VIRTUAL ORGANIZATION MANAGEMENT IN XTREEMOS: AN OVERVIEW *

Erica Y. Yang[1], Brian Matthews[1], Amit Lakhani[1], Yvon Jégou[2], Christine Morin[2], Oscar David Sánchez[2], Carsten Franke[3], Philip Robinson[3], Adolf Hohl[4], Bernd Scheuermann[4], Daniel Vladusic[5], Haiyan Yu[6], An Qin[6], Rubao Lee[6], Erich Focht[7], Massimo Coppola[8]

[1]*RAL, STFC, U.K.,* [2]*IRISA/INRIA, France,* [3]*SAP Research, CEC Belfast, U.K.,* [4]*SAP Research, CEC Karlsruhe, Germany,* [5]*XLab d.o.o., Slovenia,* [6]*ICT/CAS, China,* [7]*NEC HPC Europe, Germany,* [8]*ISTI/CNR, Pisa, Italy*

y.yang@rl.ac.uk, b.m.matthews@rl.ac.uk, a.lakhani@rl.ac.uk, yvon.jegou@irisa.fr, christine.morin@irisa.fr, oscar.sanchez@irisa.fr, carsten.franke@sap.com, philip.robinson@sap.com, adolf.hohl@sap.com, bernd.scheuermann@sap.com, daniel.vladusic@xlab.si, yuhaiyan@ict.ac.cn, qinan@software.ict.ac.cn, lirubao@software.ict.ac.cn, efocht@hpce.nec.com, coppola@di.unipi.it

Abstract XtreemOS aims to build and promote a Linux based operating system to provide native Virtual Organization (VO) support in the next generation Grids. XtreemOS takes a different approach from many existing Grid middleware by: first, recognizing the fundamental role of VO in Grid computing and hence taking VO support into account from the very beginning of our design; and, second, getting around the overheads brought by layers of existing Grid middleware by enabling native VO support in the Linux operating system. This paper presents our vision of VOs in a Grid operating system and describes various aspects of VO management in our system architecture, ranging from lifecycle management, application execution management, security, to node-level enforcement mechanisms in operating system.

Keywords: XtreemOS, Virtual Organization (VO), VO Management, Grid

*We would like to thank the support from the European Commission under IST program #FP6-033576.

1. Introduction

XtreemOS is an European project with the objective to design, implement, evaluate and distribute an open source Grid OS, named XtreemOS, which supports Grid applications, and capable of running on a wide range of underlying platforms, from clusters to mobiles. The goal is to provide an abstract interface to its underlying local physical resources, as a traditional OS does for a single computer. While much work has been done to build Grid middleware on top of existent OS, little has been done to extend the underlying OS to support Grid computing, for example, by embedding important basic services directly into the OS kernel. The approach being investigated is to base XtreemOS on existing Linux. 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.

A key feature of XtreemOS is native support for Virtual Organizations. The term Virtual Organization (VO) is well-established within the context of economics. The typical goals of a VO include the temporal collaboration of several entities from different departments and their respective organizations towards achieving a common business objective. In Grid computing, synergies can be achieved by grouping users (provided by an identity infrastructure) which share OS- and application-specific resources and interfaces of computing nodes in a Grid [1].

The exact realisation of a VO differs from project to project. Some approaches concentrate on the legal or contractual arrangements between the participating entities. Other task-oriented approaches emphasize the workflow to achieve a goal. VOs can range from long-lived collaborations with many users (typically found in large-scale scientific applications) to short-lived, dynamic ventures to achieve one task between a small number of participants (typically commercial scenarios). A general purpose Grid OS should take a flexible approach to satisfy as wide a range of applications as possible; the use cases in XtreemOS reflect this diversity.

Thus, XtreemOS defines a minimal definition of the features of VOs and provides a toolbox which can be configured to the needs of the application. Key components of a VO are: VO administrators; a set of users and resources in different domains; a set of roles which users and resources can play in the VO; a set of policies on resource availability and access control; an expiry time of the VO. VO goals or workflows are not modeled, though XtreemOS tools allows these to be supported at application level. This will typically require enforcement of policies, event notification of the completion of processes, and monitoring of exceptional events, such as jobs still executing at VO expiration.

2. Requirements

The requirements for XtreemOS have been derived from a range of 14 applications [3]. In the following, we focus on requirements directly related to VO management and security services.

Three different roles are involved in managing VOs: *Domain administrators* maintain a pool of resources that are allowed to be integrated into a VO. They have the ultimate control over what resources will be available to a VO and regulate how a *VO user* uses its resources, and they ensure the reliability and security of resources being provided to VOs. *VO administrators* compose VOs from the resources provided by various domains, and they manage (e.g. create, delete, and modify) user accounts and the permissions VO users have within a VO. The role of the VO administrator can be assigned to one or more persons from the participating institutions.

Users can register with one or more VOs so that they can utilize resources from different VOs concurrently and independently. An individual may have different roles and be assigned with different capabilities in different VOs. To obtain a VO user account it is not necessary to have a pre-existing local user account in one of the domains belonging to the VO. Moreover, it must be possible to transfer data, files and directories between local and VO accounts (e.g. by copy or mount). Overlapping VOs are required, i.e. multiple VOs can be established on the same node. The applications require exchange of information between different VOs by means of messages (also instant messages), shared memory and data transfer.

It is required that VO management actions be highly automated, allowing them to be completed in a specified time threshold, typically to the order of a few seconds. It must be possible to guarantee the lifetime of a VO for a specified or an unspecified amount of time (until a notification e.g. by a user or application). VO management must be supported by an API, a command line interface and a GUI including VO monitoring facilities.

XtreemOS has to allow for dynamically changing the composition of VOs during application runtime, e.g., if certain computing resources fail. In such circumstances, the unavailable resources need to be automatically substituted by alternative resources also including a migration of the affected running application components.

Data stored on resources must only be accessible by users and administrators that are members of a VO with the appropriate access rights. Confidential data communicated must be encrypted. Loss of integrity of stored data must be preventable and detectable. Data should be hashed and digitally signed by a trusted key stored on the OS. The integrity of transferred data must be validated before being committed (need for an OS reference monitor mechanism). The OS must be capable of signing and verifying signatures of data in an end-to-

end manner. A transaction framework is necessary, considering the distributed nature of the resources. Users should authenticate via single sign-on to gain authorized access to VO resources. Administrators are capable of recording the usage (by whom and when) of resources without users being able to deny (repudiate) such usage. This includes a secure audit service with the ability to the record timestamps and the VO in which a certain resource was used.

Isolation of VO users: As users may be involved in multiple VOs, it is then necessary to separate their user data and have means of determining which VOs they are currently working in, when accessing data. Isolation of data per-VO: Data stored on the same resource for different VOs must show non-interference. Isolation of services per-VO: Parties in different VOs must not be able to recognize that they are sharing resources nor can they gain knowledge of what other parties are doing with those resources. If one of two virtualized services on the same physical resources fails, this must not interfere with the other. It is proposed to investigate in how far virtual machines or containers (e.g. provided by OpenVZ) can be used for the purpose of isolation.

3. VOs in XtreemOS Architecture

This section first sums up the VO management challenges addressed by XtreemOS and then, describes various aspects of VO management in our system architecture, ranging from lifecycle management, application execution management, security, to node-level enforcement mechanisms.

3.1 Challenges

XtreemOS aims to provide native support for the management of VOs in a secure and scalable way, without compromising on flexibility and performance. Several key challenges are identified from both the requirement analysis and investigation of most state-of-art Grid VO solutions.

Interoperability with diverse VO frameworks and security models. Different VO management frameworks and security models have been developed so far and new ones keep on emerging. The diversity of their implementations is embodied in their adoption of different user identities (e.g. X.509 end user certificates, Shibboleth handles), different message sequences (e.g. push, pull and agent models), different places to convey security attributes (e.g. proxy certificates or SAML tokens) and different policy models (e.g. role-based access control). XtreemOS must be able to interoperate with, rather than replace these existing solutions and even traditional local security mechanisms (e.g. Kerberos). It is a challenge that the operating system-level abstraction of VOs in XtreemOS allows for integration of various existing VO structures.

Flexibility of policy languages. XtreemOS puts emphasis on that both scientific and enterprise business applications are equivalently supported. Users from these two representative application domains have different views of policies in a VO, in terms of subjects (users), objects (resources), access rights, Service Level Agreement (SLA) and QoS constraints. Therefore VO policies in XtreemOS have to be expressive and flexible enough to accommodate various levels of resource access rules.

Scalability of management of dynamic VOs. In order to support large numbers of users in a dynamic environment (dynamicity of resources and of users) while still providing accurate isolation of these users, solutions such as rather static files containing user information must be avoided. For example, when VOs are dynamically changed, it is impractical for the VO manager to update gridmap files on all resources, due to the heavy admin burden caused as well as the difficulty to maintain data consistency.

Strong isolation, access control and auditing. Some applications request for strong isolation of user applications on the Grid: hiding user identities, protecting files and processes, strict division of performance load, and so on. These requirements are typical for most of the industrial applications. In some environments this ability to generate strong isolated execution environment could even be used to isolate individual processes on a single resource. Implementing such requirements is difficult without operating system support. Furthermore, a secure Grid system must provide strict access control from the service level down to the system object level (files, sockets, ...). In all cases, it must be possible to monitor and log operating system service usage as well as system object accesses. The audit log must contain references to user credentials (security ticket) and be securely provided to the resource owner as well as the VO manager.

3.2 VO Management (VOM)

In this paper, we use the concept, VO Management (VOM), to cover all the infrastructural services that are needed to manage the entities involved in a VO and ensure a consistent and coherent exploitation of the resources, capabilities, and information inside the VO under the governance of the VO policies. A VO policy is defined as an authorization statement that describes what activities a subject (e.g. an entity in a VO) is allowed to perform on an object (e.g. resources) with certain constrains (e.g. time, location), if there is any.

There are several stages of VO lifecycle: VO identification, VO formation, VO operation, VO evolution, and VO dissolution. VOM plays a different role in different stages of this lifecycle. During the identification stage, VOM is mainly responsible for user management (e.g. registration, attribute management) and

VO policy specification (e.g. constrains on resource usages). During the formation stage, VOM involves in the processes of resource matching, negotiation and establishment of Service-Level Agreements (SLAs) by applying VO policies. The operation stage leverages the information made available during the previous stages. In this stage, VOM coordinates logging, accounting, auditing operations on nodes and ensures the availability of such information, if needed. For jobs that require interactive sessions, VOM also provides authorized users with facilities (e.g. credentials) to access the sessions of runtime applications. The evolution stage takes place when the VO is altered during its lifespan, for example, by a change in the participating entities or in their conditions of use. During the last stage, VOM ensures the deletion of non-persistent information (e.g. temporary files and accounts) and the reclamation of credentials.

3.3 VOM and AEM

In XtreemOS, application execution is managed by Application Execution Management (AEM) services [2]. AEM services can be conceptually grouped into two types of services: Job Management Services (JMSs) and Resource Management Services (RMSs). JMSs cover all the job related tasks, such as job scheduling, monitoring, event handling, and execution management. JMSs are mostly operated on an individual job basis, that is, these services do not have a global view of the system. RMSs cover all the resources related tasks, including resource monitoring, selection, matching, negotiation, and allocation. This section focuses on the interactions between VOM and AEM during the job submission stage which is illustrated in Figure 1.

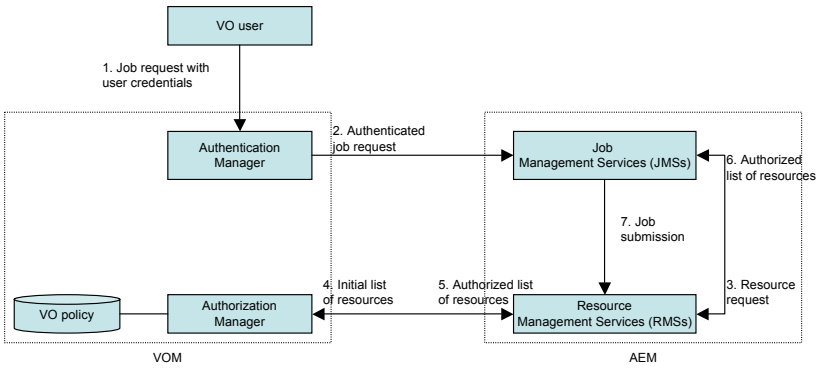

Figure 1. Interactions between VOM and AEM during the Job Submission Stage

VOM consists of two main components: authentication manager and authorization manager. The job submission process starts from a VO user submitting a job request to the authentication manager. The authentication is performed

against the credentials attached to the job request. Once a user is authenticated, the job request is forwarded to JMSs, which in turn contacts RMSs to select an initial list of resources. The selection is based on the job description and resource characteristics. The list is then subject to the scrutinization of the authorization manager to ensure that the job request and selected resources conforms to the overall VO policies in this context. As a result of the policy checking, an authorized list of the resources is then sent back to RMSs which will forward it to JMSs for conducting resource negotiation. Once the resources are successfully negotiated, JMSs will submit the job to RMSs which will then be responsible for launching the job on remote nodes.

In this process, the functionalities of VOM are to ensure that: a) only authenticated VO users can access AEM services; and b) the job execution will conform to VO policies by engaging the authorization manager in the process of resource selection.

3.4 Security

Because VOM involves in different stages of a VO lifespan, the design and implementation decisions on security services will have a significant impact on the efficiency and quality of the final XtreemOS operating system. This section discusses some of the key issues.

In Figure 1, when a user initiates a job request, the request is shown to be bundled with the user's credential for authentication purpose. In practice, this can be implemented in two different ways. The usual approach is to implement the authentication independent from underlying operating system level authentication. This is a popular approach adopted by many existing Grid middleware, e.g. Globus. It has the simplicity of implementation but comes with the complexity of configuration and management. That is, people who provide Grid services have to set up and manage two largely independent layers of services. In reality, it is common that more layers of services (e.g. Web services on top of Grid services) are introduced to establish extra levels of controls.

In XtreemOS we propose an alternative approach by aiming to integrate VOM as part of the OS. More specifically, VOM can be implemented as a service that can be integrated directly with existing authentication infrastructure. The benefits of this approach are described as follows. First, it reduces the management and performance overheads introduced by the layers of controls. Second, the hassle of accessing VO resources can be reduced. XtreemOS targets for both simple applications (i.e. a single request for the entire execution) as well as complex applications that involve interactivity and multiple execution requests, for example, for debugging purpose. Because of this reason, we anticipate that the integration of VOM as an OS level service should streamline the session management required for managing complex application.

3.5 Node-Level Enforcement

The policies specified by a VO, such as security, resource limitations, scheduling priorities and rules on how shared resources could be used by VO members, will be finally checked and ensured at resource nodes.

In order to adapt to different VO models and reduce kernel code changes, XtreemOS will use the PAM (Pluggable Authentication Modules) system [7], which allows a system administrator to add (possibly VO-specific) authentication methods by installing new PAM modules.

Local user accounts in XtreemOS are allocated dynamically on each resource to match the actual global users exploiting that resource. The XtreemOS PAM plugins would be in charge of implementing (or interfacing to) a local service allocating fresh local UID/GID couples upon request, of managing local UID names, of managing user home-directories (either from XtreemFS, the XtreemOS Grid file system [4], or on a scratch directory) and of managing the user credentials for XtreemFS access. The dynamic allocation of user accounts ensures XtreemOS scalability and reduces the complexity of VO management: no need to configure resources when users are added or removed from VOs.

During session initialization, XtreemOS stores the user security ticket in the kernel session keyring: this ticket will be associated to all local processes generated by the user request (fork, execve, etc.), and will be retrieved each time the global user identity or credentials need to be exploited: access to local or external service, auditing, ...

Dynamic management of local UID/GID also provides some level of isolation between Grid users: they do not share access to local files, and it is possible to hide the real identity of a user in the local name space. XtreemOS does not exclude the use of virtual machines, or process containers, to provide stronger isolation properties, like performance isolation (e.g. hiding CPU usage, memory limits).

The policy enforcement points provide access control and auditing on operating system services. Fine grain access control is also possible when the application activity generates requests to external services. This is the case for XtreemFS or NFSv4 filesystems. Fine grain access control and auditing on operating system objects (processes, sockets, ...), requiring the support of the Linux kernel, can be provided through the LSM (Linux Security Module framework).

4. Related Work

This section briefly discusses two exemplary tools that are relevant to XtreemOS in the area of VO management.

VOMS [5] is an important VO reference implementation to XtreemOS because it is currently a popular approach to integrate VO information (e.g. a user's roles in a VO) into node-level enforcement mechanisms.

However, managing VOMS effectively is an non-trivial task because authorization decisions are often a result of a joint process between the VOMS server (participating in the form of VOMS credentials) and nodes. Because both node policy and configuration will be taken into account, it is difficult to figure out what privileges a user has at a given time for a certain job. This is a practical issue which can be a potential hurdle for the future acceptance of XtreemOS.

In order to make node-level access control and account mapping decisions, nodes need to be knowledgeable of VO properties (e.g. roles and groups). However, managing such knowledge consistently and coherently can be non-trivial. This becomes a potential scalability problem for large VOs with a significant number of properties. It also makes it difficult to create new VOs and introduce new properties dynamically.

CAS takes control over the policy specification by explicitly spelling out the relationship between VO users and resources [6]. CAS represents a push model of enforcing VO policies because its policy enforcement is decoupled from VO information (e.g. user groups/roles). Therefore, it is comparatively easy to create dynamic VOs using the CAS model. However, it is not always easy to figure out what VO resources users need to use in advance.

Overall, VOMS and CAS represent two different ends of the spectrum. VOMS is lean to a pull model where access control is done at nodes by pulling policy information on demand (i.e. from the VOMS credentials) whilst CAS is a push model where authorization decisions are being pushed to nodes. Both are complementary to each other. Because XtreemOS aims to provide generic OS level support for Grids, we are investigating a combined use of both models in our system.

5. Conclusions

In this paper, we have described the outcome of the first stage of the XtreemOS project which has concentrated on developing requirements and initial architectural design of VO support, at both the kernel level and within the Grid support services of XtreemOS. This work is ongoing. The initial prototype of XtreemOS which will provide the basic instantiation of this architecture is under implementation. This will then be tested on a variety of use cases and further refined.

Further extensions to the basic VO support are planned. These would include: mechanisms for federated authentication; trialing of expressive policy languages; the role of virtualisation to support highly secure commercial VOs; the integration of trust domains. Further, we regard it an essential for a practical

system that there should be some assurance provided that the systems does meet recognised security criteria. Work is ongoing to derive a systematic analysis of threats to the XtreemOS system, with a view to validating the integrity of XtreemOS.

References

[1] Ian Foster, Carl Kesselman and Steven Tuecke. The Anatomy of the Grid Enabling Scalable Virtual Organizations. http://www.globus.org/alliance/publications/papers/anatomy.pdf

[2] XtreemOS consortium. Requirements and specification of XtreemOS services for Application Execution Management. Deliverable D3.3.1, November 2006. http://www.xtreemos.org/publications/public-deliverables/.

[3] XtreemOS consortium. Requirements Capture and Use Case Scenarios. Deliverable D4.2.1, January 2007. http://www.xtreemos.org/publications/public-deliverables/.

[4] XtreemOS consortium. The XtreemOS File System. Requirements and Reference Architecture. Deliverable D3.4.1, December 2006. http://www.xtreemos.org/publications/public-deliverables/.

[5] DataGrid VOMS (release v0.7.1). http://edg-wp2.web.cern.ch/edg-wp2/security/voms/

[6] CAS in Globus 4.2. http://www.globus.org/toolkit/docs/development/4.2-drafts/security/cas/

[7] Andrew G. Morgan and Thorsten Kukuk. The Linux-PAM guides. http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/.