XtreemOS: a Grid Operating System Making your Computer Ready for Participating in Virtual Organizations

Christine Morin*
PARIS Project-team
XtreemOS Scientific Coordinator
IRISA/INRIA
Christine.Morin@irisa.fr

Abstract

The overall objective of the XtreemOS project is the design, implementation, evaluation and distribution of an open source Grid operating system (named XtreemOS) with native support for virtual organizations (VO) and capable of running on a wide range of underlying platforms, from clusters to mobiles. The approach we propose is to investigate the design of a Grid OS, XtreemOS, based on the Linux existing general purpose OS. A set of system services, extending those found in the traditional Linux, will provide users with all the Grid capabilities associated with current Grid middleware, but fully integrated into the OS. The underlying Linux will be extended as needed to support VOs spanning across many machines and to provide appropriate interfaces to the Grid OS services. Installed on each participating machine, the XtreemOS system will provide for the Grid what an operating system offers for a single computer: abstraction from the hardware, and secure resource sharing between different users. It would thus considerably ease the work of users belonging to VOs by giving them (as far as possible) the illusion of using a traditional computer, and releasing them from dealing with the complex resource management issues of a Grid environment. By integrating Grid capabilities into the kernel, XtreemOS will also provide a robust, secure and easy to manage infrastructure for system administrators.

1 Introduction

Grids [10] enable the sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices that are geographically distributed and owned by different organizations for solving large-scale computational and data intensive problems in science, engineering, and commerce [4]. Virtual organizations [11] are temporary or permanent alliances of enterprises or organizations that come together to share resources, skills, core competencies in order to better respond to business opportunities or large-scale application processing requirements, and whose cooperation is supported by computer networks.

In the last few years some interesting projects like Globus [1], Legion [2] and UNICORE [3] were developed. The *à la Globus* approaches are designed as a sum of services infrastructure, in which tools are developed independently in response to current needs of users. In particular, Globus started out with the bottom-up premise that a Grid must be constructed as a set of tools developed from user requirements, and consequently its versions (GT2, GT3, GT4) are based on the combination of working components into a composite Grid toolkit that fully exposes the Grid to the programmer. However, a risk associated with those approaches is that, as the number of services grows, the lack of a common programming interface to components and the lack of a unifying model of their interaction can have a negative impact on ease of use. Typically, these issues must be dealt with by the programmer, who is forced to spend valuable time on basic Grid functions (e.g. providing their own mechanisms for service interoperability), thus needlessly increasing development time and costs [12].

We believe that the Grid infrastructure must absolutely reduce the burden on the application developer investing on the open source operating systems and extending them towards Grid, simplifying the life of the high-level Grid services implementers because they could rely on the native services of the operating sys-

*This paper presents the work of the XtreemOS consortium.

tem kernel for tasks such as resource or process management.

In this paper, we present our approach in designing the XtreemOS Grid operating system. The key technological challenges in XtreemOS, compared to the classical middleware (à la Globus) approaches, are mainly related to the fact that XtreemOS aims to be a first step towards the creation of a true open source operating system for Grid platforms supporting distributed resources, such as PCs and mobile devices like PDAs or mobile phones. It is just as a traditional operating system does for a single computer providing an abstract interface to its underlying local physical resources. While much work has been done to build Grid middleware on top of existent operating systems, little has been done to extend the underlying operating systems for enabling and facilitating Grid computing, for example, by embedding some important basic services or functionalities directly into the operating system kernel.

In Section 2, we present an overview of XtreemOS design principles. We detail our approach for secure application execution in Section 3. The application execution service which targets efficient and reliable application execution is described in Section 4. Section 5 is devoted to XtreemFS, XtreemOS file system, which federates collections of data, providing efficient and transparent access to them. Section 6 concludes.

2 Overview of XtreemOS Design Principles

The challenge that XtreemOS project wants to tackle is to design and implement an Operating System (OS) that will support the management of very large and dynamic ensembles of resources, capabilities and information composing virtual organizations. We propose a Linux-based OS to support Virtual Organizations (VO) across multiple administrative domains. XtreemOS aims at making VO management easy for administrators and work within VOs easy for users. The cost of administering and operating a VO (e.g., adding or removing nodes, changing access policy, authenticating and authorizing users) should be minimized to a bounded value rather than simply increase with the number of users and resources participating in the VO. Moreover, the dynamicity of users and resource usage needs to be handled in a flexible way. Users and resources are often autonomous, not subject to the control of a centralized entity. A user or a resource can join or leave a VO at any time. Such cases could bring heavy management burdens to administrators, which must be alleviated by the newly designed mechanisms.

We first introduce the considered model of system. Then, we present XtreemOS requirements describing a set of scenarii of use. We finally present an overview of XtreemOS approach and architecture.

2.1 Model of System

A VO can be seen as a temporary or permanent coalition of geographically dispersed entities (individuals, groups, organizational units or entire organizations) that pool resources, capabilities and information to achieve common objectives. There usually will be legal or contractual arrangements between the entities. The resources can be physical equipment such as computing or other facilities, or other capabilities such as knowledge, information or data.

2.1.1 Grid Physical Infrastructure

In a VO, information is stored and services and applications are executed by a set of computers in a grid. A grid is assumed to be made of an uncountable number of computers that are called grid nodes (or simply nodes). Computers can be clusters, single PCs or mobile devices (MD) such as personal digital assistants (PDA) or mobile phones. Computers are interconnected through heterogeneous networks in the range of System Area Networks (SAN), LAN, metropolitan networks, and WAN, including wireless networks. Networks interconnecting grid computers do not exhibit a uniform quality of service. Mobile devices can only be considered as access points to the VO resources as they are only intermittently connected and have a limited number of resources. The more powerful computers may act as both an access point to VO resources and a resource provider for a given VO. Computers will generally be located in different administrative domains (belonging to different institutions or not).

2.1.2 Applications and Services

We consider the execution of distributed applications and services within virtual organizations. A distributed application is made up of one or several tasks. A task may be a process or a set of communicating threads or processes. A task is executed on a single grid node. Different tasks can be executed on different grid nodes. Various communication paradigms such as message passing or data sharing may be used for communication between tasks and within a task.

2.1.3 Virtual Organizations

Virtual Organization (VO) membership agreements are created as a means of isolating a selected subset of members from the overall Grid. VO wide agreements are still dominated by the Grid-wide agreement and cannot override these terms and conditions. Therefore, VOs formed within a particular Grid, using its agreed namespace, services and infrastructure must be compliant with its regulations. It should not be possible for VOs to be formed with participants that have not passed the basic qualification for entry in the Grid infrastructure.

We make the following assumptions:

- There are one or more organizations that are responsible for membership in a Grid infrastructure they sanction the initial membership of entities in the Grid and provide offline, online or inline validation of their membership. These authorities are trusted. Certificate authorities (CA) and real-world entities (persons, organizations) are not part of XtreemOS but are involved in the registration of Grid users.
- Any VO also has an authority or manager that provides a server for registration and membership in a VO.
- Organizations or individuals may be members of multiple VOs, given that they are registered members with the overall Grid infrastructure.
- Contracts, agreements and reputations do not need to be understood explicitly by the OS, but it is assumed that there are services that provide proof and assertions of the status of these to OS-level services.

Key components of a VO are an administrator of the VO, who is authorized to manage VO membership and policies, a set of participating users (called Grid users) in different participating domains, a set of participating resources in different participating domains, a set of roles which users/resources can play in the VO, a set of rules/policies on resource availability and access control, an (renewable) expiry time of the VO.

A VO and its implementation by an operating system can reside in several stages of VO lifecycle: VO identification, VO formation, VO operation, VO evolution, and VO dissolution. In each stage a set of security threats to the overall system exists.

2.2 Scenarii of Use

The following scenarii are intended to give a flavour of the way XtreemOS will operate, and enable the novel aspects of XtreemoS to be related to everyday experience.

2.2.1 VO and Security

Usability In contrast to a toolkit approach such as Globus, where there are two separate but highly interdependent complex entities to be managed in tandem (the underlying OS and the middleware), the XtreemOS project adopts an approach in which the standard operating system running on a machine is a Grid OS, that is to say, the operating system is fully Grid-enabled. Once the XtreemOS system has been installed on a machine, this machine is ready to participate in a VO with no need to install additional system software. Modifications to Linux to natively support VOs are done with a careful design to keep backward compatibility while providing build-in VO management interfaces that are as secure and simple to use as possible. System services and utilities such as login and shell programs, together with libraries, are extended in a modular approach so as to favor VO-level resource sharing requirements while keeping maximal transparency to users.

Logging A person who wants, using XtreemOS, to access resources and services provided by a single node or a set of nodes which may be affiliated with a VO first need to register as a Grid user and then to register to the given VO. He/she acquires credentials (a certificate) that will be used to prove her/his identity on login. The user must have a local account on some computer where the certificate will be stored but does not need to have a local account on the nodes he wishes to use.

Running applications and accessing resources To run applications or access services, a Grid user should log into a VO he/she belongs to. If the loggin is successful, the user will be provided with a simple shell that will enable him to issue and respond to requests (similar to the Linux shell Linux commands are actually requests to the OS to do something and the OS can and does refuse some of these commands). Requests can only be issued in the context of a VO, and can only be targeted at other members of the VO. Therefore, in order to access any resources anywhere, the user must join an appropriate VO.

Resources are owned by an owner and made available via VOs. The resource owner may make the resource available to other VOs. However, while the VO may have access to the resource, the resource owner remains in ultimate control of their resource.

Authorization is managed by the VOs. Basically, membership of a VO provides authorization to use the resources accessible to the VO, subject to any finer grain access control policy the VO may implement. At its simplest, this operates like the Linux group concept. When a user tries to access a resource, the resource firstly checks the requestor is a co-member of one of its VOs, and then uses that VOs policies to decide if the access is permitted. The resource is also expected to check with its own base VOs policy before giving access to anything on the local host.

Security Policies Access security in XtreemOS will be policy driven. This means that for each resource (which includes VOs, applications, hosts, etc., in fact anything that requires protection) there will be a policy specifying who can access it and what they can do with it. In the case of a resource such as a file, the who could be a list of individuals and/or VOs, and the what could be read, write or execute actions similar to the conventional Linux file permissions (with a VO being considered as a sort of group). However, in a distributed and VO-based environment access will typically involve more than one entity, each with its own policies.

2.2.2 Application Management

Integrated control for execution XtreemOS will allow users to have a much simple and, at the same time, powerful environment to control the execution of their applications. As all layers will be integrated, the system will be able to offer information about the progress of the job, accurate monitoring of the used resources, error information, etc.

In the current Grid world, given that the managers for the different layers are not integrated, a lot of information is lost in the way and the one that survives it is not correlated making it very difficult to use. For instance, in current Grid systems it is difficult to know why an application failed, when and with exactly what resources it run, etc.

The integration of all services in a single OS will remove the lack of integration and offer users an execution environment with plenty of monitoring information and a powerful control of execution.

Reducing scheduling level In current Grid systems, there are many scheduling levels and these levels are not coordinated, which means that decisions made at one level may be contradictory to decisions made at a different level. This lack of coordination has two negative side effects. On the first hand, plenty of time and resources are lost in scheduling tasks that could be avoided in an integrated system. On the other hand, the final schedule is not as good as it could be because the decisions made by one level are not necessarily taken into account in another level.

The integration of resource management will avoid these multiple scheduling levels leaving only the necessary ones that will be coordinated.

Accurate accounting Nowadays, accounting is done by the grid middleware, but the Grid middleware is not really aware of the resources used, how shared they were, etc.

In XtreemOS, as the management of resources will be all integrated, this accounting will be much more accurate and could be used in a future for usage billing/compensation without the problems we have today.

2.2.3 Data Management

XtreemFS, XtreemOS Grid file system, should exhibit a Unix-like (Posix) behaviour where possible. It should support extended meta-data, hierarchical names (the traditional directory structure), private, shared and collaboration data, and data archives. It should also support named Grid pipes, used by workflows where different processes produce data and some others consume it, the various processes being located on different nodes. Access rights should be managed in a manner such that file access could be granted to Grid users according to VO policies.

2.3 Overview of XtreemOS

Current general-purpose operating systems have not been designed to support VOs. Each OS considers itself to be an island, with other systems being considered minor subsidiary players to be handled individually and specifically. VO management is usually implemented in Grid middleware.

XtreemOS is designed as an operating system and facilitates the use and management of VO resources by making transparent the resource distribution and heterogeneity as well as reconfigurations due to the dynamic nature of VO (at anytime, a computer may join or leave a VO voluntarily or consequent to a failure). Installed on each participating machine, the XtreemOS system should provide for the grid what an operating system offers for a single computer: an abstraction of the hardware and secure resource sharing between different users. It would thus considerably ease the work of users belonging to VOs by giving them (as far as

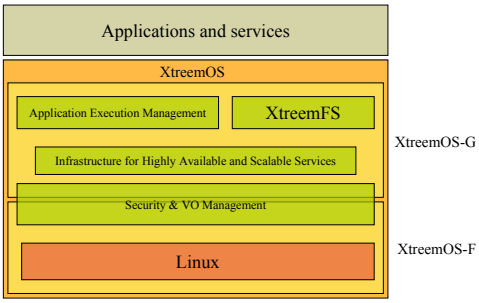

Figure 1. XtreemOS architecture

possible) the illusion of using a traditional computer, thereby releasing them from dealing with the complex resource management issues of a grid environment.

The XtreemOS operating system is based on Linux, which is extended rather than replaced (see Figure 1).

Internally, XtreemOS is composed of two parts: XtreemOS foundation called XtreemOS-F and XtreemOS high-level OS services called XtreemOS-G. XtreemOS-F is the modified Linux kernel embedding VO support mechanisms and providing an appropriate interface to implement XtreemOS-G services. XtreemOS-G is implemented on top of XtreemOS-F at user level. XtreemOS-G comprises of several Grid OS distributed services to deal with resource and application management in VOs. The VO and security management service is implemented in two tightly coupled parts: the XtreemOS-G part relies on mechanisms implemented in XtreemOS-F as explained in Section 3. XtreemFS, the XtreemOS Grid file system, described in Section 5 and the application execution management service, presented in Section 4 both rely on the infrastructure for highly available services [5]. This infrastructure provides a set of building blocks (such as for instance publish/subscribe, distributed server, virtual node, directory service) that allow the upper services to cope with the large scale and dynamicity of VO.

There are three flavours of XtreemOS-F system, one for each kind of Grid node: PC, cluster and mobile device. The cluster flavour of XtreemOS-F relies on the Kerrighed [14] Linux-based single sytem image. A cluster thus appears as a single powerful higly reliable node in the grid, its individual nodes being invisible at Grid level.

3 Secure Application Execution

The goal of VO support in XtreemOS is to provide mechanisms to set up and manage VOs in a scalable and flexible manner, and mechanisms which ensure access to various resources with fine-grained, mandatory access control without sacrificing site autonomy. VO support functionalities in XtreemOS are realized by the cooperative activities of VO-level and node-level management services [7, 8]. Particularily, the main objectives of node level VO support are:

- to facilitate the administration tasks for a single PC node to join or leave VOs,
- to enforce VO policies locally with system-level isolation of multiplexed VO accesses to the same node,
- to significantly increase the usability of shared resources on local nodes for VO users.

The key challenge here is to reach a harmonisation between VO-level policies and local policies on nodes which depend on autonomic domain administrators. On one hand, the enforcement of multiple VO security policies should be differentiated, on the other hand, this kind of enforcement should not be conflicting with any local policy of nodes and it will not impaire the usability of resources for grid users.

Node level VO management encompasses two main tasks:

- User Identity Transferring
  The task is to develop an account-based mechanism for the smooth transferring from global grid identities to local Linux accounts.
- Resource Management and Access Control
  The task is to develop resource management mechanisms compatible with local Linux Discretionary Access Control (DAC) model while supporting required access control specified in VO policies, which, to the contrary, generally adopt the Mandatory Access Control (MAC) model.

3.1 Session Management

A grid session on a Linux-XOS node covers all activities on beneath the same credentials. A grid session starts with the acceptance of the user crendential. When the session is terminated, no more activity can exist on behalf this credential. Opening a new session on a grid node takes the three classical steps: authentication, authorisation and session creation. These three

steps can be implemented in XtreemOS through PAM plugins as explained in Section 3.2. Linux-PAM is a system of libraries that handle the authentication tasks of applications (services) on the system. The library provides a stable general interface API that privilege granting programs (such as login and su) defer to perform standard authentication tasks.

Authentication the credentials presented with the request (ticket, proxy certificate, ...) are checked and validated: user identity from some certificate authority, attribute certificates from VOs. Depending on the VO model, this phase can necessitate transactions with external grid services.

Authorization after the request has been fully authenticated, further authorization checks can be taken, for instance that access to this node is not denied for this user.

Session creation the last step after authentication and authorization is the creation of a user context. This step involves the selection of a local UID/GID for the session, the initialisation of the user environment (from the job description document), creating a scratch homedir or mapping the user homedir from a grid file system, starting some auditing/logging/monitoring/management service, storing the proxy certificate in the session context (keyring), dropping some capabilities and running the user request.

A session on a grid node can be limited to running a single process (simple session). It is also possible to interact with the session through the associated management service, for instance to start new processes (composite session). XtreemOS API must provide means to run simple application –a single request for the whole execution– and to run complex applications –a session is first opened and then multiple execution requests can be handled in this session–. Session can be interactive or not. To facilitate interaction with applications using GUI interfaces, XtreemOS sessions must provide secure X11 forwarding as in SSH.

The session is terminated when the application of a simple session is terminated, the management service received an end-of-session request, or the proxy certificate is no more valid.

3.1.1 Local UID Management

In order to run processes on a grid node, a local UID/GID must be allocated for each session. This UID/GID can be static (the user owns a local account on the node) or dynamic (the user is unknown). A dynamic UID/GID is returned to the free UID/GID pool at the end of the session once all corresponding processes have terminated and once all local objects (files, etc.) have been deleted.

The possession of a local account can be managed in a kind of gridmapfile (a configuration file listing user Distinguished Names and the local accounts) and analysed by a session PAM plugin. It should also be possible to specify the local account identification using a trusted (by the local domain) attribute of the user proxy certificate.

3.1.2 Access control

Access control to an object in XtreemOS depends on whether it is a kernel object or a grid object. Kernel objects are local files, local processes, local shared segments, ... on a grid node. Applications access these objects using native Linux API: open/read/write for files, signal for processes, ... In order to avoid deep modifications to the Linux kernel, access control to kernel objects should use the native DAC/ACL system of Linux in XtreemOS.

Access control to grid objects will be managed inside grid services (grid file system, etc.). It is possible to "cache" grid objects inside the kernel in order to improve efficiency. This is the case of local grid file replicas. In the case where many concurrent sessions open the same grid file, they use the same local replica. Linux ACLs appears to be the most appropriate tool for handling this case. Access rights to replicas are removed by the grid file system service when the file is closed.

3.2 Modification to Linux

Virtual Organizations (VO) support in Linux requires the control, mapping, allocation, monitoring and enforcement of global, grid and VO visible resources onto single Linux nodes. Linux is unaware of the global grid entities therefore mechanisms for recognizing, controlling and enforcing their usage on the Linux machines should be added. We chose to use existing infrastructure in the Linux kernel:

- Global grid entities are mapped to local ones (like user and group IDs (UID/GID)). Existing mechanisms for local entities will be used to enforce resource control.
- The kernel key retention service will be used for storing user private keys, certificates and proxies associated with user processes and sessions in kernel space. This mechanism will avoid the need for

kernel changes for transporting grid-related information in process context.

Modifications to Linux mainly reside at system service level rather than kernel level. The chosen approach is minimal with respect to core kernel code changes and tries to keep required kernel changes localized in dynamically loadable kernel modules. The interaction between kernel space and user space daemons is done through existing APIs.

The following extensions are added to facilitate the use of VO functionalities by users:

- A new PAM module will be developed to take over the initialization of grid sessions. After authenticating users and getting authorization information from higher-level VO services, this PAM module will do account mapping and translate VO-level policies into POSIX-compliant local policies, i.e. UID/GID, ACL and POSIX Capabilities. This PAM module itself is designed as a pluggable framework to fit with various high-level VO models. For example, authentication plugins are capable of authenticating users with respect to their PKI certificates, or from a MyProxy repository, or by interacting with federated identity providers like Shibboleth. To obtain attributes information for users, authorization plugins could be developed to access VOMS servers or role-based access control frameworks like Permis. The plugin architecture of this PAM module makes node-level management mechanisms independent from higher-level VO frameworks. In XtreemOS, a domain system administrator installs the plugins corresponding to the VOs having access to the local nodes. The system is extensible: new plugins corresponding to new VO models (or credentials) can be developed and installed dynamically. Virtual organizations managed using different VO models can cohabit on the same node.

- Besides the PAM module guarding the login process, there will be runtime services that monitor and control users' activities during their grid sessions. The responsibilities of these services include checking and adjusting resource limits of processes in a VO context (e.g. by set_rlimit()), logging resource usage of processes, and providing error or debug information feedback to users.

- System services and utilities such as SSH could be extended to allow grid users to use remote nodes interactively without the need of explicitly creating traditional user accounts. Once login, grid users with identities like certificates could have a shell access to remote nodes in a VO. It may favor the requirements from advanced users who wish to develop and debug applications on grid nodes rather than being limited to submit batch jobs. This modification allows for a mimic approach of using the Grid as same as using traditional high performance computers.

Overall, by extending the Linux operating system with builtin VOs support, XtreemOS can provide outstanding performance and enhanced security, while minimizing administration costs of VOs compared with existing middleware solutions for VO.

4 Efficient and Reliable Application Execution

The Application Execution Management (AEM) system is in charge of efficient and reliable job execution [6]. A *job* is one or more Linux processes that collaborate to achieve a certain goal or objective. The idea is that a job is a resource allocation unit and resources are consumed by the processes (or their threads). A *resource* is any physical or logical component of limited availability within a computer system. These resources, besides static characteristics will have some dynamic ones that will also be used in the allocation process. Finally, we will have to take into account that resources can be allocated only partly (some processors) and that they may belong to different VO.

The AEM system comprises a set of services that covers necessities of users and jobs in executing new jobs, controlling their execution, monitoring their execution from different perspectives (resource consumption, performance, status, etc). The AEM system of XtreemOS is not targeted to specific users or types of jobs, so it must be as generic and flexible as possible, differentiating the services or functionality from details such as job specification. We take into consideration what users are expecting from a grid system and what a grid system has to offer in terms of security, scalability, efficiency, fault tolerance, and management of dynamicity and heterogeneity.

User Point of View From the point of view of users (or the job itself), what is expected from the AEM services is a set of facilities that allows to efficiently exploit the advantages of executing jobs in a grid. That is, the huge amount of resources should be available in an easy and efficient way. By easy we mean that XtreemOS must support remote execution of jobs submitted in a standard UNIX way without modifications in the binary code, and with minimum user intervention when

submitting it. Obviously, this approach reduces the amount of information provided to the AEM system to the default context values. However, our goal is to combine these default values with additional user-provided hints about resources and scheduling goals in order to best support the job . We think that hints such as my job consumes a lot of cpu, or memory are something that we can expect from users. By easy we also understand that we must provide tools to users (and jobs) to monitor job execution. These tools must be as easy to use as executing a UNIX *ps* command.

Job Point of View From the point of view of a job, the AEM system should offer an efficient job execution management, that selects the most appropriate set of resources, and automatically migrates them in case of resource failures. The characteristics of the Grid (dynamicity and heterogeneity) makes it desirable for the AEM to hide (as maximum as possible) these issues to users.

System Point of View From the point of view of the system, the AEM has to guarantee the access to authorized resources and their limited utilization. Jobs are executed in the context of a Grid user and a VO. The AEM has to ensure the utilization of allocated resources and should offer the required services by the jobs. In terms of scheduling, the system has different goals from the users. Once the job is submitted to the local system, local scheduling policies will be applied in order to maintain a certain independence from the whole system. Local systems expose their resources but they want to maintain the control.

4.1 Services for Application Execution Management

We have defined a set of services that allow us to offer all functionality requested by users and allow a good level of control (which is not normally the case in Grid systems). In addition, we have proposed these services taking scalability into account and thus most of these services are not long-term running and do not have a global view of the system. The proposed services can be summarized as follows:

jScheduler that decides the best resource selection for the job. It does not have a global view of the system and is only active from the submission to the queuing of the job in a resource (or set of resources). There is no such thing as a global grid scheduler (it would not scale).

LTScheduler that controls local resources and manages lists of local jobs. This service has a local-system view and runs permanently. If such a service already exists in the resource, our version will only implement a layer interface to negotiate scheduling agreements.

jController that controls scheduling agreements during the whole life of the job and acts like a gateway for the job. It has a local view (the job) and lives for as long as the job exists. It could take the decision to start a job migration.

jExecMng that manages the efficient utilization of the allocated resources. For instance, this process decides the correct placement of processes to resources to improve performance. It reacts to some jController decisions. It lives during the job execution.

jMonitor that monitors the status, performance and resource consumption of the job. It lives during the job execution.

jEvent that manages events (i.e. Linux signals) to the job. It lives during the job execution.

JobDirectory that manages a list of jobIDs and the contact information (jController). This service will be needed to implement ps-like functionality. This permanent service will be the only one that has a global view of the jobs in a VO.

rMonitor that controls resource dynamic information. It runs permanently.

jResourceMatching that performs the matching between a list of resources and the job requirements, taking into account the dynamic information of the resources. It lives during the scheduling cycle.

rAllocation that takes care of allocating resources to a job. It receives requests to allocate resources to a job and runs permanently.

5 Efficient and Transparent File Access

XtreemFS [9], XtreemOS Grid file system, is designed to allow efficient and transparent access to files stored in different institutions. We first present the requirements guiding us for the design for XtreemFS. Then, we present an overview of XtreemFS architecture. Finally, we briefly sketch the functionning of XtreemFS.

5.1 Requirements

Fault Tolerance and Scalability As a system running on standard hardware connected with WAN links, the system must tolerate hardware outages. Moreover, it must handle amounts of data and requests that exceed the capacity of single machines. This is achieved by distributing the system over multiple machines. Adding more machines to the system should not involve a disproportional increase in overhead.

Federation XtreemFS will keep data of many institutions. While it allows a global view on all data available in the system, it must also guarantee the availability of data in presence of network disconnections or institutions leaving the federation of systems.

Institutions should be able to use their local system with their local data when they are disconnected or when parts of the overall system are disconnected.

POSIX Compliance To support traditional UNIX applications, we have to be POSIX compliant. Because strict POSIX compliance severely restricts scalability of the file system, we allow XtreemOS-aware applications to fine tune guarantees in order to improve their performance.

Name Spaces The main aspects of data management in XtreemOS are mainly user and group files and their sharing for collaboration purposes. Expecting to handle large numbers of files, it is recognised that the directory hierarchy and filenames of traditional file systems are not sufficient. In addition to hierarchical directory structures, the system hence requires support for extended meta-data. A semantic naming of files allows a database-like arrangement of the file system, including a retrieval of files by means of queries based on attributes.

Commodity Hardware To be attractive to a wide audience, the file system is supposed to run on commodity hardware. That means it does not assume the presence of unusual large amounts of main memory (>4 GB), RAID protected drives, etc.

5.2 Overview of XtreemFS Architecture

XtreemFS is a distributed file system structured according to the *object-based file system* approach [13]. Its core abstraction, the object, is the pure content of a file without its metadata. The system consists of the following components:

- Object Storage Device (OSD) stores objects and implements a read/write interface to them.
- Metadata and Replica Catalog (MRC). This component stores file metadata (extended and POSIX) and replica locations of files. It also makes authorisation decisions according to access policies.
- Replica Management Service (RMS). This component cooperates with other services to decide when and where replicas are created and when replicas should be removed from the system.
- The Access Layer consists of a client-side library and a POSIX compatible file system module. The library offers access to all XtreemFS features for XtreemOS aware applications. The file system module allows mounting of XtreemFS as part of the traditional UNIX file system layout.
- XtreemFS supports applications with an Object Sharing Service (OSS). It allows sharing of data residing in volatile memory with object granularity. In this context, objects are volatile memory regions containing dynamically allocated objects and/or memory-mapped file data.

The object-based file system architecture splits files into their pure content, the objects, which are stored on so called object storage devices (OSDs), and the file metadata, which is put on dedicated metadata servers. XtreemFS extends the architectural concept of object-based storage to Grid environments by replacing the centralised metadata servers with a federation of metadata servers in order to ensure independence of participating organisations while maintaining a global view of the system.

In order to achieve scalability and fault tolerance, XtreemFS also features replication and partitioning/striping for both file metadata and the file content. Dynamically created communication overlays coordinate concurrent accesses and ensure the data's consistency in a scalable way. Data can be replicated across organisation boundaries, and therefore special attention needs to be paid to the latencies that the connecting wide area networks introduce and to failure cases like those of possible network partitioning.

Files can not only be placed manually to different stores, but an automatic system will monitor file access and resource conditions to automatically optimise data layout in the Grid. In addition, semantic naming and advanced query functions allow users to find data in huge archives, with the aim of overcoming limitations for the organization of data of traditional hierarchical file systems.

5.3 XtreemFS in Operation

A look on the events triggered by reading from a file illustrates how the components of the Grid File Service work together. The read function call is mapped by the user-level system libraries onto a kernel system call. The kernel figures out that the corresponding file lives in a Grid file system, and forwards the request to its user-level driver, the first XtreemOS component involved. The user-level driver asks the replica service for a location handle of the file. In turn, the replica catalogue determines the best available file copy or creates a new replica if needed. Using the received location handle, the driver contacts the named file access service instance and requests the required data. Advanced functions come into play for determining the "best" replica and when successive reads are requested. Determining the "best" replica requires constant monitoring of all resources that are involved in the data transfer between the file host and the client, like file host load and network usage. Successive reads will be monitored by the file access service to find out distinct read patterns, which allow the optimization of the subsequent reads, for example with prefetching data.

6 Conclusion

We presented in this paper an overview of the XtreemOS Grid Operating system, which is under design and development in the framework of the XtreemOS European project. XtreemOS is designed to provide a native support to virtual organizations. With XtreemOS installed on their computer, users can take advantage of Grid resources of a virtual organization they belong to while having the illusion to execute their applications on their local computers. XtreemOS objective is to make as much as possible the Grid invisible allowing efficient, reliable and secure application execution in a simple way. XtreemOS extends the widespread Linux operating system with kernel modules and a set of user level services. Three flavours of XtreemOS will be implemented, for PC, clusters and mobile devices.

Acknowledgement

This work has been partially supported by the FP6 Integrated Project XtreemOS funded by the European Commission (Contract IST2006-0033576). This paper presents ideas developed as a result of the work of the XtreemOS consortium. The author thanks the participants in the XtreemOS consortium for the fruitful discussions about these ideas.

References

[1] Globus. http://www.globus.org.

[2] Legion. http://www.cs.virginia.edu/ legion.

[3] Unicore. http://unicore.sourceforge.net.

[4] M. Baker, R. Buyya, and D. Laforenza. Grids and grid technologies for wide-area distributed computing. *Software Parctice and Experience*, 2002.

[5] XtreemOS consortium. Design of an infrastructure for highly-available and scalable grid services. Deliverable D3.2.1, November 2006.

[6] XtreemOS consortium. Requirements and specification of xtreemos services for application execution management. Deliverable D3.3.1, November 2006.

[7] XtreemOS consortium. Security Requirements for a Grid-based OS. Deliverable D3.5.2, November 2006.

[8] XtreemOS consortium. Specification of federation resource management mechanisms. Deliverable D2.1.1, November 2006.

[9] XtreemOS consortium. The xtreemos file system - requirements and reference architecture. Deliverable D3.4.1, November 2006.

[10] I Foster and C. kesselman, editors. *The Grid: Blueprint for a new computing infrastructure*. Morgan Kaufmann, 1999.

[11] I. Foster, C. Kesselman, and S Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[12] A. Grimshaw, M. Humphrey, and A. Natrajan. A philosophical and technical comparison of legion and globus. *IBM Journal of Research and Development*, 48(2), March 2004.

[13] M. Mesnier, G. Ganger, and E. Riedel. Object-based storage. *IEEE Communications Magazine*, 8:84–90, 2003.

[14] C. Morin, R. Lottiaux, G. Vallée, P. Gallard, D. Margery, J.-Y. Berthou, and I. Scherson. Kerrighed and data parallelism: Cluster computing on single system image operating systems. In *Proc. of Cluster 2004*. IEEE, September 2004.