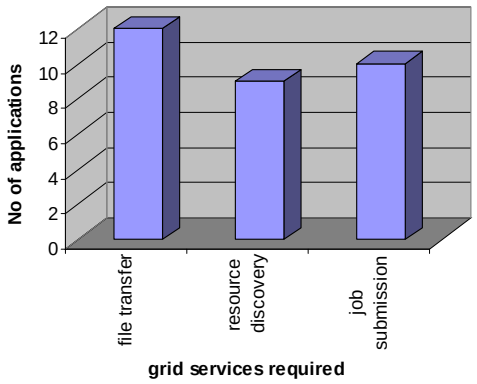


Figure 34: Main Grid services required

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 35).

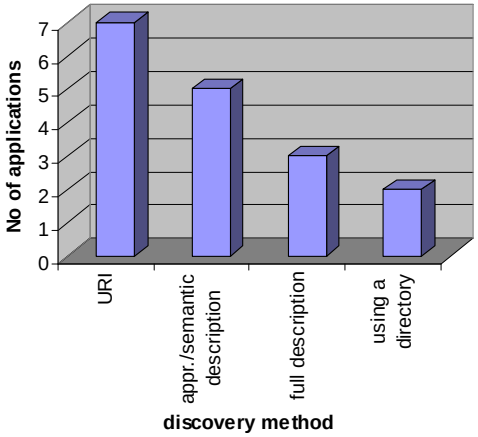


Figure 35: Methods for resource and service discovery

Questions referenced: Q2.18, Q2.19, Q2.20
Importance: **Obligatory**
Implementation order: **2.8**

R16 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.

Questions referenced: Q2.21, Q2.22, Q2.23
Importance: **Obligatory**
Implementation order: **4.7**

R17 XtreemOS must support the execution of interactive and batch jobs

The majority of applications in WP4.2 is executed interactively as shown in Figure 36. 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 both interactive applications and batch jobs.

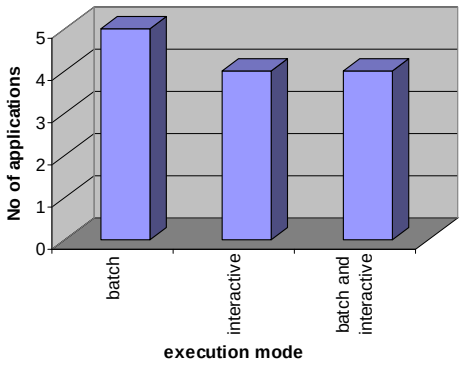


Figure 36: Application execution modes

Questions referenced: Q2.24
Importance: **Obligatory**
Implementation order: **2.2**

R18 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). Middleware systems like WebAS also require a complex role management at the application layer.

Questions referenced: Q2.27, Q2.28
Importance: **Obligatory**
Implementation order: **7.2**

4.2 Virtual Organization Support in XtreemOS

Virtual Organizations (VOs) are composed of resources as agreed in bilateral contracts between institutions providing these resources.

R19 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 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.

Questions referenced: QWP2.1.1
Importance: **Obligatory**
Implementation order: **2.5**

R20 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).

Questions referenced: QWP2.1.1
Importance: **Obligatory**
Implementation order: **5.2**

R21 XtreemOS allows to establish and manage hierarchies of VOs.

Hierarchies of VOs can either be created from scratch or by dynamically establishing sub-VOs. Suppose that a program P running in VO1 creates a new VO2. If VO2 was independent of VO1 this would represent a security hole and would facilitate the execution of malicious applications. Therefore it must be possible to establish VO2 as a sub-VO of VO1, with the same or limited rights. This means a hierarchy of VOs is required instead of just one layer. Sub-VOs can be defined as including a subset of VO1 users and have access to a subset of VO1 resources. Local admins can configure individual resources in VO1 so that access to any sub-VO of VO1 is granted

automatically. Otherwise, access from VO2 to resources must be granted individually after VO2 has been created.
Questions referenced: Discussions
Importance: **Obligatory**
Implementation order: **8.0**

R22 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.
Questions referenced: QWP2.1.2, QWP2.1.3
Importance: **Obligatory**
Implementation order: **7.4**

R23 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.
Questions referenced: QWP2.1.4
Importance: **Obligatory**
Implementation order: **5.2**

R24 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)
Questions referenced: QWP2.1.6
Importance: **Obligatory**
Implementation order: **7.0**

R25 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.
Questions referenced: QWP2.1.7, QWP2.1.8
Importance: **Obligatory**
Implementation order: **4.6**

R26 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 (HLA, 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).
Questions referenced: QWP2.1.9
Importance: **Obligatory**
Implementation order: **7.3**

R27 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.
Questions referenced: QWP2.1.10, QWP2.1.11
Importance: **Obligatory**
Implementation order: **7.1**

4.3 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.

R28 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.
Questions referenced: Q2.1.12, Q2.1.13, Q2.1.14
Importance: **Obligatory**
Implementation order: **4.7**

R29 Restart must mimic the original environment
XtreemOS must provide the illusion of the original environment considering in particular for 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.
Questions referenced: Q2.1.16, Q2.1.17, Q2.1.18
Importance: **Obligatory**
Implementation order: **5.7**

R30 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.
Questions referenced: Q2.1.16, Q2.1.17, Q2.1.18
Inportance: **Obligatory**
Implementation order: **4.6**

R31 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.
Questions referenced: QWP2.1.19
Importance: **Obligatory**
Implementation order: **7.4**

R32 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.
Questions referenced: QWP2.1.20
Importance: **Obligatory**
Implementation order: **7.0**

R33 Checkpointing/restart must be implemented as kernel module
Checkpointing and restart must be implemented on OS level as kernel module.
Questions referenced: QWP2.1.23
Importance: **Obligatory**
Implementation order: **6.4**

R34 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.
Questions referenced: QWP2.1.22, QWP2.1.23, QWP2.1.24
Importance: **Obligatory**
Implementation order: **7.1**

R35 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.
Questions referenced: QWP2.1.24-QWP2.1.28
Importance: **Obligatory**
Implementation order: **6.5**

4.4 Federation Management

R36 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 TIFON and JOBMA, which use one node only). Figure 37 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.

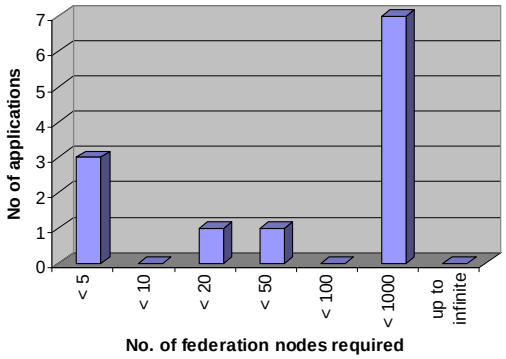


Figure 37: Maximum number of federation nodes required by the applications

Questions referenced: QWP2.2.2, QWP2.2.3 (relates to Q2.6)
Importance: **Obligatory**
Implementation order: **2.3**

R37 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. Resources that do not fulfill the defined service qualities must not be used by the application. XtreemOS also allows applications to specify a topology of the resources which provide the best performance.

Questions referenced: QWP2.2.4 (relates to Q2.13, 2.14)
Importance: **Obligatory**
Implementation order: **5.3**

R38 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.

Mechanisms/suggestions

The XtreemOS API must provide means to specify the constraints when starting the application. Resources not fulfilling the constraints must not be used for the execution.

Questions referenced: QWP2.2.5
Importance: **Obligatory**
Implementation order: **5.5**

R39 Shared file system within a federation

XtreemOS must offer shared file system within federations. Also related to Section 4.8.

Questions referenced: QWP2.2.7
Importance: **Obligatory**
Implementation order: **3.9**

R40 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.
Questions referenced: QWP2.2.8, QWP2.2.9, QWP2.2.10
Importance: **Obligatory**
Implementation order: **4.4**

R41 Virtual nodes in federations
It must be possible to replicate processes on multiple federation nodes to increase robustness in case of resource failures, similarly to the grid-level virtual nodes mechanism. (see also R4).

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.
Questions referenced: QWP2.2.6, QWP2.2.11 (related to Q2.7)
Importance: **Obligatory**
Implementation order: **8.3**

R42 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 4.3.
Questions referenced: QWP2.2.12 - QWP2.2.24
(all related to corresponding requirements in Section 4.3)
Importance: **Obligatory**
Implementation order: **5.3**

4.5 XtreemOS Interfaces

R43 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.

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.
Questions referenced: QWP 3.1.1
Importance: **Obligatory**
Implementation Order **5**

R44 Demand for POSIX like extension
Mandatory access control (ACL) as defined e.g. in POSIX .1e, IEEE 1003.1e/2c (which was withdrawn). Note, all 28 calls must be provided. Furthermore, it would prove useful if functions for management of processes on remote machines would be part of the XtreemOS API.
Questions referenced: QWP 3.1.2
Importance: **Obligatory**

Implementation Order **5.3**

R45 XtreemOS API language support
XtreemOS must support C, C++, Java, and Fortran 77. Furthermore, Ada, Python and Perl should be supported. Fortran 77 can be supported via C bindings.
Questions referenced: QWP 3.1.3
Importance: **Obligatory**
Implementation Order **3.3**

R46 Degree of Interoperability
It should be possible to use XtreemOS as a backend for GT4 WS-GRAM.
Questions referenced: QWP 3.1.5
Importance: **Optional**
Implementation Order **6.4**

4.6 Highly Available and Scalable Grid Services

R47 Guaranteed number of nodes for a specified time
In order to execute parallel applications on top of XtreemOS, it is necessary to guarantee that the number of allocated nodes is constant during the whole job execution. To this end, a fault-tolerance mechanism termed virtual nodes is necessary that is replicating specified nodes. However, this replication has to be specified explicitly by the user or application.
Questions referenced: QWP3.2.2
Importance: **Obligatory**
Implementation Order **3.1**

R48 Possibility to have a guaranteed bandwidth between the allocated nodes during the entire runtime of applications
For certain applications, a minimum bandwidth 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 used bandwidth with high precision.

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.

Questions referenced: QWP3.2.2
Importance: **Obligatory**
Implementation Order **5**

R49 Priority in the availability of certain services on the Grid
We gathered the average ranking of the services that must be implemented:

Service	Priority
Authentification	7.7
Data management	7.5
Security	7.3
Job monitoring	6.7
Application deployment	6.8
Resource scheduling	5.8
Resource monitoring	5.8
Scalability of the whole Grid	5.6
Resource allocation	5.6
Decentralization	5.3
Resource exploration	5.2
Network allocation	5.0
Network monitoring	4.9

Service Level Agreements	4.9
Grid services for mobile devices	4.7
Migration services	3.9

Remarks:

- 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 TIFON 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.

Questions referenced: QWP3.2.1
Importance: **Obligatory**
Implementation Order **4.8**

R50 Relative importance of the measurement criteria to estimate the quality of Grid services

We gathered the average relative ranking of the importance of certain measurements to estimate the quality of the Grid services.

Quality of Grid service	**Measurement**
Number of nodes failed during runtime	6.8
Ability to reserve resources in advance	6.2
Number of timeouts waiting for data	6.1
Time for authorization	5.6
Number of not satisfied Service Level Agreements	4.7
Time to connect mobile devices	4.0
Services to manage Virtual Organization	3.9
Time for deployment	3.7
Time for job migration	3.6
Time for negotiation	3.5
Time for scheduling	3.1
Time to set up Virtual Organizations	2.8
Migration services	2.9
Migration time	2.6
Time for exploring appropriate resources	2.5

Remarks:

- We can classify the measurements into four different groups with varying importance to the application:
  - The easiness to match the required and available resources and to use them properly once allocated (number of nodes failed during runtime; ability to reserve resources in advance; number of timeouts waiting for data)
  - The time needed to grant access to these resources (time for authorization; number of not satisfied Service Level Agreements; time to connect mobile devices) Service Level Agreements describe in general the quality of service for the different Grid services.
  - The time before having an effective application running on the system (services to manage Virtual Organization; time for deployment; time for scheduling)
  - The ability to migrate and dynamically adapt to available resources (time to set up a Virtual Organization; migration services; migration time; time for exploring appropriate resources)
- Each one of these four groups can be analyzed like decisive for one application. Statistically speaking, it seems fair to use the table of average values to gather a privileged order to improve the measurements available. But we cannot use it as scale of importance within the different measurements.
- Grid services must be established that enable applications and users to gather those information.

Questions referenced: QWP 3.2.3
Importance: **Obligatory**
Implementation Order **5.3**

4.7 Application Execution Management

R51 Monitoring System

The monitoring system must provide information about running applications (either application or user defined). The resource consumption should be viewable during the whole application execution and there should be a message or mail notification for changes in the application, workflow, or environment. Furthermore, notifications are necessary in 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).

Half of the applications require a notification by email and the other half of the applications needs a callback function.

Questions referenced: Q3.3.3, Q3.3.17, Q3.3.24
Importance: **Obligatory**
Implementation Order **3.7**

R52 Resource Accounting

It must be possible to record the usage of a resource by a user at any given time. A way to implement / use special cost models must also be provided.

Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen.

Figure 38: Rating for resource accounting

Questions referenced: Q3.3.26, Q3.3.27, Q3.3.28
Importance: **Obligatory**
Implementation Order **5**

R53 The kind of monitoring data that the applications need to query

The monitoring system must provide the following monitoring data to several applications:

- **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
- **Job's resource usage:** job memory usage, 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

Questions referenced: Q3.3.26, Q3.3.27, Q3.3.28
Importance: **Obligatory**
Implementation Order **3.7**

R54 The kind of information that can be queried from the command line while the application is running

Several applications require to gather the following information from the command line during the runtime of the application:

- **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
- **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 memory usage, 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

Questions referenced: Q3.3.26, Q3.3.27, Q3.3.28
Importance: **Obligatory**
Implementation Order **3.4**

R55 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, so 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.

Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen.
Figure 39: Rating tracing of application execution vs. tracing of resource

Questions referenced: Q3.3.4, Q3.3.5, Q3.3.13
Importance: **Obligatory**
Implementation Order **4.2**

R56 Kind of events required to trace from the application's execution

The system must trace the following events:

- **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)

Questions referenced: Q3.3.11, Q3.3.12
Importance: **Obligatory**
Implementation Order **4.2**

R57 Kind of state changes that must be traced from the application's execution

The system must trace the following changes:

- **Grid scheduling states:** job waiting, job running, job finished, job migrating
  Kernel level states: thread blocked, thread running userspace, thread running kernel space
- **Communications/parallel library events:** waiting data, sending data, waiting on synchronisation point (barriers)
- **Resource events:** CPU running, CPU idle

Questions referenced: Q3.3.11, Q3.3.12
Importance: **Obligatory**
Implementation Order **4.3**

R58 Trace Format

XtreemOS must provide support for the Paraver format and must develop an XtreemOS specific trace format.
Questions referenced: Q3.3.11, Q3.3.12
Importance: **Obligatory**
Implementation Order **5.7**

R59 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.
Questions referenced: Q3.3.11, Q3.3.12
Importance: **Obligatory**
Implementation Order **5.4**

R60 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)
Note that 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 prove that workflow scheduling is possible. To this end, a workflow application runs on top of XtreemOS.
Questions referenced: Q3.3.6, Q3.3.14, Q3.3.19, Q3.3.20, Q3.3.22
Importance: **Obligatory**
Implementation Order **5.4**

R61 Stopping execution

It should be possible to stop the execution of an individual application as well as the execution of an entire workflow. Again, the workflow management might be implemented by a workflow management tool on top of XtreemOS.

Questions referenced: Q3.3.15, Q3.3.16
Importance: **Obligatory**
Implementation Order **3.2**

R62 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.

Questions referenced: Q3.3.18
Importance: **Obligatory**
Implementation Order **8.3**

R63 Spawn jobs to other Vos

An application running on a virtual organization can spawn a job to another virtual organization.

Questions referenced: Q3.3.7
Importance: **Obligatory**
Implementation Order **7.9**

R64 Manual exploration and selection of hardware

The application needs to be able to select the resources it uses.

Questions referenced: Q3.3.23
Importance: **Obligatory**
Implementation Order **5.1**

R65 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.

Questions referenced: Q3.3.8, Q3.39, Q3.3.10
Importance: **Obligatory**
Implementation Order **4**

R66 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.).

Questions referenced: Q3.3.21
Importance: **Obligatory**
Implementation Order **6.9**

4.8 Data Management

This section describes relatively few concrete requirements. Rather, the data aspects of the applications are given, which should be taken into consideration when designing the data management facilities.

The answers to the questions not describing concrete requirements are given first, followed by explicitly stated requirements. Note that a lot of questions (QWP3.4.4-QWP3.4.20) only apply to the 12 file-based applications.

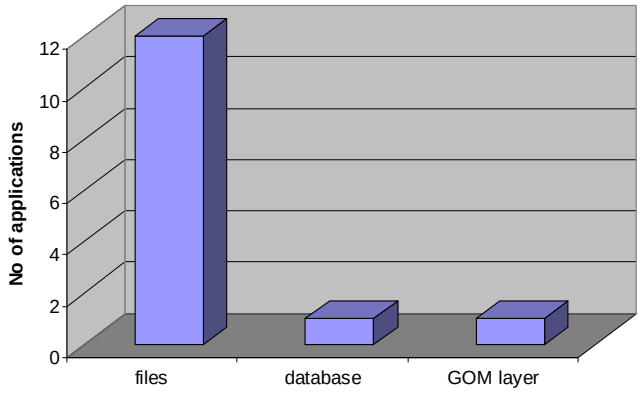


Figure 40: Files, data bases, Global Object Management (GOM) layer

The application that uses a database uses one database whose size is ca 16 GB – 6 TB.

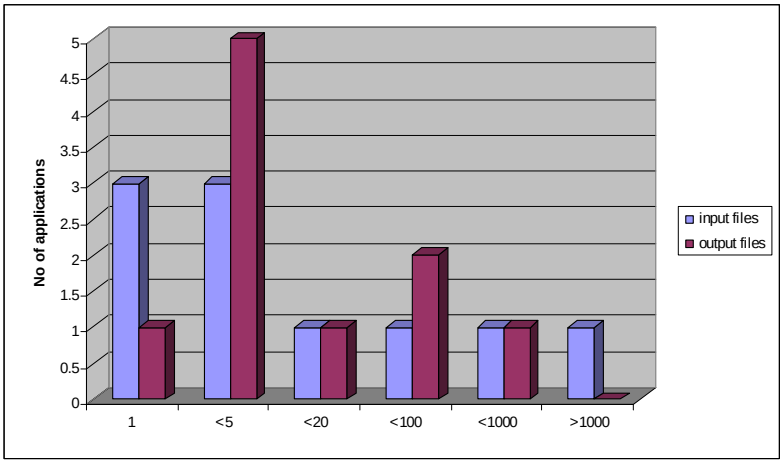


Figure 41: Number of input and output files

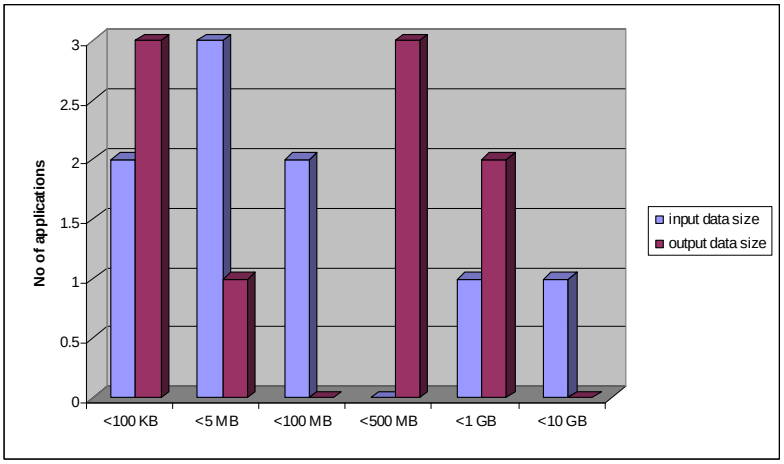


Figure 42: Estimation of the size of input and output data in MB

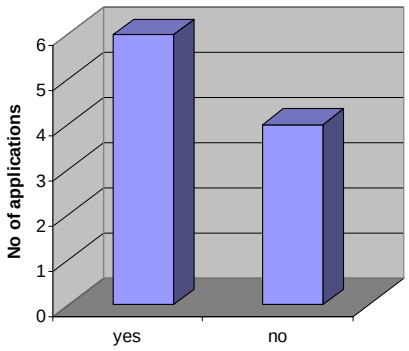


Figure 43: Can all input and output files be specified at initialization?

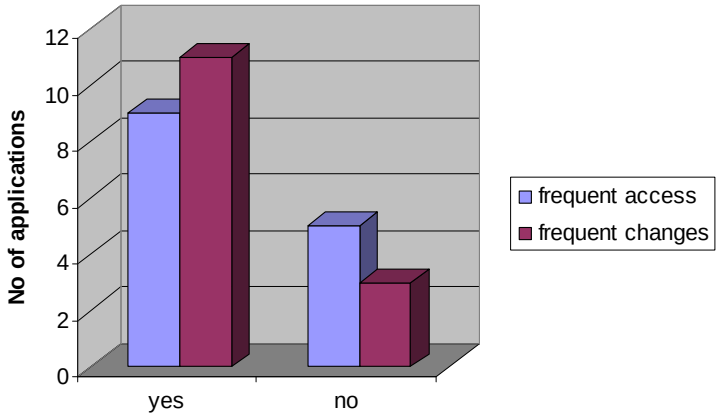


Figure 44: Are applications accessing the corresponding data frequently?

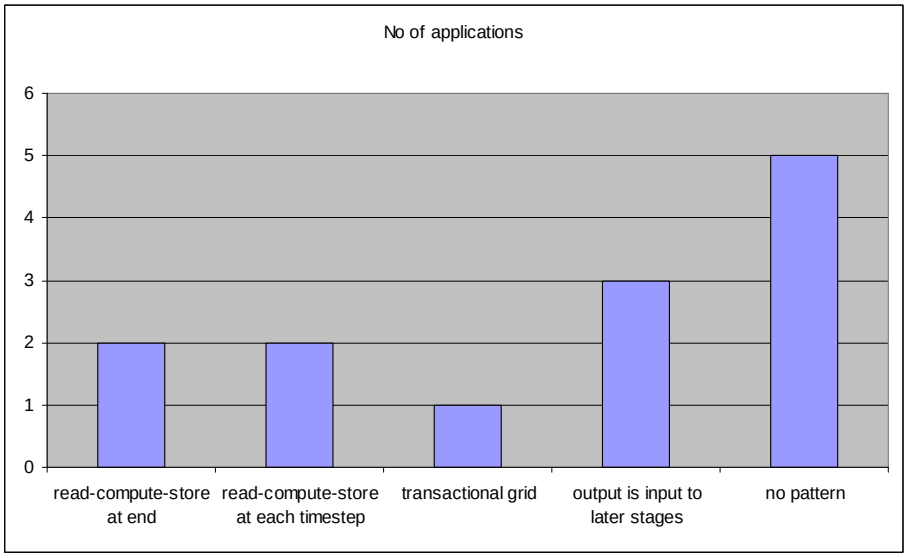


Figure 45: Data workflow and data access patterns

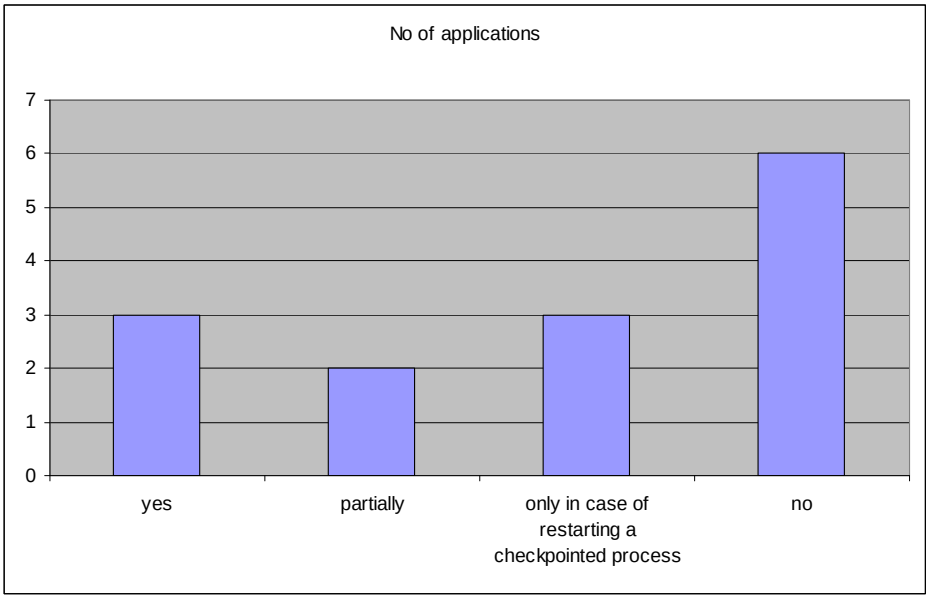


Figure 46: Pre-fetching of application data before job execution

R67 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.

Questions referenced: Q3.4.9
Importance: **Obligatory**
Implementation Order **3.7**

R68 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.

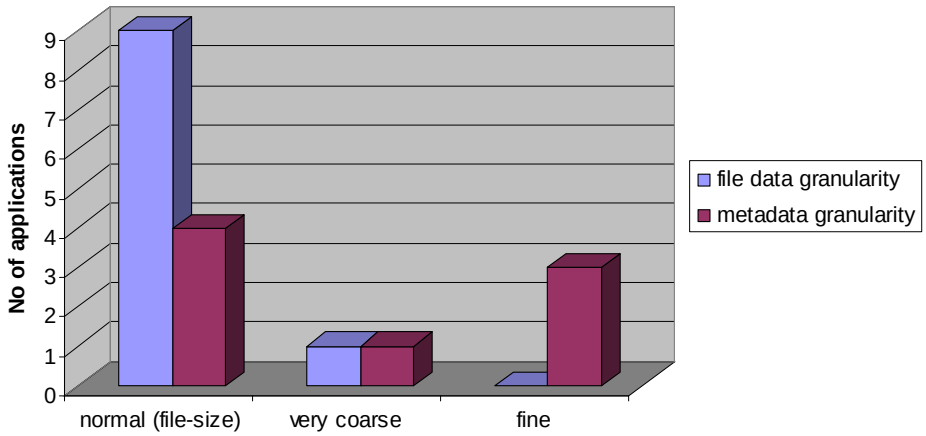
Questions referenced: Q3.4.12
Importance: **Obligatory**
Implementation Order **4.1**

R69 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.

Questions referenced: Q3.4.13
Importance: **Obligatory**
Implementation Order **4.5**

R70 File metadata access control granularity
It must be possible to set the access rights on parts of individual metadata.



Questions referenced: Q3.4.16, Q3.4.17
Importance: **Obligatory**
Implementation Order **8**

R71 File data and metadata change monitoring/notification
It must be possible to check for file data and metadata changes. It must also be possible to subscribe to the information on file data and metadata changes, e.g. through registering a callback function.
Questions referenced: Q3.4.18, Q3.4.19
Importance: **Obligatory**
Implementation Order **7.4**

R72 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 frequently.
Questions referenced: Q3.4.28
Importance: **Obligatory**
Implementation Order **4.3**

R73 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.
Questions referenced Q3.4.28
Importance: **Obligatory**
Implementation Order **6.5**

R74 Data versioning
The versioning of data to allow incremental changes is required.
Questions referenced: Q3.4.29
Importance: **Obligatory**
Implementation Order **8.9**

R75 Explicit move/copy between resources
It must be possible to explicitly copy/move data between different resources.
Questions referenced: Q3.4.8
Importance: **Obligatory**
Implementation Order **4.7**

R76 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.
Questions referenced: Q3.4.30, Q3.4.33
Importance: **Obligatory**
Implementation Order **6**

R77 Data transmission monitoring
A facility to monitor ongoing data transmission must be provided.
Questions referenced: Q3.4.31
Importance: **Obligatory**
Implementation Order **3**

4.9 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 [Foster et al 2001, The Anatomy of the Grid]. This will be referenced within each requirement specification by way of clarification.

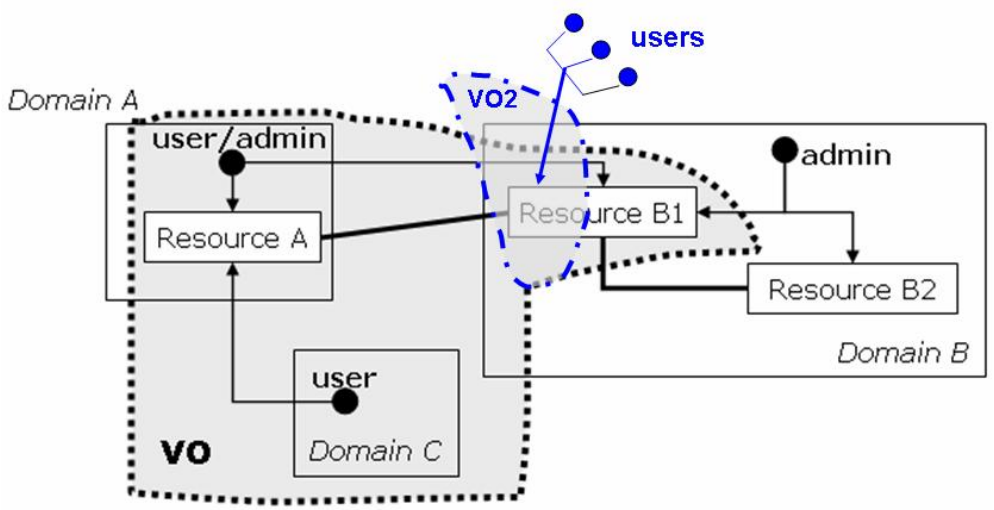


Figure 47: A generic architecture for distribution, usage and administration of resources in a Grid environment

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 are therefore, if XtreemOS 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**

R78 **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 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 re-

source 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.

Importance: **Obligatory**
Implementation Order: **3.9**

R79 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 25% responded, 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.

Importance: **Obligatory**
Implementation Order **3.1**

R80 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.

Importance: **Obligatory**
Implementation Order **5.1**

R81 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 a 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.

Importance: **Obligatory**
Implementation Order **4.9**

R82 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 assert 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.
Importance: **Optional**
Implementation Order **4.8**

R83 **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.
Importance: **Obligatory**
Implementation Order **5.4**

R84 **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.
Importance: **Obligatory**
Implementation Order **4.4**

R85 **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.
Importance: **Optional**
Implementation Order **6.3**

R86 **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.
Importance: **Obligatory**
Implementation Order **5.6**

R87 **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 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 partici-

pants 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 multilayered 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.

Importance: **Obligatory**
Implementation Order **5.2**

R88 **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 automic 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.

Importance: **Obligatory**
Implementation Order **5.4**

R89 **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)

Importance: **Optional**
Implementation Order **1.9**

R90 **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.

Questions referenced: Q3.5.28, Q3.5.30
Importance: **Obligatory**
Implementation Order **6.0**

R91 **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

Questions referenced: Q3.5.32
Importance: **Obligatory**
Implementation Order **5.0**

R92 **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

Questions referenced: Q3.5.33
Importance: **Obligatory**
Implementation Order **4.0**

R93 **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%.

Questions referenced: Q3.5.34
Importance: **Obligatory**
Implementation Order **7.2**

R94 **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

Questions referenced: Q3.5.31
Importance: **Obligatory**
Implementation Order **5.2**

4.10 Support for Mobile Devices

Note that the implementation orders for requirements R94 to R99 were rated by less than 50% of the applications. Therefore the results may not be representative.

R95 **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.

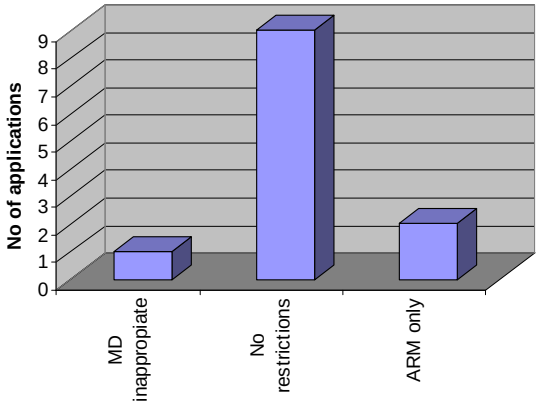


Figure 48: Typical executing environment

Questions referenced: QWP2.3.1
Importance: **Obligatory**
Implementation Order : **5.0**

R96 **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.

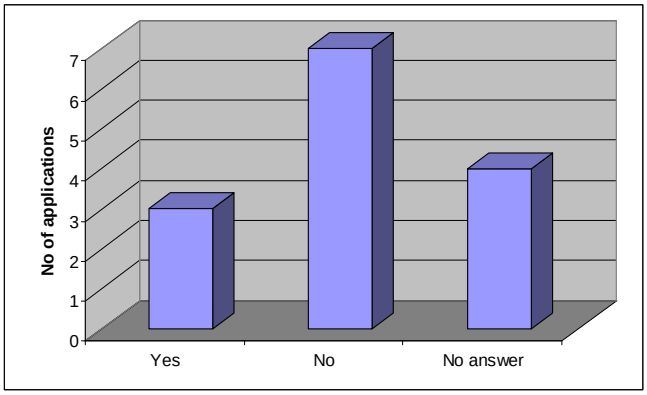


Figure 49: Java support

Questions referenced: QWP2.3.2
Importance: **Obligatory**
Implementation Order : **5.3**

R97 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.

Questions referenced: QWP2.3.2
Importance: **Obligatory**
Implementation Order : **5.5**

R98 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.

Questions referenced: QWP2.3.3
Importance: **Optional**
Implementation Order : **8.4**

R99 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.

Questions referenced: QWP2.3.4, QWP3.6.3, QWP3.6.4, QWP3.6.5
Importance: **Obligatory**
Implementation Order : **4.2**

R100 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.

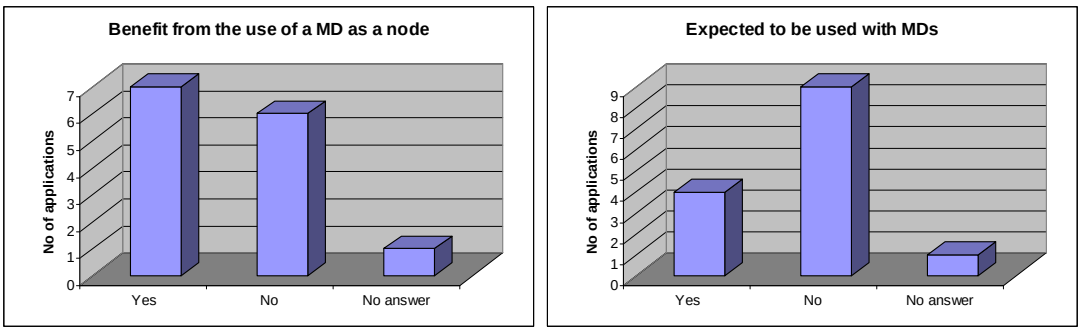


Figure 50: Usage of MDs

Thus, apart from being maked 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.
Questions referenced: QWP3.6.1, QWP3.6.2
Importance: **Obligatory**
Implementation Order : **6.5**

R101 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.
Questions referenced: QWP3.6.6
Importance: **Optional**
Implementation Order : **8.8**

5 Conclusion

This deliverable presented the requirements to XtreemOS from the applications' point of view. In total twelve reference applications with typical use case scenarios were analyzed by the members of WP4.2.

From this set of questions general requirements were deduced in a reproducible manner (the complete questionnaire can be found in the Appendix). These requirements are forming the base for a further requirement refinement phase. They allow to decide on architectural decisions for XtreemOS and to select dedicated frameworks and technologies in future deliverables.

The requirements themselves are organized in areas of concern and were discussed with dedicated consortium partners. The areas of concern cover important operating system tasks and responsibilities but pays special attention to the support of Virtual Organizations, management, scalability and robustness, interfaces, data and storage management and security.

Teams in WP4.2 also rated the requirements with respect to the importance and implementation order for each of their applications. Although the reference application scenarios are very different, the responses to the importance and implementation order of the requirements show a similar assessment. Merely the requirements on XtreemOS for mobile devices showed inconsistencies in the importance of the requirements. This can be explained by the fact that for most reference applications mobile devices play a marginal role.

In this deliverable, the application use cases and requirements have been captured as far as these can be specified during the initial six project months. They are considered as a source of inspiration, as a guideline as well as a basis for quality management. We wish to emphasize that the requirements are not static but will be revised, extended and modified where appropriate driven by the evaluations in WP4.2 and by the activities in SP2 and SP3.

6 References

[Wall96] M. Wall. Galib. A C++ Library for Genetic Algorithm Components. Massachusetts Institute of Technology, 1996

[Koza95] J. R. Koza. Genetic programming for economic modeling. In S. Goonatilaje and P. Treleaven, editors, *Intelligent Systems for Finance and Business*, pages 251-269. Wiley, 1995.

[Foster05] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In *Proceedings of the IFIP International Conference on Network and Parallel Computing*, volume 3779 of *LNCS*, pages 2-13, 2005.

[Czajkowski04] K. Czajkowski et al. The WS-Resource Framework, *The Globus Alliance*, 2004. http://www.globus.org/wsrf/specs/ws-wsrf.pdf

[GTK4Sec] Globus *Toolkit Version 4 Grid Security Infrastructure: A Standards Prospective*, The Globus Security Team. http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf

[OASIS06] OASIS – Web Services Resource 1.2, *WS-Resource Committee Specification*, January 2006. http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-cs-01.pdf

[Frey05] Frey G.: *JAVA Programming With the SAP Web Application Server*. SAP Press, 2005.

[Goebel05] Goebel A. and Ritthaler D.: *SAP Enterprise Portal: Technology and Programming*. SAP Press, 2005.

[Heine05] Heinemann F. and Rau C.: *Web Programming in ABAP with the SAP Web Application Server (2nd Edition)*.SAP Press, 2005.

[Karch05] Karch S. and Heilig L.: SAP NetWeaver Roadmap. SAP Press, 2005.

[Weil05] Weilbach J.: SAP *xApps and the Composite Application Framework*. SAP Press, 2005.

[SAPAG] SAP AG: www.sap.com

[SAPLin] SAP on Linux: www.sap.com/linux

[SPECWeb] SPECweb2005 homepage, http://www.spec.org/web2005/

[Dialinos05] Vasilis Dialinos, Rosa M. Badia, Raül Sirvent, Josep M. Perez and Jesús Labarta, *Implementing Phylogenetic Inference with GRID superscalar*, Cluster Computing and Grid 2005 (CCGRID 2005), Cardiff, UK, 2005.

[Badia03] Rosa M. Badia, Jesús Labarta, Raül Sirvent, Josep M. Pérez, José M. Cela, and Rogeli Grima, *Programming Grid Applications with GRID superscalar*, Journal of Grid Computing, Vol. 1, No. 2, 151–170, June 2003.
[GRIDsup] GRID superscalar homepage, http://cepba.upc.es/Grid

[Felsen81] J. Felsenstein, *Evolutionary trees from DNA sequences: A maximum likelihood approach*., J. Mol. Evol. 17: 368-376, 1981.

[Olsen94] G.L. Olsen, H. Matsuda, R. Hagstrom, and R. Overbeek, *fastDNAml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood.*, Comput. Appl. Biosci. 10: 41-48, 1994.

[Stewart01] C. A. Stewart, D. Hart, D. K. Berry, E. A. Wernert, and W. Fischer, *Parallel implementation and performance of fastDNAml - a program for maximum likelihood phylogenetic inference*, proceedings of SC01, 2001.

[DBE] Digital Bussiness Ecosystem official site, http://www.digital-ecosystem.org/

[Servent] SUN Servent official site, http://swallow.sourceforge.net

[DBEStu] DBE Studio official page, http://dbestudio.sourceforge.net

7 Appendix

The Appendix is published in a separate document.