Latifa Boursas Mark Carlson
Wolfgang Hommel Michelle Sibilla
Kes Wold (Eds.)

Communications in Computer and Information Science 18

Systems and Virtualization Management

Standards and New Technologies

Second International Workshop, SVM 2008
Munich, Germany, October 2008
Proceedings

Springer

Communications
in Computer and Information Science 18

Latifa Boursas Mark Carlson
Wolfgang Hommel Michelle Sibilla
Kes Wold (Eds.)

Systems and Virtualization Management

Standards and New Technologies

Second International Workshop, SVM 2008
Munich, Germany, October 21-22, 2008
Proceedings

Springer

Volume Editors

Latifa Boursas
Wolfgang Hommel
Leibniz-Rechenzentrum (LRZ)
Boltzmannstraße 1, 85748 Garching, Germany
E-mail: {latifa.boursas, wolfgang.hommel}@lrz-muenchen.de

Mark Carlson
Sun Microsystems, Inc.
500 Eldorado Boulevard, Broomfield, CO 80021, USA
E-mail: mark.carlson@sun.com

Michelle Sibilla
IRIT, Université Paul Sabatier
118 Route de Narbonne, 31062 Toulouse CEDEX 9, France
E-mail: sibilla@irit.fr

Kes Wold
Distributed Management Task Force, Inc.
1001 SW 5th Avenue, Suite 1100, Portland, OR 97204, USA
E-mail: kes@woldconsulting.com

Library of Congress Control Number: 2008937459

CR Subject Classification (1998): D.4, C.3, F.1, H.5, C.2

ISSN 1865-0929
ISBN-10 3-540-88707-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-88707-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12548743 06/3180 5 4 3 2 1 0

Preface

This volume contains the proceedings of the Second International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and New Technologies (SVM 2008) held in Munich, Germany, during October 21–22, 2008.

The SVM 2008 proceedings are intended for use by students of systems and virtualization management. The reader is presumed to have a basic knowledge of systems management technologies and standards at the level provided, for example, the Common Information Model (CIM) standard for modeling management resources. The student of systems management will find here material that could be included in an advanced study program. These proceedings should furthermore allow students to acquire an appreciation of the breadth and variety of systems and virtualization management research.

The proceedings also illuminate related standards and research issues, answering questions such as: what are the implications of virtualization for distributed systems management, which advances in information models and protocols aid in managing virtualization, what new problems will we incur when managing virtualized systems and services, and how might management itself benefit from virtualization? Topics related to managing distributed systems, virtualization of distributed resources/services and work in management standardization are also highlighted.

There were 15 regular paper submissions. These went through an active review process, with each submission reviewed by at least three members of the Program Committee. We also sought external reviews from experts in certain areas. All these inputs were used by the Program Committee in selecting a final program with 13 regular papers.

We gratefully acknowledge the support of the Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ), headed by Prof. Dr. Heinz-Gerd Hegering, through the provision of facilities and other resources, and our sponsor, the Distributed Management Task Force (DMTF).

Many individuals were very generous with their time and expertise. We thank the Program Committee and the external reviewers for their efforts in the assessment and evaluation needed to put together the technical program. We thank Dr. Wolfgang Hommel for his support in planning the logistics and his advice on organizing the conference.

We would especially like to thank the members of the MNM-Team for the local organization. We thank the system developers for creating and supporting

the invaluable EasyChair conference management system. Finally, we thank Springer for providing complimentary copies of the proceedings of SVM 2008.

October 2008

Latifa Boursas
Mark Carlson
Wolfgang Hommel
Michelle Sibilla
Kes Wold

Organization

The SVM 2008 workshop was organized by the DMTF Academic Alliance in cooperation with the Leibniz Super Comupting Centre and took place in Garching near Munich in Germany.

Organization Committee

Latifa Boursas	Munich University of Technology, Germany
Mark Carlson	Sun Microsystems, USA
Jeff Hilland	Hewlett-Packard Company, USA
Wolfgang Hommel	Leibniz Supercomputing Centre, Germany
Michelle Sibilla	Paul Sabatier University, France
Ellen Stokes	IBM, USA
Kes Wold	Wold Consulting, USA

Program Committee

Nazim Agoulmine	University of Evry Val d'Essone, France
Latifa Boursas	Munich University of Technology, Germany
Mark Carlson	Sun Microsystems, Inc., USA
Vitalian A. Danciu	Ludwig Maximilians University, Germany
Olivier Festor	INRIA Lorraine, France
Heinz-Gerd Hegering	Leibniz Supercomputing Centre, Germany
Wolfgang Hommel	Leibniz Supercomputing Centre, Germany
Jorge E. López de Vergara	Autonomous University of Madrid, Spain
Andreas Maier	IBM, Germany
Gregorio Martinez	University of Murcia (UMU), Spain
Jishnu Mukerji	Hewlett Packard, USA
Michelle Sibilla	Paul Sabatier University, France
Carlos Westphall	UFSC, Brazil

Referees

Javier Aracil
Walter Fuertes
Francisco Gomez Arribas

Sponsoring Institutions

The Distributed Management Task Force (DMTF).

Table of Contents

Systems and Virtualization Management: Standards and New Technologies

Virtual Execution Environments and the Negotiation of Service Level Agreements in Grid Systems*

Dominic Battré[1], Matthias Hovestadt[1], Axel Keller[2], Odej Kao[1], and Kerstin Voss[2]

[1] Technische Universität Berlin, Germany
{battre,maho,okao}@cs.tu-berlin.de
[2] Paderborn Center for Parallel Computing, Universität Paderborn, Germany
{kel,kerstinv}@uni-paderborn.de

Abstract. Service Level Agreements (SLAs) have focal importance if the commercial customer should be attracted to the Grid. An SLA-aware resource management system has already been realize, able to fulfill the SLA of jobs even in the case of resource failures. For this, it is able to migrate checkpointed jobs over the Grid. At this, virtual execution environments allow to increase the number of potential migration targets significantly. In this paper we outline the concept of such virtual execution environments and focus on the SLA negotiation aspects.

1 Introduction

At least in the academic domain, Grid systems evolved as a commonly accepted working instrument. Universities are providing their resources to Grid systems, enabling researchers to easily access them, e. g. for executing compute intensive simulations. In the commercial domain however, the Grid systems are not widely used yet, at best limited to small island solutions.

The commercial potential of Grid computing however has been recognized already in the early days of Grid computing. Firstly, companies like IBM, Hewlett Packard, and Microsoft investing noticeable efforts on research and the support of research communities. Secondly, already in 2003 the European Commission (EC) convened a group of experts to clarify the demands of future Grid systems and which properties and capabilities are missing in current existing Grid infrastructures. Their work resulted in the idea of the Next Generation Grid (NGG) [1,2,3]. This work clearly identified that guaranteed provision of reliability, transparency, and Quality of Service (QoS) is an important demand for successfully commercialize future Grid systems. In particular, commercial users will not use a Grid system for computing business critical jobs if it is operating on the best-effort approach only.

* This work has been partially supported by the EU within the 6th Framework Programme under contract IST-031772 "Advanced Risk Assessment and Management for Trustable Grids" (AssessGrid).

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 1–12, 2008.

Service level agreements (SLAs) are powerful instruments for describing the obligations and expectations of customer and provider within a commercial business relationship [4], e. g. specifying the QoS requirement profile of a job. Modern resource management systems (RMS) are working on the best-effort approach, not giving any guarantees on job completion to the user. Since these RMS are offering their resources to Grid systems, Grid middleware has only limited means in fulfilling all terms of negotiated SLAs.

The EC-funded project HPC4U [5] focused on closing this gap between the requirements of SLA-enabled Grid middleware and the capabilities of RMS, started working on an SLA-aware RMS, utilizing the mechanisms of process-, storage- and network-subsystems for realizing application-transparent fault tolerance. The project finished successfully in 2008, having an SLA-aware Grid fabric as its major outcome. The core of this Grid fabric is an SLA-aware RMS, able to fulfill the terms of negotiated SLAs also in the case of resource failures. For this, the RMS is acting as an active Grid resource, migrating jobs affected by outages transparently to other Grid resources, as long as these Grid providers agree on all terms of the SLA bound to the job to be migrated.

This system is using the Globus toolkit as Grid middleware, fully functional, and available as open source. However, the system currently requires a target resource which is very similar to the source resource, since the project uses kernel-level checkpointing mechanisms. Due to the heterogeneous nature of Grids, only a small subset of resources are eligible migration targets. Increasing the number of potential migration targets would directly increase the level of fault tolerance and reliability.

In this scope we introduced the concept of virtual execution environments. Instead of executing the compute job directly on the compute node of a cluster system, a virtualized environment is established which complies to the requirements of the job to be migrated. In this paper, we will first highlight the architecture of the HPC4U cluster system and outline the concept of virtual execution environments. In the main part we will describe how the requirements on these environments for the particular migration subject can be integrated into the SLA negotiation process. The paper ends with an overview about related work and a short conclusion.

2 SLA-Aware Resource Management

For the realization of an SLA-aware RMS, the project decided to use the planning-based RMS OpenCCS [6]. A new scheduler has been implemented, assigning jobs to resources according to their SLAs. Since the RMS has to guarantee the SLA-compliance also in case of resource outages, the HPC4U system does not only encompass the OpenCCS RMS, but also subsystems providing QoS and fault tolerance on storage, network, and process layer.

At this, checkpointing of running applications is the core functionality for providing fault tolerance. Without any checkpoints available, the results of a running application are lost in case of resource outages. In the light of long

running jobs, potentially running over weeks, using numerous computers of a cluster in parallel, the total number of lost compute hours can be immense. Therefore the RMS will regularly checkpoint the running application (e. g. by creating one checkpoint every 60 minutes). In case of a resource outage, the application can be restarted from the latest checkpointed state.

Since the checkpoint dataset must encompass not only the process memory and state, but also the network and storage, an orchestrated operation of all subsystems is mandatory to ensure consistency. In this process, the process subsystem first stops the running application, so that the state of the application remains fixed, not changing over the execution of the checkpoint operation. Now, the process subsystem creates a kernel-level process checkpoint of the running application. In case of MPI-parallel applications, this checkpoint consists of an image of all parallel running processes as well as the network state. After this, the application's storage partition is saved, so that a consistent image of the application environment has been generated. Finally, the process subsystem unblocks the application, so that its computation resumes.

This mechanism is fully implemented and operational. In the verification and validation tests of the HPC4U project it has proven to be applicable also to commercial applications, where the user may not recompile or relink the code. In fact, the entire checkpointing remains transparent for the user who does not have to modify his application nor the way of starting it.

This way of checkpointing an application might be problematic for applications communicating with cluster-external entities. Since the application remains stopped during the entire checkpointing process, it may not communicate with external processes. Depending on the duration of the checkpoint operation, this may result in timeout errors. However, the typical application targeted with this mechanism only communication within the cluster, e. g. in case of parallel applications.

2.1 Migration and Compatibility Profile

In contrast to application level checkpointing, a kernel-level checkpointed process can not be restarted on arbitrary target systems. Beside high level characteristics like operating system or processor type, the target machine has to be compatible with regard to versions of installed libraries and tools. When a checkpointed job is restarted on an incompatible resource, the job would directly crash at best. In the worst case, the application would resume its computation but return incorrect results.

An obvious way to face this situation and ensure a successful restart on the target machine is to request identical machines. At this, the hardware of a suitable target machine must be identical to the source machine. The same holds for the software installation. Both machines have to have identical operating systems with identical upgrade levels (e. g. RedHat AS4, upgrade 2). These requirements are fulfilled as long as the job is restarted within the same cluster system, since such clusters are usually homogeneously configured.

However, the HPC4U system may not only restart the job on the same cluster, but on any cluster within the same administrative domain or even on any resource within the Grid. In the case of restarting within the same administrative domain, compatibility can be configured statically (e. g. jobs of cluster A may be restarted on cluster B), because administrators know about the particular configuration. When migrating to Grid resources, a dynamic mechanism is mandatory.

The HPC4U project introduced the compatibility profile, holding compatibility requirements of the checkpoint dataset on the target resource. This ranges from high-level requirements like operating systems, processor architecture, or available main memory, up to low-level requirements like version information of loaded system libraries. When migrating over the Grid, this profile is part of the SLA negotiation process. Hence, if the remote system accepts the SLA request, it confirms to provide a resource compatible to the checkpoint dataset.

3 Virtual Execution Environments

The previous section underlined the difficulties of retrieving compatible resources for migration within Grid systems. Only if the target resource matches the compatibility profile, the checkpointed job may be migrated.

It is common practise that Grid resources are operated under a decentralized autonomy of local resource providers. Hence, local administrators decide on the operating system to be installed on compute resources. Moreover, is is the responsibility of local resource administrators to install updates, e. g. applying available patches to the local compute node environment. This local autonomy is the root cause of the difficulty of finding compatible resources. Since each difference in the operating system patchlevel impacts the compatibility for the migration process, it would be beneficial to have commonly accepted and available execution environments, guaranteeing the compatibility regarding paths and libraries.

This goal can be achieved by using virtualization technology on the compute resources. Instead of executing applications directly on a compute node, a virtual system is started first. Within this virtual system arbitrary operating systems can be started. The application then is started within this virtual system environment. This way, it is possible to start the application within a well-defined system environment.

The concept of virtual execution environments (VEEs) has been implemented in a first version, allowing to execute a job within a virtualized system environment on a compute node. The realization implies a number of additional requirements for the RMS, particularly for the daemons running on the compute nodes, where the virtual machine is to be established at runtime. To limit the impact on the overall system, a major design principle was to keep the existence of virtualized components as transparent as possible.

On the compute node we have OpenCCS daemons responsible for managing the nodes and executing jobs. The central component here is the Node Session

Manager (NSM), building a linking element between the daemons running on the front-end node and the compute node. It does not only configure and monitor the node, it also invokes the Execution Manager (EM) daemon, if an application needs to be started. This EM then runs with user privileges and executes the commands specified by the user.

For achieving VEE transparency, the tasks of the NSM have been enhanced to handle VEE-related jobs. This VEE-enabled NSM is called NSM+. The first task of the NSM+ in case of VEE-bound jobs is the establishment of the virtual machine on the compute node. For this it invokes commands specified in the OpenCCS configuration file. In the scope of this implementation, the Xen system has been used. Hence, the NSM+ running in Dom0 starts the image of the specified VEE as DomU.

Once the DomU has started, a second NSM+ is started within the DomU, responsible for performing all classic NSM tasks like node configuration or EM invocation. However, the NSM+ running in DomU does not communicate with other OpenCCS daemons directly, but uses the NSM+ running in Dom0 as communication proxy. Hence, the NSM+ running in Dom0 is called *proxy-mode NSM+*, whereas the NSM+ running in DomU is called *VM-mode NSM+*.

4 SLA Negotiation Aspects

The Grid community has identified the need for a standard for SLA description and negotiation. This led to the development of WS-Agreement/-Negotiation [7]. These upcoming standards rely on the Web Services Resource Framework (WSRF).

This WS-Agreement protocol now has been extended to allow flexible SLA negotiation schemes between contractors and service providers. Briefly, modifications consist in the addition of one operation: getQuote(). This is only an extension, which allows to change the original single-round acceptance model to a two-phase acceptance model. This introduce the negotiation possibility, in other words the bargaining capability.

The protocol implemented within the EC-funded AssessGrid project, used for realizing the migration using virtual execution environments, is a two-phase commit negotiation. The user first requests a template from the provider, which gives the provider capabilities. The user then describes his specific needs in a quote request. The quote sent by the provider gives the offer made by the provider, based on the request made by the user. The user is then able to accept this quote, and sign it. The SLA contract is then signed if the provider accepts the user's signed SLA.

4.1 Original WS-Agreement Protocol

The Web Services Agreement Specification (WS-Agreement, or WSAG for short) was produced by the GRAAP working group in the Open Grid Forum (OGF). The version on which is based the Assessgrid negotiation is the draft version of April 2007. The WS-Agreement protocol is based on a single round offer, accept

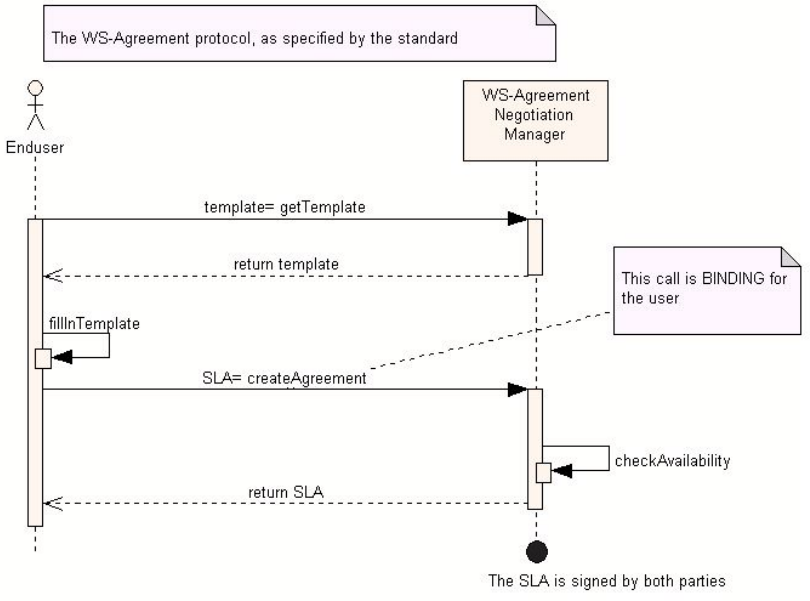


Fig. 1. Negotiation Sequence Diagram

message exchange between the negotiating parties. The basic synchronous wsag: CreateAgreement() operation directly captures this exchange, as can be seen in the following diagram:

The WS-Agreement specification supports additional asynchronous patterns, but the semantics of this abstract exchange is always the same: the initiator sends an obligating offer, with explicit terms of agreement, which the responder may accept or reject by unilateral decision. The agreement relationship is in place as soon as the responder decides to accept.

The agreement initiator can operate either on behalf of a service consumer or a service provider. As depicted in the sequence diagram in figure 1 , the initiator first requests the SLA template from the agreement responder querying a property of the AgreementFactory WS resource - calling getResourceProperty(). Then the initiator fills in the template taking into account the corresponding creation constraints. The result is the SLA Offer, that is sent by the initiator to the responder by calling createAgreement() operation. Now the responder is free to accept the offer (by returning an EPR to an Agreement resource) or to reject it by sending a fault response. The initiator is now bound to the agreement and he can inspect its properties. Considering the most usual case in which the initiator is the service consumer, a user could start using the service once the service state is ready.

4.2 Protocol Extension

The effort made by the AssessGrid team to develop a flexible and robust SLA negotiation protocol, can be seen as a continuation of work within the

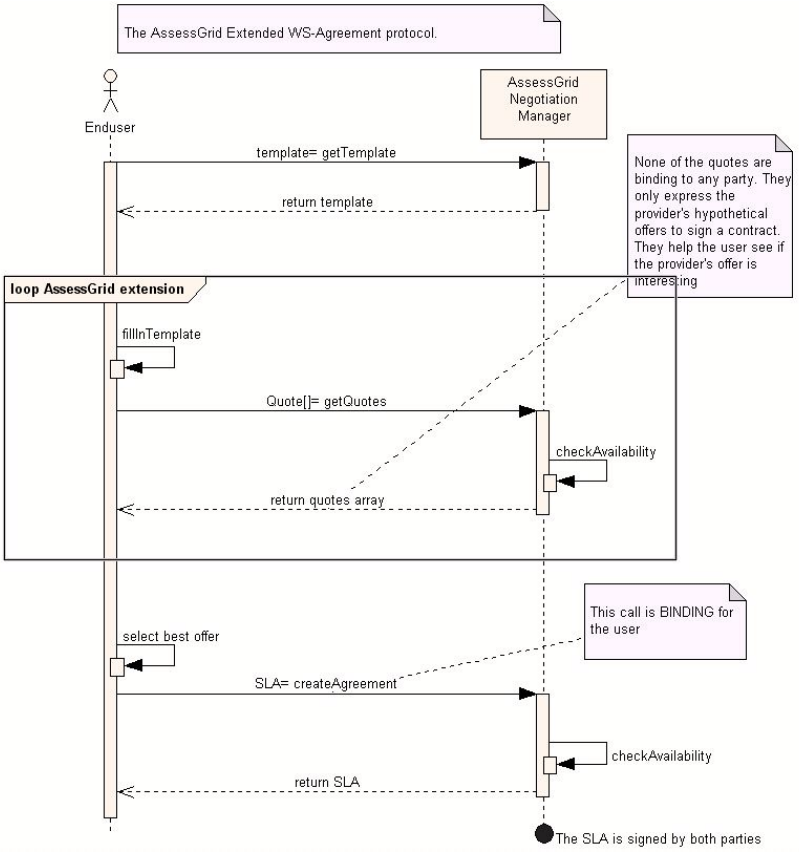


Fig. 2. Enhanced Negotiation Sequence Diagram

WS-Agreement specification. The extended protocol answers the requirements where a negotiation before a final agreement is needed.

The agreement mechanism within the WS-Agreement draft specification does not meet the negotiation requirement. The main drawback comes from the single round offer, accept agreement mechanism.This has an important consequence: there is no possibility for a service consumer to request offers from different providers so that he can choose the best one among them. In order to do so, he would have to become the agreement initiator and would have to call createAgreement() from several providers to propose some SLA to each. The problem is that he would then be bound to every provider that decides to accept. The concept of SLA quote does not exist in the WS-Agreement draft specification, it is not possible for a consumer to simply ask a provider what his terms would be without being committed to the provider by this action. In the real word, negotiation processes usually begin by the initiator asking non-committing questions to the other party.

The solution proposed has been to introduce the concept of SLA quote into the agreement mechanism. The concept of this enhancement is depicted in figure 2. The getQuotes() methods offers the end-user the possibilty to have a first evaluation of a request for service. Based on this first quote, the user can then decide to accept it using the createAgreement() method. If the provider's quote is not satisfactory, a new quote can be requested by entering a new quote request, with slightly different parameters.

4.3 Negotiation on Virtual Execution Environments

The central component of this new infrastructure is a central system image repository, holding system images of different operating systems, e. g. commonly used Linux distributions in different versions and different patchlevels (cf. figure 3). Each of these images has a unique ID. The contents of this image repository can be published within the Grid using well established mechanisms like resource information catalogues. Since these information services are public, not only a single provider is able to access the contents, but all Grid stakeholders.

Already at SLA negotiation time, the service customer now has the opportunity to specify a system image to be established at execution time, instead of the current practise of not knowing if the job will be executed on an up-to-date Debian system or an outdated SuSE. Hence, the customer can test his job in the specified environment by establishing the image from the system image repository locally, being sure that the job will succeed and return the expected results.

Providers will typically only support a subset of images from the image repository, e. g. images of distributions that are known by the system administrators, or that have proven to run stable on the compute resources. Providers are free in the selection of supported system images, which are then published in the Grid resource information catalogue.

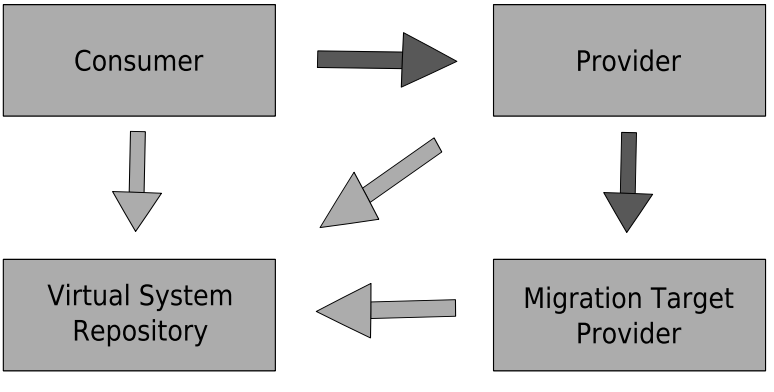


Fig. 3. Virtual System Repository

At level of Grid middleware the customer requested virtual system ID has to be matched against the provider supported characteristics of their resources, like it is already common practise with all other system parameters like number of nodes or amount of main memory. Hence, currently existing matching mechanisms, e. g. used by Grid broker systems, can be used for also matching the virtual system IDs.

If a provider agrees on an SLA specifying a virtual system image ID, it has to establish the specified system image at runtime on the compute node and then execute the user job within the specified environment. If the customer did not specify any image ID, the provider may choose a default virtual system to be executed on the compute node. In this case, the customer does not have any knowledge about the system that is used for executing his job, but this only corresponds to the current situation.

At runtime, the virtual system is then established on the compute node, so that all job checkpoints can be executed in a well defined and well known environment. Hence, all dependencies of the checkpointed job on the system environment are known, since the system has been executed in the specified virtual environment.

This is particularly beneficial for the migration process, since it is no longer mandatory to generate a compatibility profile, as explained in the previous section. Instead of querying for libraries or library versions, the resource provider is able to query for resources that can execute the particular virtual system ID. Hence, instead of describing requirements on a compatible environment, now the compatible environment can be requested directly.

In case of a resource failure, the provider first tries to identify job to be migrated. The goal is to release them from the current schedule, so that the remaining resources suffice the requirements of the remaining jobs, so that their SLAs can be fulfilled. Regarding the SLA-bound jobs to be migrated, the terms of their SLAs are subject of the SLA negotiation process between the migrating provider and the provider taking over the job.

Particularly this virtual system ID is part of the SLA negotiation process of the source and the target resource provider, which is providing the migration target resources. If the target resource provider agrees on the SLA, it agrees on establishing the specified virtual system on the compute node at runtime, where the checkpointed job is to be resumed. This mechanism also applies to all further migration operations of this job, e. g. if the new resource provider again has to query the Grid for backup resources due to resource failures. At the bottom line, the job is only migrated, if the target provider agrees on all terms, so that - thanks to the migration capabilities - all SLAs of all jobs are fulfilled.

In practise, resource failures in a cluster system typically affect more than only one single job, making the handling of multiple affected jobs necessary. However, it is not necessarily the best choice to migrate the job that has been directly affected by the resource outage (e. g. the time between job completion and SLA-specified deadline does not allow migration). Instead, the system should

pay attention to factors like revenue or remaining time between job completion and deadline.

The scheduler of the RMS takes these factors as heuristic for selecting an order to remove jobs from the system, which overlap with the jobs affected by the outage (including the job itself). As long as the revenue loss caused by migration is smaller than the penalty which has to be paid for violating the SLA of the failure affected job, the job qualifies for the migration candidate set, if the RMS is also able to get an SLA offer of some other provider.

If a set has been identified, which migration allows the fulfillment of the SLA of the affected job, the RMS tries to get SLA offers from other providers. If this is successful, the migration can be initiated. Otherwise the RMS has to remove the respective jobs from the set, continuing with the set generation. If no set can be found, the already agreed SLA offers are canceled.

5 Related Work

In [2], important requirements for the Next Generation Grid (NGG) were described, i. e. Grids to be successfully deployed in commercial environments. Mandatory prerequisites of such Grids are flexibility, transparency, reliability, and the application of SLAs to guarantee a negotiated QoS level. The Grid community has identified the need for a standard for SLA description and negotiation. This led to the development of WS-Agreement [7].

The usage of virtualization technology at provider level within RMSs has been described in [8]. Here the Xen virtual machine monitor is used for increasing system utilization and response time of a cluster system operated with the Sun Grid Engine RMS. Long running sequential jobs are executed in a virtualized environment and suspended for executing short running parallel jobs. This work underlined the general applicability of virtualization technology on compute nodes, but did not address fault tolerance nor the execution of parallel applications within virtual environments.

The benefits of using virtual machines for running applications in the Grid had been addressed from the perspective of virtual organizations in [9]. Their implementation uses VM-based workspaces configured with Xen hypervisors [10], which offer a client to create and manage other virtual machines. VM images are configured for specific applications and deployed as Edge Services. By using the VOMS credentials of a VO administrator, deployed VM images can be accessed and applications of the end-user can be executed without additional configuration effort. This work has a similar approach as our developments since it aims to increase the flexibility of the resource usage and also considers the negotiation of resource requirements. However, the capability of using fault-tolerance mechanisms for the provisioning of SLAs is left out of consideration.

The deployment of a configured VM on a hypervisor has no significant overhead since it can be performed in less than a second [11] which is comparable to the overhead of Grid tools. The deployment can be realized by using preconfigured VM images or refining configuration which influences the flexibility

and deployment time. In [12] an XML description is presented which should balance both aspects. Here, the flexibility is pointed out as an important issue if brokers configure the workspaces according to end-users requirements. Our system uses pre-configured VM images. However, generally no constraints exist concerning the approach for the description and deployment of the workspaces.

6 Conclusion

Introducing SLA mechanisms to all levels of the Grid is a prerequisite for attracting the commercial user to use Grid environments. For the level of Grid fabric, the project HPC4U realized an RMS able to negotiate on SLAs, ensuring their fulfillment also in the case of resource outages by migrating jobs even to resources over the Grid. Since kernel-level checkpointing is used for realizing application transparent fault tolerance mechanisms, the target migration resource has to be highly compatible to the source resource. Due to the heterogeneous nature of the Grid, this requirement significantly limits the number of available spare resources.

The concept of virtual execution environments allows to execute a job in an environment, which complies to all requirements of a job. As long as a provider is able to execute a given virtual environment on the compute nodes of his cluster system, the system qualified as potential migration target. This increases the number of spare resources, thus also increasing the overall level of fault tolerance and reliability.

The mechanisms presented in this paper preserve the local autonomy of resource administrators, who still can decide which virtual system should be supported. Moreover already available resource querying mechanisms within the Grid can be used for finding resource providers supporting a specific virtual execution environment. SLA negotiation mechanisms and resource scheduling presented in this paper ensure that the migrated job successfully resumes at the target resource, so that no SLA is violated.

The system described here has been implemented in the scope of the AssessGrid project. It uses the Globus toolkit and the OpenCCS resource management system. Currently, virtual execution environments are defined statically at provider level. In the next step, this static configuration will be superseded by establishing image repositories at Grid middleware level.

References

1. Priol, T., Snelling, D.: Next Generation Grids: European Grids Research 2005-2010 (2003), ftp://ftp.cordis.lu/pub/ist/docs/ngg_eg_final.pdf
2. Jeffery, K.(ed.): Next Generation Grids 2: Requirements and Options for European Grids Research 2005-2010 and Beyond (2004), ftp://ftp.cordis.lu/pub/ist/docs/ngg2_eg_final.pdf
3. De Roure, D.(ed.): Future for European Grids: GRIDs and Service Oriented Knowledge Utilities. Technical report, Expert Group Report for the European Commission, Brussel (2006)

4. Sahai, A., Graupner, S., Machiraju, V., van Moorsel, A.: Specifying and Monitoring Guarantees in Commercial Grids through SLA. Technical Report HPL-2002-324, Internet Systems and Storage Laboratory, HP Laboratories Palo Alto (2002)
5. Highly Predictable Cluster for Internet-Grids (HPC4U), EU-funded project IST-511531, http://www.hpc4u.org
6. Open Computing Center Software (OpenCCS), http://www.openccs.eu
7. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement) (2007), http://www.ogf.org/documents/GFD.107.pdf
8. Fallenbeck, N., Picht, H.J., Smith, M., Freisleben, B.: Xen and the Art of Cluster Scheduling. In: Second International Workshop on Virtualization Technology in Distributed Computing (VTDC 2006) (2006)
9. Freeman, T., Keahey, K., Foster, I.T., Rana, A., Sotomoayor, B., Wuerthwein, F.: Division of labor: Tools for growing and scaling grids. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 40–51. Springer, Heidelberg (2006)
10. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: SOSP 2003: Proceedings of the nineteenth ACM symposium on Operating systems principles, pp. 164–177. ACM, New York (2003)
11. Keahey, K., Foster, I., Freeman, T., Zhang, X., Galron, D.: Virtual workspaces in the grid. In: 11th International Euro-Par Conference (2005)
12. Keahey, K., Foster, I., Freeman, T., Zhang, X.: Virtual workspaces: Achieving quality of service and quality of life in the grid. Sci. Program. 13(4), 265–275 (2005)

An Extension of XACML to Improve the Performance of Decision Making Processes When Dealing with Stable Conditions

Romain Laborde and Thierry Desprats

IRIT UMR 5505, Université Paul Sabatier
118 route de Narbonne, 31062 Toulouse cedex 9, France
{laborde,desprats}@irit.fr

Abstract. XACML (eXtensible Access Control Markup Language) is an XML-based language for access control that has been standardized by OASIS. In this language, any entities involved in access control (i.e. users, resources, actions and environment) are specified by a set of attributes. This specification also includes the description of an architecture that explains how the policy decision point (PDP) retrieves the needed attributes values when it evaluates the policy to take its authorization decision. In this paper, we show that retrieving attributes values using a synchronous method as it is stated in XACML specification can be a bottleneck to the performance of the authorization decision making process. Especially, it is true when getting an attribute value is long and when the changing of this value doesn't impact the policy result frequently. Thus, we propose an improvement of the XACML architecture. It uses an asynchronous approach that accelerates the decision making process when PDP deals with expressions that include such attributes. Experimental results prove the performance is improved.

Keywords: XACML – Policy based Management – Access Control.

1 Introduction

Today, resources and services are complex and heterogeneous. Users are changeable and members of multiple organizations. To fit these requirements, access control models and access control solutions have followed the evolution of the environment to protect. Traditional Identity Based Access Control (IBAC) models [1] have been replaced by new models like Role-based Access Control Models (RBAC) [2] and more recently Attribute Based Access Control models (ABAC) [3]. Unlike IBAC and RBAC, the ABAC model can define permissions based on just about any security relevant characteristics of requestors, actions, resources, and environment, known as attributes. This approach makes ABAC scalable and flexible and thus more suitable for distributed, open systems that are identity-less like the Internet (where subjects are identified by their characteristics, such as those substantiated by certificates).

XACML (eXtensible Access Control Markup Language) is an XML-based language for access control that has been standardized by OASIS [4]. XACML policies

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 13–24, 2008.

are ABAC-Based. XACML specification also includes the description of an architecture that explains the attributes values are retrieved using a query/response approach during the decision making process. This synchronous method is a bottleneck to the performance of the authorization decision making process for attributes whose process for retrieving the value is long and the changing of its value doesn't impact the policy evaluation frequently. We name stable conditions the policy conditions whose results change rarely.

Thus, the XACML architecture should be improved in order to accelerate the decision making process when it evaluates these attributes. We build our approach on 1) removing the stable conditions from the evaluated policy and 2) adding the ability to adopt an event driven behaviour to XACML. The idea is to benefit from asynchronous notification mechanisms to inform the decision making process the value of a stable condition attribute has changed in such way the value of the associated stable condition has also changed.

This paper extends the concepts introduced in [14]. Especially it refines the definition of stable conditions and provides a profile to identify them. In addition, it proposes an experiment that proves this idea enhances the performance of the decision making process.

The rest of the paper is organized as follow. Section 2 introduces the XACML policy language and its architecture. Section 3 presents a scenario that points out the performance issue and provides a generic definition of stable conditions. Section 4 describes an enhancement of the XACML architecture to deal with stable conditions, and then it presents our implementations of both synchronous and asynchronous approaches for the scenario. Experiments prove the asynchronous approach is more efficient for stable conditions. Finally, section 6 describes the related works and section 7 concludes the paper.

2 XACML Standard Overview

XACML describes both an attribute-Based access control policy language and a request/response language. In addition to the request/response language, XACML provides a management architecture that describes the different entities and their roles related to the decision making process.

2.1 The XACML Policy Language

XACML policy language is used to describe general access control requirements in term of constraints on attributes. Specifically, attributes could be any characteristics of the subject, resource, action, or environment in which the access request is made. Considering attributes allows the language to be very flexible. Moreover, XACML language is extensible. It has standard extension points for defining new functions, data types, combining logic, etc.

At the root of all XACML policies is a policy or a policy set. A policy set is a container that can hold other policies or policy sets, as well as references to policies found in remote locations. A policy represents a single access control policy, expressed through a set of rules. Because a policy set or policy may contain multiple

policies or rules, each of which may evaluate to different access control decisions, XACML needs some way of reconciling the decisions each makes. This is done through a collection of combining algorithms. Each algorithm represents a different way of combining multiple decisions into a single decision. XACML specification provides six algorithms: Deny-overrides, Ordered-deny-overrides, Permit-overrides, Ordered-permit-overrides, First-applicable and Only-one-applicable. It is also possible to define new algorithms.

In order to make the search of the policy rule that applies to a given request more efficient, XACML has introduced the concept of target. A target is basically a set of simple requirements for the subject, resource and action that must be met for a policy set, policy or rule. Targets use simple boolean functions that are fast to evaluate. For example, a target can restrict the scope of a security policy to the service FTP only. The idea behind this concept is to find out quickly the policy rule to evaluate in a complex policy set or policy.

Once the request matches the targets of a policy set, one of its policies and rules together, the rule is evaluated. The core of a policy rule is its condition section which is a boolean expression like the targets. However, conditions may include more complex expressions on the attributes of subjects, resources, actions, and also environment.

Finally, the decision returned can be: (1) *not applicable* if no policy set or policy applies; (2) *indeterminate* if there is an error or a missing attribute; (3) *permit* or *deny* as specified in the applied rule. In addition to the authorization decision, the reply can contain an obligation section. We don't detail this feature that is out of the scope of this article.

2.2 The XACML Management Architecture

XACML specification proposes also a management architecture that is described by a data-flow model. A simplified version of this model is depicted in Fig. 2(a).

The model operates by the following steps: (1) Policy Administration Points (PAP) write policies and policy sets and make them available to the Policy Decision Point (PDP). These policies or policy sets represent the complete policy for a specified target; (2) The access requester sends a request for access to the Policy Enforcement Point (PEP); (3) The PEP sends the request for access to the context handler in its native request format, optionally including attributes of the subjects, resource, action and environment; (4) The context handler constructs a standard XACML request context and sends it to the PDP; (5) The PDP requests any additional subject, resource, action and environment attributes from the context handler; (6) The context handler requests the attributes from a Policy Information Point (PIP); (7) The PIP obtains the requested attributes; (8) The PIP returns the requested attributes to the context handler; (9) The context handler sends the requested attributes. The PDP evaluates the policy; (10) The PDP returns the standard XACML response context (including the authorization decision) to the context handler; (11) The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP that enforces the authorization decision.

The way the PDP gets the required attributes values when it evaluates the policy can constitute a bottleneck to the performance of the decision making process. We introduce an example to point out the issues.

3 Notion of Stable Condition

3.1 Example of a Scenario

In this example, we consider the management of a FTP server protected by an XACML-Based access control solution as illustrated in Fig. 1. The FTP server holds two directories: private and public. The private directory contains confidential files about the organization whereas files that are accessible to everybody are stored in the public directory.

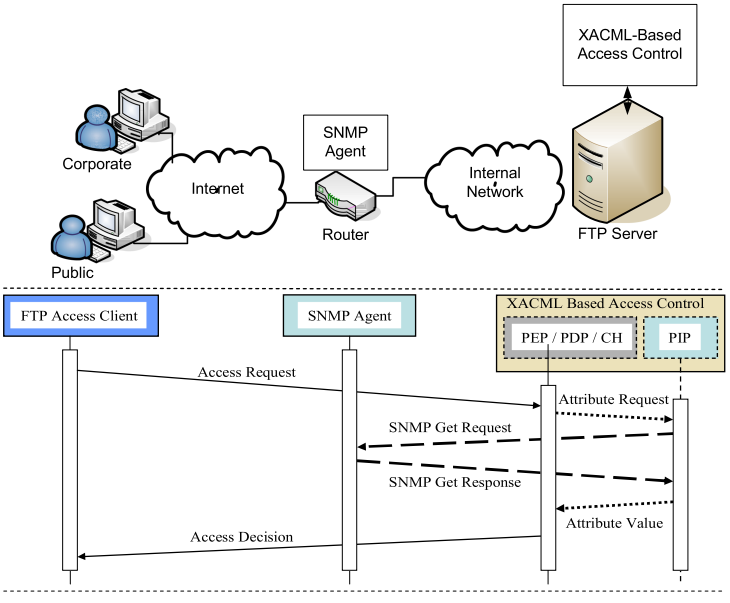


Fig. 1. Architecture of the example

The policy states that users with role "corporate" can access to any files in the private directory. Files in the public directory can be accessed by anybody unless the network is too busy, which means the bandwidth use rate on the edge router is greater than 60%. This last constraint protects the network from the congestion in order to ensure people from the organization to be able to access any files in the private directory at any time. The used XACML policy is described in [14].

The edge router hosts an SNMP agent that provides the bandwidth rate. In order to simplify our example, we consider that there is a management object that carries this value. Actually, this is not true in most cases and the standard Management Information Bases (MIB) imply the SNMP manager to send several requests to calculate this value [5].

Considering the XACML dataflow model and mainly steps 5 to 10, every time a user accesses any file in the public directory, the context handler needs to ask the PIP the value of the bandwidth use rate. The PIP, which also acts the role of SNMP

manager, gets this value by sending a message SNMP GET-REQUEST. When the SNMP agent receives the request, it looks in its MIBs and returns the value through a message SNMP GET-RESPONSE. Then, the PIP can reply to the context handler that can provide the attribute to the PDP. Now the PDP can evaluate the policy.

The main issue is this process consumes both time and network bandwidth. In addition, the network is busy being an exceptional event emphasizes the uselessness of systematically evaluating the bandwidth use rate. The next section analyzes the characteristics of such kind of attributes.

3.2 Definitions

As the example above shows, it is not necessary to evaluate each time if the bandwidth use rate is greater than 60% because this state should be exceptional. We name stable conditions this kind of constraints.

Definition 1: A *stable condition* can be viewed as an expression that always returns the same result during a given period considered to be long.

In our example, the condition "the bandwidth use rate is less than 60%" is mostly always true. Another example, which is more common, could be "the current time should be between 7:00 and 19:00". This example is less demonstrative because the requesting process is faster. This comes from the current time value is given by a process running on the same machine as the XACML access control solution. But, like the bandwidth example, this expression returns always true from 7:00 until 19:00 and always false between 19:00 and 7:00.

How to identify/characterize a stable condition? First, we define *intrinsic attributes* of request as the attributes sent by the PEP to the Context Handler in an authorization request. Authorization requests contain at least attributes about the requesting subject, the target resources and the planed action. In our example, the role of the user, the name of the resource and the name of the action are intrinsic attributes. Based on this definition, we characterize a *stable condition* as:

Definition 2: A *stable condition* is an expression where every argument does not directly or indirectly depend on the value of one of the intrinsic attributes of the request.

A condition has to satisfy both definitions 1 and 2 to be considered as stable.

Definition 2 implies that a stable condition can contain only environmental attributes. Any subject attribute that is not present in a PEP's request should be retrieved using one of the subject attribute inside the request, which means a subject attribute depends on the value of an intrinsic attribute. The same reasoning can be applied to resources and actions attributes.

As a consequence, a stable condition can occur only inside a XACML condition element because XACML target elements only consider subject, resource and action attributes.

We call *eligible stable condition* a condition that satisfies definition 2. In addition to this static characteristic, definition 1 implies the analysis of the successive results returned by eligible stable conditions evaluations to check their invariance.

4 XACML Extensions

4.1 Proposed Architecture

The evaluation of stable conditions being useless, we propose to enhance the XACML framework in such a way stable conditions are considered only when it's necessary. In addition, we want enhanced XACML systems to be still compliant with the original XACML specification.

Our idea is: (1) remove stable conditions from the policy evaluated by the PDP; (2) and modify this policy according the changes of stable conditions values.

Let *R* be a policy rule that includes a stable condition *C*. Let *R-True* (resp. *R-False*) be rule *R* without *C* when *C* returns true (resp. false). For example, rule PublicAccess will be divided into rules PublicAccess-True when the network is operational and PublicAccess-False when it is congested. See [14] for these rules.

In reaction to this event, the PIP will reflect the stable condition changing within the XACML framework by alerting the PAP (See NotificationMessage in Fig. 3), which is the entity responsible of the policy administration. The PAP should be able to dynamically modify of the rule(s) (*R-True* into *R-False* or vice-versa) concerned by this changing. The PAP stores both *R-True* and *R-False*. It can also read a configuration file that indicates what rule(s) should be changed for each notification message.

In order to make the PDP switches *R-True* to *R-False* (and vice-versa), we add XACML the ability to adopt an event driven behavior. The idea is to benefit from asynchronous notification mechanisms to deal with management of stable conditions.

First, we make the PIP being able to be notified by some environment attribute value providers. It can receive notification messages that inform it the value of a stable condition attribute has changed in such way the value of the associated stable condition has also changed.

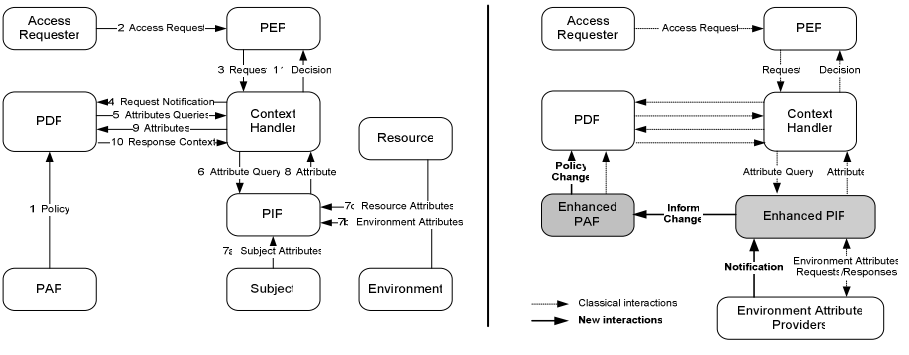


Fig. 2. (a) Simplified XACML data flow model (b) Proposed architecture

This new architecture allows the PDP to use either *R-True* or either *R-False* during policy evaluation without neither having to consider *C* nor having – a fortiori - to retrieve the values of the stable condition attributes in *C* as it should have achieved in the case of *R* evaluation. The decision to use either *R-True* or *R-False* is dynamically driven by the significant changes of the stable condition attribute values. In Fig. 2.(b)

we show the impacts of our proposition on the XACML standard framework are limited. Classical entities behaviors and interactions between them are not modified. The behavior of both PAP and PIP has been enhanced without enabling their standard activities. The OASIS standard roles of management entities are conserved. The PIP is still acting as an information collector, and the PAP as the policy administration point.

Initially, the PAP sets the PDP with a policy that includes R-True or R-False depending on what the value is when the status of the system is operational. The PIP has subscribed to a trusted Environment Attribute Provider (EAP) its interest to be notified when C is changing. The PIP is listening for relevant notification messages issued from the EAP. When such a notification is received, the PIP builds a specific XACML-based message mentioning the C condition and its current value. This message is then sent to the PAP. This latter evaluates the value of the condition. If the value is false, it changes the current policy by using the R-False rule. When it is true, R-True is loaded. Note that during this process, no synchronous interaction is achieved between the XACML framework and the EAP concerned by the stable condition attributes. The evaluation process concerning the non stable condition attributes is achieved in the standard way.

4.2 Description of the Implementations

In order to prove our approach is more efficient, we have implemented two prototypes for the scenario described in section 3 using the synchronous and the asynchronous methods. Our testing environment is the following one:

- For the edge router, a PC with a Pentium Core 2 Duo 2.13GHz, 1Gbyte RAM and Linux Kubuntu DAPPER 6.06.1 LTS. We use NET-SNMP version 5.2.1.2 (http://net-snmp.sourceforge.net/) that provides an SNMP agent and a command line tool to send SNMP traps.
- For the FTP server, is a PC with a Pentium Core 2 Duo 1.66 GHz, 1Gbyte RAM and Windows XP Pro. We use the sun's XACML implementation version 1.2 (http://sunxacml.sourceforge.net/) that provides an XACML PDP and also a java API for implementing PEPs, PIPs and PAPs. In addition, we use the java API SNMP4J version 1.8.2 (http://www.snmp4j.org/) to implement the SNMP client and the SNMP trap server.
- For the network, Ethernet 100Mbps network. The router is connected to the FTP server using a switch (there is no router between them).

Prototype 1: The Basic XACML Architecture

The architecture of the first prototype is described in Fig. 1. Each time any one wants to access the public directory, the PDP asks the specific PIP to request the MIB value of the current bandwidth rate. Because NET-SNMP only provides the standard MIB we have simulated it by implementing a PIP that only requests the IF-MIB::ifInOctets MIB object (i.e. the number of bytes in the input interface). We suggest to read [5] to get the real process for calculating the bandwidth value. When IF-MIB::ifInOctets modulo 100 is less than 90, the PIP responds a value less than 60 else it sends a value bigger than 60. Finally, the PDP evaluates the policy.

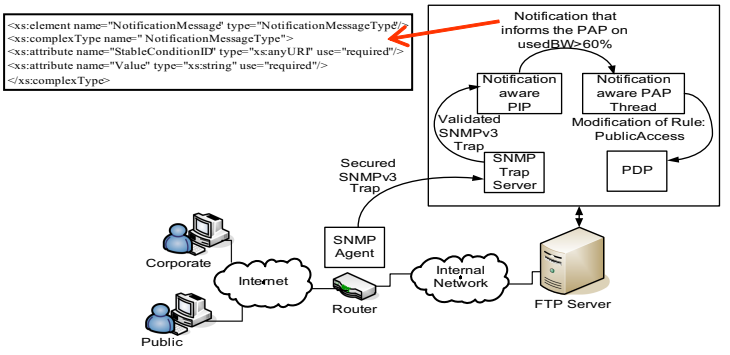


Fig. 3. Example of implementation

Prototype 2: The Extended XACML Architecture

The architecture of the second prototype is depicted in Fig. 3. SNMPv3 traps are sent from the edge router to an SNMP trap server that runs on the FTP server. In respect with our definition of a trusted EAP, we have chosen to use SNMPv3 traps because this version provides authentication, integrity, confidentiality and anti-replay mechanisms [6]. In this prototype, we have used MD5 for authentication and DES for confidentiality. Traps are sent to alternatively indicate the bandwidth use rate is greater than 60% (OID:.iso.org.dod.internet.experimental.siera.bw.greater60) and traps that indicate the bandwidth use rate is less than 60% (OID: .iso.org.dod.internet.experimental .siera.bw.less60).

When traps are received by the SNMP trap server, they are validated (integrity and confidentiality) and forwarded to the notification aware PIP. Then, the PIP translates OID of the stable condition changing to its associated standardized notification message. The PIP informs the PAP of the changing that is identified by the URN. Finally, the PAP modifies the public access rule. When the received notification is <NotificationMessageStableConditionID = urn:irit:siera:notification:name:bwlessthan60 Value="false"/>, the new rule denies the access to the public directory. When the notification is <NotificationMessageStableConditionID = urn:irit:siera:notification:name:bwlessthan60 Value="true"/>, the access is granted.

4.3 Experimental Tests

Our experimentation consists in three tests:

Test1 - Using the first prototype, we send five times one hundred requests "a user wants to access the public directory". We calculate for each request the time for getting the response.

Test2 – Using the second prototype, we send five times one hundred requests "a user wants to access the public directory". We calculate for each request the time for getting the response.

Test3 - Using the second prototype, we send five times one hundred traps (alternatively .iso.org.dod.internet.experimental.siera.bw.greater60 and

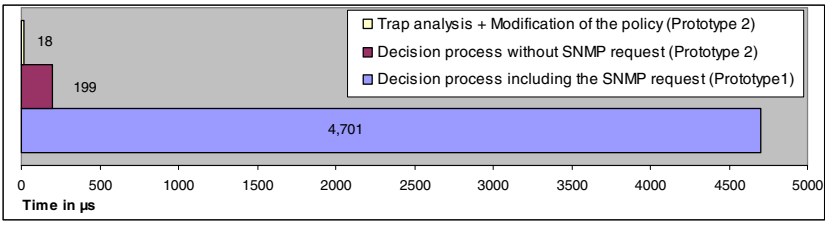


Fig. 4. Experimental results

.iso.org.dod.internet.experimental.siera.bw.less60). We calculate for each trap the time for analyzing the trap and modifying the policy.

Fig. 4. provides the average time in microseconds for each test (column 1, 2 and 3 are respectively the average time for test 1, 2 and 3). It is interesting to note the impact of the SNMP request/response process handling (i.e. the creation of the SNMP request, the SNMP agent process and the analysis of the SNMP response) on the decision making process. The decision process making without the stable condition is 23 times faster. Especially, these tests have been performed in 100Mbps Ethernet network that does not include any high level intermediate device, which could slow down the SNMP messages. Also, the process of receiving a trap, analyzing it and modifying the policy is short according the global decision process. It represents 0.3% of the decision process that includes the SNMP request (test1) and 8.7% of the decision process without SNMP request (test2).

4.4 Analysis

Based on our experimental results, we can conclude our approach, which extends the XACML framework to support asynchronous notifications, enhances the performances of:

1. the *decision making process*. The ratio between our extension and the basic XACML framework is extensive (1/23 in our test scenario).
2. the *underlying network*. When the PIP sends to a remote server a request for each user's request, at least two messages are sent over the network (in test1, we have sent 2*500=1000 SNMP messages). Notifications require only one message to be opportunely transferred on the network (in our case, only one SNMP trap is sent and only when it is necessary.)

By giving capabilities to XACML systems to benefit from asynchronous notification processing, we increase their ability to deal with scalability. This leads a XACML server able to process more requests during the same time. In our scenario, a simple calculus indicates a PDP can evaluate up to 5000 requests "a user wants to access the public directory" per second in our extended XACML prototype, which uses asynchronous notifications, compare to only 212 requests per second with the basic XACML architecture with synchronous messages. Thus, it allows administrators to consider fine grain rules – considering environment attributes – when they specify access control policies. Particularly, it gives them credits that exceptional situations in policies will not inevitably slow down the XACML server.

5 Related Works

Privileges Management Infrastructure (PMI) solutions (e.g. PERMIS [7] or Akenti [8]) propose two ways for retrieving attributes: the push and pull models. Their architecture is slightly different from XACML as there is no context handler and no PIP explicitly mentioned. Thus, the PEP talks directly to the PDP. In the push model, the PEP provides the PDP all the attributes. For example, the PEP sends an attribute certificates with the Distinguished Name (DN) of the user and its roles. In the pull model, the PEP provides some of attributes. The PDP completes the list of the required attributes. In this case, the PEP only sends the DN of the user and the PDP gets the attributes certificate in a LDAP repository based on the user DN. Environment attributes are limited to time or date.

Important research works have been carried out to deal with specific requirements in context-sensitive environments like mobile and pervasive computing, ambient intelligent systems or smart spaces. They focus on proposing "context-aware" architectures and/or frameworks. It means the context leads the behaviour of the system. Inspired from the PMI push model, Al Muthadi et al. [9] have proposed a "Context Provider" that can get contextual information from either sensors or others data sources. The "Inference Engine", which is the core of their context-aware security systems, can either query a Context Provider for context values or ask it to be notified when a condition changes. However, they do not explain when/why request/query or notification messages should be used. No mechanism improves the decision making process. The same approach is adopted in [10] where a "Dynamic Context Service" is a trusted entity used for acquiring context information either directly or via a third party. A policy governs the frequency at which the context is acquired and updated and specifies some thresholds for notifying the core system. However, when a condition changes, the whole policy is re-evaluated that is a bottleneck for high speed environments.

Covington et al. have proposed in their initial works a PMI pull-like model named Generalized RBAC [11]. This model adds environment roles that can be activated or not. They use caching techniques and add some mechanisms to estimate the freshness of their values. The approach improves the performance of the decision making process because deactivated environment roles are not considered during the evaluation process. But, caching techniques cannot be used in our context, as stable condition changing notifications should be considered immediately. More recently, Convington and Sastry have proposed the Contextual ABAC (CABAC) model [12] that is the update of GRBAC principles to the ABAC model. In the same way, environment attributes could be activated/deactivated. They just propose a conceptual access control model that doesn't deals with implementation issues. Despite, they point out the fact that some of environment attributes need to be evaluated every time while others are subject to non changing conditions during a entire session, no clear ratiocination has been developed on this subject.

6 Conclusion

We have shown in this article that all the attributes should not be retrieved in the same manner. We have defined what stable conditions are and why they should not be

treated by the XACML systems like usual conditions. We have provided an architecture that improves the performance of the decision making process when it deals with stable conditions. This architecture is compliant with existing XACML specification. Finally, we have presented our test environment, and our experimentations that prove our solution is better.

However, our solution has two drawbacks: the administrator should know which of the conditions are stable and (s)he needs to configure the EAP, the PIP and PAP. As a consequence, it is more difficult to specify a policy now.

Our future work, stimulated by the results of this experimentation, will focus on automating this process. The administrator will specify a security policy without taking into account the existence of stable conditions. The decision making process will monitor the results of each eligible stable condition to provide information for automatically detecting stable conditions. When a stable condition is found, an optimization process will configure the EAP, the PIP and the PAP to use asynchronous notification mechanisms. This works aim to endow an XACML framework with a self-optimizing behaviour, which is one of the Autonomic computing properties defined in [13]. This objective brings many issues such as how to provide an implementation that automatically detects if a condition is stable or not based on the characterization given in definitions 1 and 2.

Others perspectives concern how to dynamically change the communication between the PIP and the extended EAP from being synchronous to asynchronous. Moreover, our current study is limited to SNMP notification based EAPs. Because many management solutions (e.g. NAGIOS, WBEM Server, WebServices, etc…) exist, we have to define an integrated approach to bring on the interoperability between all these different notifications sources and our extended XACML EAP.

References

[1] Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in Operating Systems. Communication of the ACM (1976)

[2] Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST Standard for Role-Based Access Control. ACM TISSEC 4(3), 222–274 (2001)

[3] Wang, L., Wijesekera, D., Jajodia, S.: A Logic-based Framework for Attribute based Access Control. In: 2nd ACM Workshop on FMSE (2004)

[4] eXtensible Access Control Markup Language (XACML) version 2.0, OASIS Standard (February 2005), http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os

[5] How To Calculate Bandwidth Utilization Using SNMP, CISCO TN 8141 (2005), http://www.cisco.com/warp/public/477/SNMP/calculate_bandwidth_snmp.htm

[6] Blumenthal, U., Wijnen, B.: User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), IETF RFC 3414 (2002)

[7] Chadwick, D., Zhao, G., Otenko, S., Laborde, R., Su, L., Nguyen, T.-A.: Building a Modular Authorisation Infrastructure. In: The UK e-Science All Hands Meeting (2006)

[8] Thompson, M.R., Essiari, A., Mudumbai, S.: Certificate-based Authorization Policy in a PKI Environment. ACM TISSEC 6(4) (2003)

[9] Al-Muhtadi, J., Ranganathan, A., Campbell, R., Mickunas, M.D.: Cerberus: A Context-Aware Scheme for Smart Spaces. In: 1st IEEE International Conference on Pervasive Computing and Commmunications (PerCom 2003) (2003)
[10] Wullens, C., Looi, M., Clark, A.: Towards Context-aware Security: An Authorization for Intranet Environments. In: 2nd IEEE annual conference on Pervasive Computing and Communications workshops (PerComW 2004) (2004)
[11] Convington, M.J., Fogla, P., Zhan, Z., Ahamad, M.: A Context-aware Security Architecture for Engineering Applications. In: 18th annual Computer Security Applications Conference (ACSAC 2002) (2002)
[12] Convington, M.J., Sastry, M.R.: A Contextual Attribute –Based Access Control Model. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1996–2006. Springer, Heidelberg (2006)
[13] Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. Computer 36(1), 41–50 (2003)
[14] Laborde, R., Desprats, T.: Dealing with Stable Environmental Conditions in XACML Systems. In: ICSNC 2007 (2007)

LVD: A Lightweight Virtual Desktop Management Architecture*

Xiaofei Liao, Xianjie Xiong, Hai Jin, and Liting Hu

Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
{xfliao,hjin}@hust.edu.cn

Abstract. Rapid improvements in network bandwidth, ubiquitous security hazards and high total cost of ownership of personal computers have created a growing market for desktop virtualization. We present the LVD, a system that combines the virtualization technology and inexpensive personal computers (PCs) to realize a lightweight virtual desktop system. Compared with the previous thin client systems, LVD supports the backup, mobility, suspending and resuming of per-user's working environment; it supports the customization of operating system and applications for each user; it supports synchronous using of incompatible applications on different platforms; it achieves great saving in power consumption. LVD consists of five modules--- the template-based VM repository, the data center, the application cluster, the VM central manager, and the client terminal, in which we have proposed wRFB protocol, magnet algorithm and the like to perform the above functions. We have implemented LVD in a cluster with VMs and compared its performance against widely used commercial approaches. Experimental results demonstrate that LVD is effective in performing the functions while imposing little overhead.

Keywords: RDP, desktop virtualization, virtual machine.

1 Introduction

Although server virtualization is now a mainstream technology for data centers and cluster computing communities, desktop virtualization has attracted considerable attention for recent years, both academically [9] and commercially. Desktop virtualization has the potential to offer a new, cost efficient paradigm shift to ease the demand for the resources while maximizing return on investment. Combining the potential cost

* This work is supported in part by National Natural Science Foundation of China (NSFC) under grants No.60703050, National 973 Basic Research Program of China under grant No.2007CB310900, Hubei Natural Science Foundation under grant No.2007ABD009, the Ministry of Education-Intel information technology special research fund under grant No.MOE-INTEL-08-06 and Wuhan Chengguang Plan under grant No.200850731350.

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 25–36, 2008.

savings with other advantages such as disaster recovery, robustness, scalability, and security make this an attractive computing model to deploy.

However, the architecture of current virtual desktop systems imposes fundamental limitations on the customization, convenience, acceleration of pixel data transfers and energy-savings, so the challenges remain as below.

First, most virtual desktop systems fail to support users to customize their own environments and back up them, not to mention the function of suspend/resume. The following scenarios happen frequently: people have different working environments at home, at the office and during the trip. At home, one may want a PC with more entertainment software, while at the office a PC with corporate applications is needed. Furthermore, people should logically suspend a machine at one Internet site, then travel to some other sites and resume on another machine. We call the hypothetical capability *suspend/resume*. Nevertheless, some previous work ignores the above needs. Collective [4] deploys the system disk for each user, the contents of which at every boot are made identical including the image's operating system and applications.

Second, it is impossible to interact with incompatible applications of different operating systems synchronous without switching the platforms. The reason lies in that the server side of all virtual desktop systems uses the remote display protocol to transfer full-screens to the client side. Hence the user has to face multiple screens and switch among them.

The third shortcoming follows from the second. As a large granularity transferring unit, full-screens increase the amount of data transmitted, resulting in more bandwidth consumption and performance degradation. As a matter of fact, only the application GUI that the user interacts with needs to be transmitted.

Finally, most virtual desktop systems fail to consider the reduction of power consumption of the data center. Undoubtedly, *green computing* has been a hot idea in cluster computing for recent years. Anecdotal evidence from data center operators [7] indicates that a significant fraction of the operation cost of these centers is due to power consumption and cooling. To lower the cost of management, we should give prominence to the problem of power consumption.

For the first challenge, WebOS is a good option to get rid of the space limit. However, since all operations are performed by JAVASCRIPT which is less than 1/50 of Java, 1/200 of php and 1/500 of C for speed, WebOS can only be applied to deal with lightweight programs. Besides, high network latency will lead to slow response time. For the second challenge, VDI [14] combines server virtualization with remote presentation technology. However, the user still has to face two operating platform screens while interacts with applications on different platforms. Little previous work concerns the third challenge. For the fourth challenge, lots of previous works have been done to reduce the power consumption, from the perspective of both local techniques and cluster-wide techniques [11]. Nonetheless, previous schemes seldom take advantage of the virtualization technology and the characteristic of particular platform to save the energy.

We propose *lightweight virtual desktop* management architecture (LVD) to address the challenges. Our model consists of five modules: the template-based VM repository (TVR), the data center (DC), the application cluster (APPC), the VM central manager (VCM), and the client terminal. The key technologies behind these modules include *wRFB protocol*, *Magnet algorithm* and the like. LVD provides a comprehensive suite of important functions:

- Backs up, suspends, resumes per-user working environment (see TVR).
- Supports rolling back to different working environment by continuous backup at different working locations (see DC).
- Supports using incompatible application of heterogeneous platforms in a single platform (see APPC).
- Saves energy in a dynamical and global way (see VCM).

We have implemented LVD in the cluster with VMs and measured its performance on real applications. We also have compared our LVD prototype system against the most current and widely-used virtual desktop systems, including Microsoft Remote Desktop, Citrix MetaFrameXP and Sun Ray. Experimental results demonstrate that LVD is effective in performing the functions while imposing little overhead.

The rest of this paper is organized as follows. Section 2 discusses the related work. The LVD architecture is described in Section 3. Section 4 evaluates the performance. We conclude our work in Section 5.

2 Related Works

Virtual machine can simulate all the hardware components, like memory, disk, CPU and so on. A computer with virtual machine monitor installed can simulate several virtual machines at the same time. Each of these simulated virtual machines can independently install operating system, run applications simultaneously. The virtual machine monitor could suspend the virtual machine managed by the VMM, and resume one virtual machine which has been suspended before. The VMM could take a snapshot of a virtual machine's memory state, and restore the machine state to the state when the snapshot is created.

We divide the previous virtual desktop techniques into two groups: the physical machine based (PM-based) techniques and virtual machine based (VM-based) techniques. The PM-based techniques use the *remote display protocol* or *WebOS* approach to realize the desktop virtualization, whereas VM-based techniques improve the traditional work by deploying the virtual machines in the server side or client side and thus benefits from better fault isolation and higher resource utilization.

Remote display protocol based technique. As the earliest desktop virtualization technique, the thin client computing model uses a remote display protocol to communicate between a server and a client over the network. The protocol allows graphical displays to be virtualized and served across a network to a client device, while the application logic is executed on the server. Typical thin client protocols include VNC [12], RDP, THINC [1], pTHINC [8] and the like. Since the technique mainly depends on continuous display synchronization between user interface on the client and application logic on the server, how to increase the display efficiency is the main challenge. Being the first thin client capable of transparently playing full screen video and audio at full frame rate in both LAN and WAN environments, THINC is the groundwork of other solutions.

WebOS based technique. WebOS is another fashion that can quickly build on-demand virtual desktop environments using web technology. A WebOS is also known as a webtop, and Ebrahim Ezzy's definition probably explains it best: "A webtop (derived

from "desktop") pushes that replication to its limit. Also known as a WebOS, it is basically a virtual desktop on the web. It is a simple, less bloated, less featured and remotely accessible operating environment that runs in a browser. It delivers a rich desktop-like experience, coupled with various built-in applications." Typical WebOS systems include YouOS [5], EyeOS, Glide [6] and Orca desktop [10]. The advantage lies in that less bandwidth consumption and better cost-performance. However, the drawback is obvious: since a web application executes the front end on the client, and the back end on the server, it fails to be compatible with the original applications and thus requires the reprogramming of them.

In sum, PM-based technique has drawbacks including that (1) the user's working environment can not be suspended and resumed; (2) the full screen image is transmitted from the server side to the client side resulting in a great amount of transferring data and degradation of the system performance.

Server virtualization based technique. To simplify administration and to reduce management and operating costs while maintaining reliability and safeguarding against disasters, the significant benefits of server virtualization technology is now being applied for companies' desktop users. *Virtual Desktop Infrastructure* (VDI) is such an integrated desktop virtualization solution combining server virtualization with remote presentation technology. The desktop operating systems and applications run inside the virtual machines and the server side distributes such VMs to each user on the client side. Users use a remote display protocol to access the VM on the server and the server returns the full features of the VMs.

Client virtualization based technique. The typical systems are *Internet Suspend/Resume* (ISR) [13] designed by CMU and Collective designed by Stanford. ISR is a mobile computing technology which can preserve one's uniquely customized computing environment as one package and then moves it to different locations. ISR is implemented by layering virtual machine technology on distributed file system technology. It enables a hands-free approach to mobile computing in which commodity hardware may be widely deployed for transient use. Through rapid and easy personalization and depersonalization of anonymous hardware, a user is able to suspend work at one machine and to resume it at another.

Generally, server virtualization based technique (1) trades off the user's ability to customize their own environment; (2) forces the user to switch among multiple platforms while the user wants to run incompatible applications of different operating systems. Client virtualization based technique requires a powerful client side running the VM.

3 LVD Design

The motivation of designing LVD is to solve the challenges discussed above: (1) how the applications be customized and configurable freely by the end user; (2) how per-user states be backed up in a storage-saving way and an optional data-sharing way; (3) how the incompatible applications running on different operating systems be used by the end user at the same time without switching the platforms; (4) how the great savings be achieved in power consumption by our scheduling policy; (5) how the application display commands be transmitted effectively.

3.1 Overview

As illustrated in Fig.1, LVD system consists of *template-based VM repository* (TVR), *data center*, *application cluster* (APPC), *VM central manager* (VCM), and *client terminal*. We start by showing how LVD works from a user's perspective.

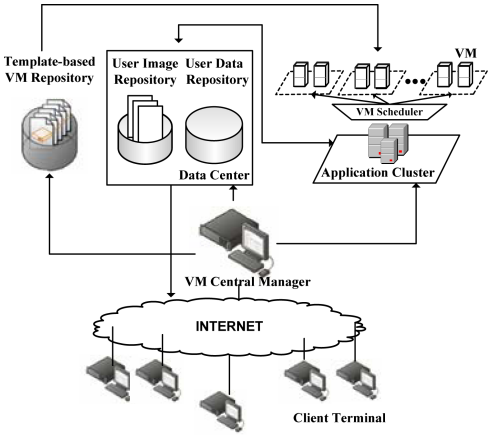


Fig. 1. Architecture of LVD system

When a user logs in at the first time, firstly, he is presented with a list of appliances by the VCM on which he can click to begin constructing his personal working environment. Second, the VCM informs the template-based VM repository to create the customized VM image including the operating system and applications. Meanwhile, the VCM chooses an appropriate physical machine to boot the image among all the application servers in the cluster on the basis of load balancing and energy saving. Third, the user accesses the running VM using our remote display protocol (*wRFB*). The user transmits the commands to the VM and the VM returns the application window updates to the user interface. Finally, when the end user exits, his working environment consisting of the user image and user data is backed up in the data center.

If the user has already logged in previously, he fetches and runs the latest copies of VM image from the data center over the Internet, or resumes the suspended VM image locally when the network is unavailable.

3.2 Template-Based VM Repository (TVR)

The template-based VM repository is responsible for customized VM image provision and updating. TVR contains all versions of various operating systems and software. Once a user's request arrives, the ordered operating platform and applications (or components) will be encapsulated and thus a unique VM image is created.

Moreover, TVR need to be updated constantly. All software upgrades, be they small or big, is accomplished in our system with the same mechanism. The system

administrator prepares a new version of the appliance and deposits it in the repository. TVR can inform the user that a new version of the appliance is available, encouraging the user to reconstruct the working environment, or the TVR can even force a reconstruction to disallow use of the older version.

This update approach has some advantages over package and patch systems like yum [15], RPM [2], Windows Installer, and Windows Update. Patches may fail on some users' computers because of interactions with user-installed software. Our updates are guaranteed to move the appliance to a new consistent state. Users running older versions are unaffected until they perform a reconstruction.

3.3 Data Center (DC)

The data center performs the following functions:

- Backs up per-user state in terms of copies of VM images.
- Separates per-user VM image into system data and user data.

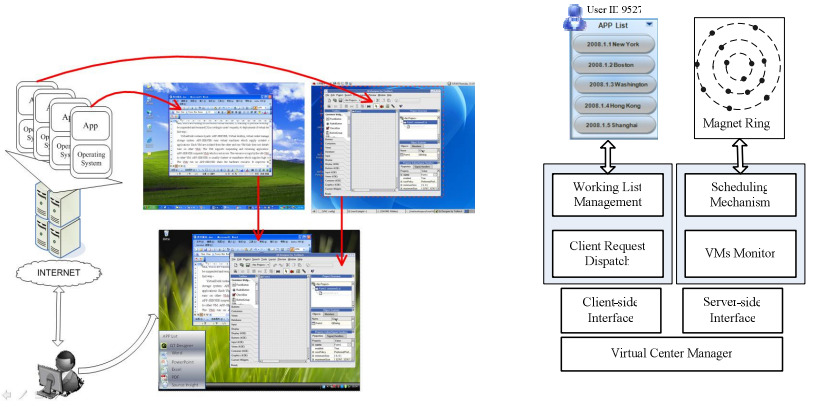
From the view of the data organization, we use the NFS protocol to store the data, which is fast, reliable and simple to support demand paging of large objects, like the images. For our prototype, NFS has the following advantage: in a typical computing session, a user may not access most of the virtual disk image. We arrange the disk data in a tree of small files, only a small number of these files will need to be transmitted to the booted VM in the application cluster. The on-demand transmission mode helps a lot when the bandwidth is limited.

From the view of the data type, the data in DC is divided into two parts: system data and user data. The system data consists of an operating system and all installed applications. User data consists of a user's profile, preferences, and private files. The separation has the following advantages: (1) the files in the user data repository could be classified and protected according to their security sensibility; (2) if a user wants to share his data, e.g. movies and files, others can get access to them through NFS conveniently.

3.4 Application Cluster (APPC)

After the customized VM image is created by TVR, or fetched from DC, APPC will boot a customized VM on one of its physical machine to provide service to the remote end user. The highlights of APPC module is an advanced remote display protocol *wRFB* (window-based remote frame buffer).

In the previous virtual desktop systems, such as VDI, Sun Desktop Virtualization Solution and Windows *Vista Enterprise Centralized Desktop* (VECD) [4], each user is assigned to a virtual desktop. If the customer wants to use applications of other platforms, he faces multiple desktops from different VMs and has to switch among them. Unlike the previous work, wRFB grabs and transfers the application GUI images from multiple VMs to the client side. Then all these window images will be merged and displayed on a single virtual desktop, as illustrated in Fig.2(a).



(a) (b)

Fig. 2. (a) wRFB protocol; (b) The logic diagram of VCM

3.5 VM Central Manager (VCM)

VCM provides the follows interfaces, as illustrated in Fig.2(b): 1) client-side interface: authenticating the user and keeping the working records of each user. For example, a user may login to the client in Hong Kong, work for a while, log out, travel a great distance to Seattle, login, work, log out, and come back to Hong Kong. It is highly possible that he wants to resume the working environment in Hong Kong a few days ago rather than Seattle. Therefore, it is necessary to trace each user's records at different locations; 2) server-side interface: locating and scheduling the VMs in APPC dynamically in a load balancing and energy saving way.

Undoubtedly, the concept of *green computing* has attracted much attention recently in cluster computing. The migration of the VM inspires us with a novel method called *Magnet* to reduce the power consumption, which has been introduced in detail in our previous work [16]. The key technique is to consolidate the load among the nodes on a multilayer ring-based overlay and then keep the redundant nodes in *low power* state like *deep sleep* or *shut down*.

3.6 Client Terminal

The client terminal can be any form of computing hardware, from desktops to handheld or wearable computers. The highlights of the client terminal module lie in as follows. On high bandwidth (e.g., 100Mbps) networks, the system performs well, the challenge here is that a terminal may be disconnected from the network or the bandwidth is highly occupied. We provide the following solutions: (1) if the terminal is mobile computers like laptop, store the latest VM image locally; (2) if the terminal is a thin-client, compare to the standard operating system and application software (e.g., Windows 2000 and the Microsoft Office suite), the user specific files are a very small fraction of the image size. Therefore, we distribute the large standard disk image-based blocks widely over the network which can be accessed at high bandwidth from

a nearby site. Only the much smaller user specific state needs to be transmitted at low bandwidth from the data center.

4 Performance Evaluation

We provide some quantitative measurement of the system to give a sense of how the system behaves and study the system performance in a various network environment.

As shown in Fig.3, our testbed consists of six computers connected on a switched Fast Ethernet network: two thin clients, a data center, a virtual central manager, a packet monitor, a template-based VM repository, a network emulator for emulating various network environments, an application server, and a web server used for testing web applications. All computers have an AMD Athlon 3500+ processor and 1GB DDR RAM. Storage is accessed via iSCSI protocol from a NetApp F840 *network attached storage* server (NAS). The guest kernel is Linux 2.4.18 ported to UM-Linux, and the host kernel for UM-Linux is a modified version of Linux 2.4.18. The virtual machine is configured to use 512MB of RAM, the memory page size is 4KB, page fault service time is 10ms, and the context switch time is 0.1ms.

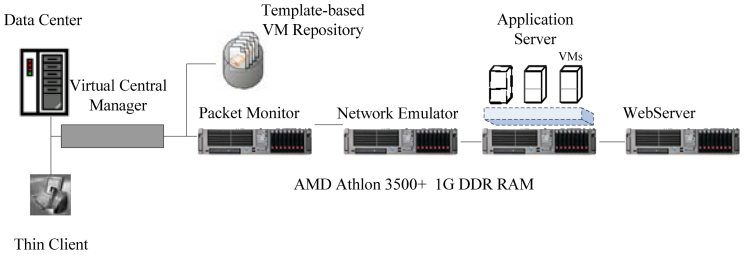


Fig. 3. Tested configuration

We have assessed the LVD display performance in a various network environment. We run each of the three application benchmarks on the baseline PC platform and the five thin client platforms on different operating systems. The five specific platform configurations considered are PC running LVD on Linux (LVD), Microsoft Terminal

Table 1. Characteristics of the application benchmarks

Benchmark Type	Application Benchmark	Operation
Latency benchmark	Java applet	Typing a character, scrolling text, filling a screen region. Downloading an image
Web-based benchmark	Web text page load	Downloading a sequence of 54 web pages. Scrolling down 200 pixels
Multimedia-oriented benchmark	Flash benchmark	Streaming a 98 KB Macromedia Flash animation chip from the server side (uses vector graphics and contains 315 550*400 frames)

Services RDP 5.0 on Windows 2000 (RDP Win2K), Citrix Metaframe 1.8 running on Windows 2000 (Citrix Win2K), LapLink 2000 on Windows NT 4.0 Terminal Server Edition (LapLink WinNT), and Sun Ray.

As illustrated in Table 1, we use a simple Java latency benchmark to measure the latency of basic operations on a thin client platform and a selection of benchmarks from the Ziff-Davis i-Bench benchmark suite to provide a measure of web-based and multimedia-oriented application performance.

4.1 Latency Benchmark Results

We measure both the latency and the data transferred for each of the four operations: draw letter, fill red box, scroll text, and load bitmap. These results are shown in Fig.4(a) and Fig.4(b), respectively.

Advantage. Compared with other platforms, LVD is able to complete any operations within 100ms, including the basic draw letter, fill box, and scroll text operations. Comparing Fig.4(a) with Fig.4(b), we can see that the latency and the data transferred are not at all correlated in many cases. While LVD had less latency for most of the tests, it also requires more data transfer.

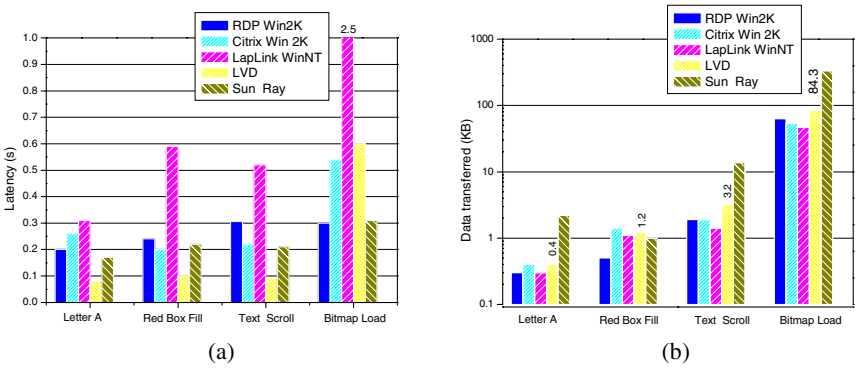


Fig. 4. (a) Latency test time to completion; b) Latency test data transferred

4.2 Web Page Load Benchmark Results

We measure the total time required to display all 109 web pages of the web benchmark and logs complete packet traces using the packet monitor. Fig.5(a) and Fig.5(b) illustrate the total normalized web page download times and total data transferred for each run, respectively.

As shown in Fig.5(a), almost all of the platforms complete the web benchmark in less than 50 seconds at network bandwidths of 4 Mbps or greater, corresponding to an average of less than half a second per page.

Limits. At network bandwidths below 4 Mbps, the performance of the thin client platforms begins to degrade. As shown in Fig.5(a) and Fig.5(b), LVD does not take

longer to complete the web benchmark at lower bandwidth, but instead lose data resulting in missed or incomplete screen updates. On the other hand, RDP and Citrix behave in a similar manner by taking longer to complete the web benchmark as the network bandwidth decreases, but continuing to send the same amount of data even at lower bandwidths.

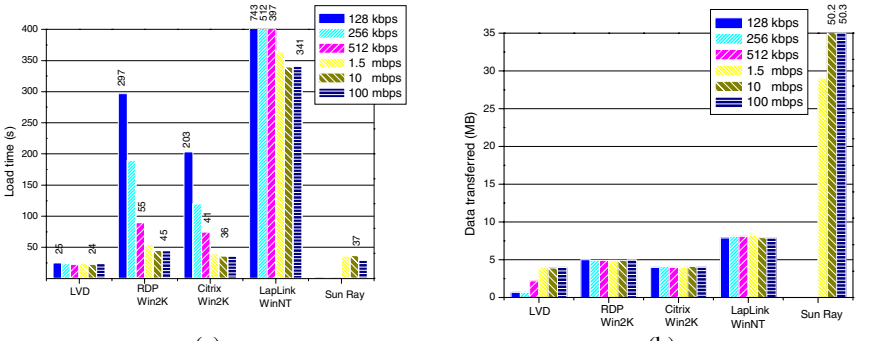


Fig. 5. (a) Web page load time; (b) web page data transferred

In general, the LVD results indicate that for web-based applications, a pixel-based encoding approach could encode screen updates with comparable efficiency as graphics-based encoding approaches.

4.3 Flash Benchmark Results

We run the flash animation test on each of the thin client platforms via network, and also run the benchmark on the baseline platforms for comparison. The results are illustrated in Fig.6(a) and Fig.6(b).

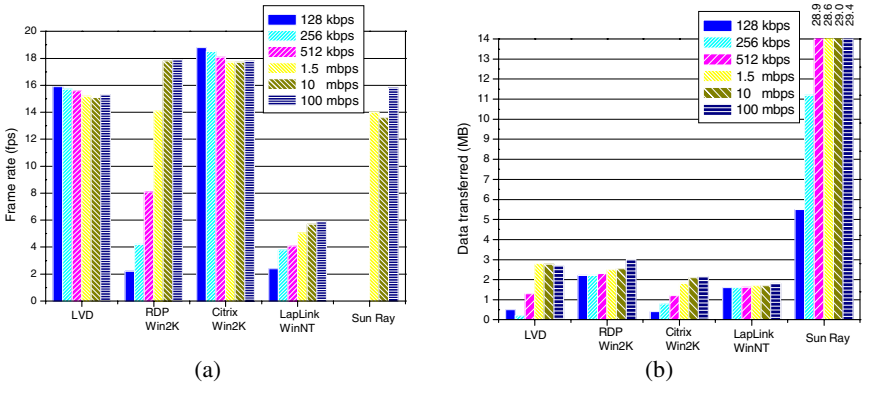


Fig. 6. (a) Flash test frame rate; (b) Flash test data transferred

We find that a frame rate close to 16fps produced good subjective results, with smooth display, no skipped screens, and no tearing or jerky movement. The 100KB animation is downloaded completely to the server prior to playback, so the bandwidth does not affect the frame rate on the baseline.

Limits. With the web test results, LVD is slightly less efficient than the platforms with graphics based encodings. LVD maintains an almost constant rate for the benchmark at all bandwidths, but drops many screens on the client's display at the low end of the range. Its performance is only comparable in smoothness to the baseline when the data transferred reaches a plateau. In general, LVD system experiences a similar fall-off in performance at bandwidth 128kbps or below, indicating that LAN bandwidths are required to support multimedia applications.

4.4 Power Reduction Results

After analyzing the characteristics of workflow of LVD, we have proposed a new schedule policy called *Magnet* for cluster with VMs to achieve the energy savings [16]. We apply the policy to the application cluster. The experimental measurements show that the new method can reduce the power consumption by 74.8% over base at most with certain adjustably acceptable overhead.

5 Conclusions

This paper presents LVD, a prototype of system management architecture for managing desktop computers. This paper concentrates on the design issues of a complete system. By combing *wRFB protocol* and *Magnet algorithm* and other techniques of virtualization, LVD provides several novel functions including backup, mobility, suspending and resuming of per-user's working environment, the customization of working environment and the synchronous using of incompatible applications on different platforms. Besides, it achieves great saving in power consumption. In the future, we will try to optimize the display performance by exploring more intelligent schemes. Also, we will analyze the strategies of the migration of multiple VMs to achieve more energy savings and better load balancing.

References

1. Baratto, R., Kim, L., Nieh, J.: THINC: A Virtual Display Architecture for Thin-Client Computing. In: Proceedings of 20th ACM Symposium on Operating Systems Principles, pp. 277–290. ACM Press, United Kingdom (2005)
2. Bailey, E.: Maximum RPM. SAMS, Indianapolis (1997)
3. Ponder, W.: Sun Desktop Virtualization Solution: Desktop Virtualization Blueprint. Sun Microsystems (2006)
4. Chandra, R., Zeldovich, N., Sapuntzakis, C., Lam, M.S.: The Collective: A Cache-Based System Management Architecture. In: Proceedings of the 2nd Symposium on Networked Systems Design and Implementation, pp. 259–272. ACM Press, Boston (2005)
5. eyeOS project, `http://www.eyeOS.org/`

6. Glide project, http://www.glidedigital.com/
7. Hopkins, M.: The On Site Energy Generation Option. The Data Center Journal, http://datacenterjournal.com/NewsArticle.asp?article_id=66 (2004)
8. Kim, J., Baratto, R.A., Nieh, J.: pTHINC: a thin-client architecture for mobile wireless web. In: Proceedings of the 15th International World Wide Web Conference, pp. 143–152. ACM Press, Edinburgh (2006)
9. Nieh, J., Yang, S.J., Novik, N.: Measuring thin-client performance using slow-motion benchmarking. ACM Transactions on Computer Systems 21(1), 87–115 (2003)
10. Orca desktop, http://www.orcaa.com/
11. Pinheiro, E., Bianchini, R., Carrera, E., Heath, T.: Dynamic Cluster Reconfiguration for Power and Performance. In: Benini, L., Kandemir, M., Ramanujam, J. (eds.) Compilers and Operating Systems for Low Power. Kluwer Academic Publishers, Norwell (2003)
12. Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual network computing. IEEE Internet Computing 2(1), 33–38 (1998)
13. Satyanaranyanan, M.: Pervasive Personal Computing in an Internet Suspend/Resume System. IEEE Internet Computing 11(2), 16–25 (2007)
14. VMware Virtual Desktop Infrastructure Partners, http://www.vmware.com/partners/alliances/solutions/
15. Yellowdog, http://linux.duke.edu/projects/yum/
16. Hu, L., Jin, H., Liao, X.: Magnet: A Novel Scheduling Policy for Power Reduction in Cluster with Virtual Machines. In: Proceedings of the 9th IEEE/ACM Conference on Cluster 2008. IEEE Press, Tokyo (2008)

CIM-Based Resource Information Management for Integrated Access Control Manager

Fumio Machida[1], Kumiko Tadano[1], Masahiro Kawato[1], Takayuki Ishikawa[2], Yoichiro Morita[3], and Masayuki Nakae[3]

[1] NEC Service Platforms Research Laboratories
[2] NEC Business Inovation Center
[3] NEC Common Platform Software Research Laboratories
1753, Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa, Japan
{h-machida@ab, k-tadano@bq, m-kawato@ap,
t-ishikawa@eb, y-morita@dc, m-nakae@bp}.jp.nec.com

Abstract. An architecture for CIM-based integrated access control middleware is proposed. The proposed architecture employs CIM standards for managing several access control modules for different resources uniformly in consolidated server environments. CIM standards allow the user of the middleware to look up the target resource information through the common interface, to describe abstract policy with grouping the same type of resources together, and to translate the abstract policy to detailed configurations for each access control module automatically. We evaluated the feasibility of the proposed architecture by a pilot implementation for file access control systems. According to the findings of the evaluation, we propose an extension of the CIM_Directory class to improve operations for exploring directories on user interfaces of the middleware.

Keywords: Access control, Common Information Model, Server consolidation.

1 Introduction

Security and reliability of systems in consolidated server environments using virtualization are major concerns. Many companies still hesitate to introduce virtual machines for server consolidation owing to concerns for vulnerability of virtualization software and spreading of security incidents or performance problems across the systems. To protect consolidated systems from security threats, system administrators conventionally install several security management tools and configure them in compliance with organizational security guidelines. Configuring such kind of security management tools properly is not an easy task, even though the systems do not include virtual machines.

The Secure Platform project (SPF) tackles this issue by developing security management middleware with secure components for operating systems and virtualization software. To achieve effective security management for consolidated server environments, systems need integrated management middleware for enforcing consistent security policies over the systems as well as secure components such as secure

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 37–48, 2008.

OS and secure hypervisor. Since the middleware aims to cooperate with some existing secure components, such as SELinux [1], as well, the architecture is designed with interoperability in mind.

This paper focuses on the architecture of an integrated access control manager (IAM) that can handle multiple access control modules in the same manner in accordance with the CIM standard [8]. Access control modules are used in various contexts, such as service access control, file access control, and network filters. The IAM provides a common interface for manipulating a security policy in user-friendly style and a function for converting the policy into specific configurations for individual access control modules. To handle the management information used in the individual access control module in the standardized way, the IAM adopts CIM standards as an abstract model for the target systems. CIM standards allow IAM to collect information on the access control target systematically, to make groups of resources that have the same properties, and to link the abstract information to specific information.

We illustrate the effectiveness of the CIM-based IAM architecture through a pilot implementation of the architecture focusing on the file access control. With the findings of the implementation, we propose an extension model of CIM_Directory with an additional property that indicates the list of files and directories contained in the directory. The extension enables effective directory search on a GUI for policy manipulation.

The rest of this paper is organized as follows. Section 2 describes the requirements for an integrated access control manager and the proposed architecture of IAM. Section 3 presents CIM-based resource information models for the IAM targeting file access control. Section 4 gives the details of our implementation, and Section 5 presents the results of experiments. Finally, the summary of this paper is outlined in Section 6.

2 Integrated Access Control

This section clarifies the requirements for the architecture of the integrated access control and describes the IAM architecture designed in the SPF project.

2.1 Requirements

In consolidated server environments, access control modules for security management are installed on many servers and virtual machines. They are distributed not only over servers but also over software layers. The access control module for an OS layer such as SELinux does not achieve sufficient security in a virtual machine because attacks from a privileged virtual machine or other virtual machines are not preventable. Administrators have to configure all access control modules distributed from the virtualization layer to the application layer by consistent policies. From the view point of ease of use for administrators, we summarize the requirements for the middleware that integrates management of access control as follows.

- **Management integration**
  Capability of managing multiple and several types of access control modules from a central console on the client workstation. This capability contributes to reducing the complex tasks required for configuring multiple access control modules consistently.
- **Policy abstraction**
  Introduction of an abstract policy that is applicable to different access control modules. The policy abstraction conduces to the easy manipulation of policies without the need for concern about specific styles and formats defined by individual access control modules. By using the abstract policy, dependencies and consistencies of policies can be checked easily.
- **Operation automation**
  Automation of operations such as lookup of target resource information and configuration of access control modules. The automation contributes to reducing the tasks of administrators. Operations such as resource information search and policy configuration are potentially performed automatically.

Besides the listed three requirements, administrators have several requirements, such as performance, strength of security, availability, and scalability. The above three requirements are especially important for improving manageability of access control modules in consolidated server environments.

2.2 Related Work

Linux Security Module (LSM) [2] provides a general framework for enhancing the security of Linux systems. Several security modules, such as SELinux and AppArmor [3], are based on this framework. Similarly, Xen Security Module (XSM) provides a framework for enhancing the security of virtualized environments using Xen. ACM [4] and Flask [5] are implemented by using this interface. Security administrators can protect against unauthorized access by configuring these security modules properly and consistently; however, this kind of configuration is a very complex task and a burden on administrators.

Although integrated access control systems for distributed systems have been studied by several researchers [6] [7], there is no work addressesing the architecture for integrated access control for different resources in consolidated server environments. Accordingly, we propose an architecture for an integrated access control system that assumes the use of a standardized resource information model to handle different resources.

2.3 Proposed Architecture

Figure 1 shows the architecture of the integrated access control manager (IAM) we designed in consideration of the above requirements. The IAM is comprised of Policy Manager, Resource Information Manager and ID Manager. The components of IAM interact with Agents run on the target server.

Policy Manager provides an interface for manipulating the abstract policy using user information from ID Manager and target resource information from Resource Information Manager. The abstract policy is interpreted into specific configurations

for individual access control modules with the help of a pluggable module containing interpretation rules.

Resource Information Manager provides a consistent view of the target resource information and associations between them based on the standardized information model. By modeling the property of the access control module, the Resource Information Manager enables Policy Manager to handle different types of access control in the same manner.

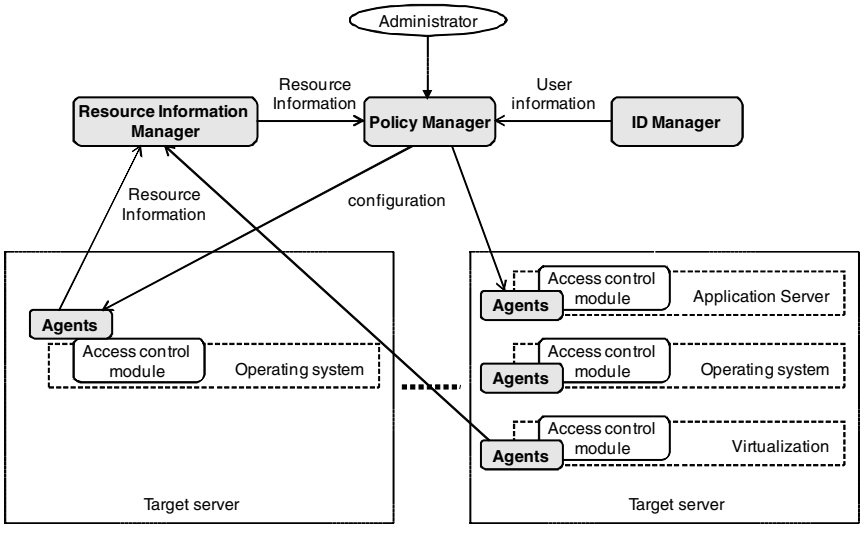


Fig. 1. Architecture overview of the integrated access control manager (IAM)

ID Manager is a directory service that manages users, groups, and organizational information. The user information is used to compose the abstract policy and may be translated to a system-specific ID or accounts at the interpretation.

Agents are separated into two functional components: Resource Information Agents and Policy Deployment Agents. Resource Information Agent interacts with Resource Information Manager to provide the target resource information. Policy Deployment Agent is called by Policy Manager to receive a specific configuration for the target access control module and configure it properly. The functionality of Resource Information Agent varies according to the type of target resources while the functionality of Policy Deployment Agent varies with the access control modules.

2.4 Interactions

Manipulation of abstract policies and deployment the policies to an individual access control module are achieved by the interactions between the components of the IAM.

In the policy manipulation process, Policy Manager queries ID Manager to get the user information that is used as a “subject” of an abstract policy. At the same time, Policy Manager collects target resource information from Resource Information

Manager. The target resource information is grouped together as resource group that is used as an “object” of the abstract policy. Administrators make policies by correlating the subject, the object and “action” that are given according to resource categories.

In the policy deployment process, Policy Manager queries Resource Information Manager to get the properties and capabilities of the target access control module. The obtained information is used to translate the abstract policy into the specific configurations for the target module. After generation of the specific configurations, Policy Manager sends them to Policy Deployment Agent, which configures the target module according to the received configuration data.

Note that the components and interactions mentioned above do not depend on the specific implementation of the access control module. If the information model and protocol are designed with standardized technologies, the IAM architecture is applicable to various access control systems.

3 Information Model for File Access Control

This section introduces the CIM model for expressing the management information used in a concrete use case of the proposed IAM architecture. The use case focuses on the integrated file access control system. Files and directories as the target resource of the IAM are modeled using the CIM model.

3.1 Scenario

As a pilot implementation of the IAM architecture, we designed the integrated access control middleware targeting to file access control on an operating system. We assumed use of an OS reference monitor that is a file access control module using access control list (ACL). The OS reference monitor is also developed in the SPF project.

The main goal of the pilot implementation is to manage distributed multiple OS reference monitors through IAM architecture. OS reference monitors are installed in each target server. To interface the OS reference monitors, Agents are installed on each target server. The set of IAM server components, namely, Policy Manager, Resource Information Manager, and ID Manager, are running on the administrator’s workstation.

In the context of the file access control, subject of the abstract policy is user account of each operating system while object is a group of files and directories. Action of the abstract policy is s set of file system operations that are limited by the file system type. Since we used Linux ext3 file system in the target server, the supported operations are "read", "write", and "execute". Administrator manipulates the abstract policy with specifying the subject, the object and the action.

In the deployment phase, the abstract policy is converted to ACLs for the OS reference monitor by Policy Manager, which calls Resource Information Manager to get the information of the target OS reference monitor: address, version, supported subject type and object type. The resource information model used in this scenario is shown in the following subsections.

3.2 File and Directory

We apply CIM to express the target resource information. The target resources of the OS reference monitor are files and directories. They are modeled in the CIM standard as depicted in Fig. 2. CIM_Directory inherits CIM_LogicalFile and logically represents a group of files contained in it. This model is suitable for the IAM because both files and directories have a potential to be the target of access control. Policy Manager can handle the target resource information as a instance of CIM_LogicalFile in the same way.

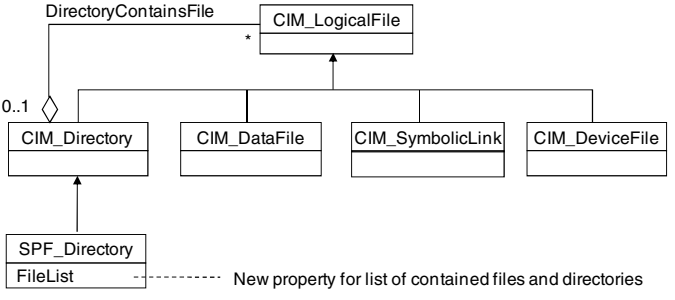


Fig. 2. CIM classes for directories and files

We propose an extension of the CIM_Diretory from an operational viewpoint. The current CIM_Directory has a component relationship to multiple CIM_LogicalFile (i.e. DirectoryContainsFile). To lookup the files and subdirectories contained in a certain directory, it is necessary to retrieve all CIM_LogicalFile instances that have component relationships with the CIM instance of this directory. The operation to get all related instances is too inefficient for only looking up the contents list of the directory. Accordingly, we added a new property, "FileList", which is an array of string describing the file name contained in the directory (see SPF_Directory in Fig. 2). This new property allows a user interface for directory browsing in order to show the contents of the directory simply by getting an instance of the target CIM_Directory.

3.3 Reference Monitor

For translation of policies to specific ACLs, the property information of the OS reference monitor is needed. Since the OS reference monitor is implemented as Linux Security Module (LSM), the model for the OS reference monitor is defined by extending CIM_SoftwareElement, as shown in Fig. 3.

SPF_ReferenceMonitor inherits the properties, such as Name and Version, of CIM_SoftwareElement. The types of subject and object supported by the OS reference monitor are expressed within the SPF_RMTagetSettingData extended from CIM_SettingData. SPF_RMTargetSettingData has two new properties: "SubjectType" and "ObjectType". The values of the new properties are strings that identify the types of subject and object. In our implementation, SubjectType is set to "OU=accounts",

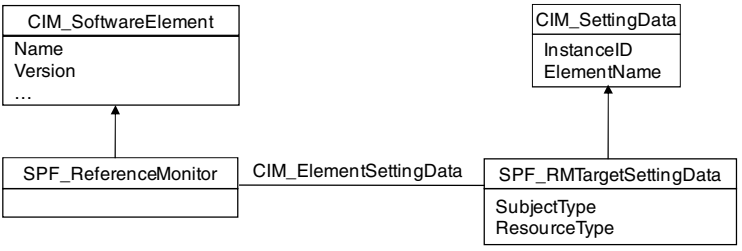


Fig. 3. CIM class definition of OS reference monitor extending CIM_SoftwareElement

which corresponds to the type information managed in ID Manager, and ObjectType is set to "CIM_LogicalFile", which represents files and directories.

3.4 File Access Capabilities

In the policy manipulation step, Policy Manager has to handle the set of actions for abstract policy. For the file access operations supported by the ext3 file system, the action corresponds to the set of operations: "read", "write", and "execute". We used CIM_Capabilities to express the set of actions for abstract policy. Policy Manager selects an operation as action of the abstract policy from the instance of CIM_Capabilities. Capabilities for file systems are modeled as SPF_FilesystemCapabilities that has three boolean properties, namely, "ReadSupported", "WriteSupported", and "ExecuteSupported" (See Fig. 4).

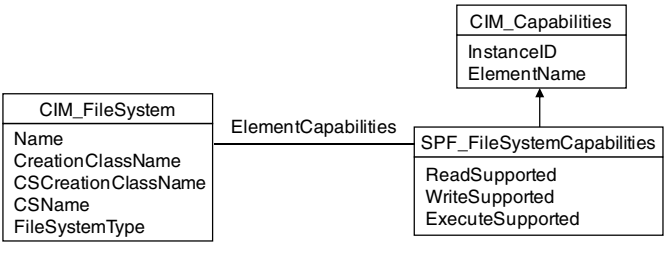


Fig. 4. CIM class definition for file access capabilities

4 Implementation

4.1 Overview

An overview of the IAM implementation is shown in Fig. 5. All components of IAM are implemented by Java except for the scripts that are executed in Agents, which are called from Policy Manager and Resource Information Manager through web service interfaces. Policy Manager has an XML database to store the abstract policies formatted with XACML [9], while Resource Information Manager has a mechanism for a

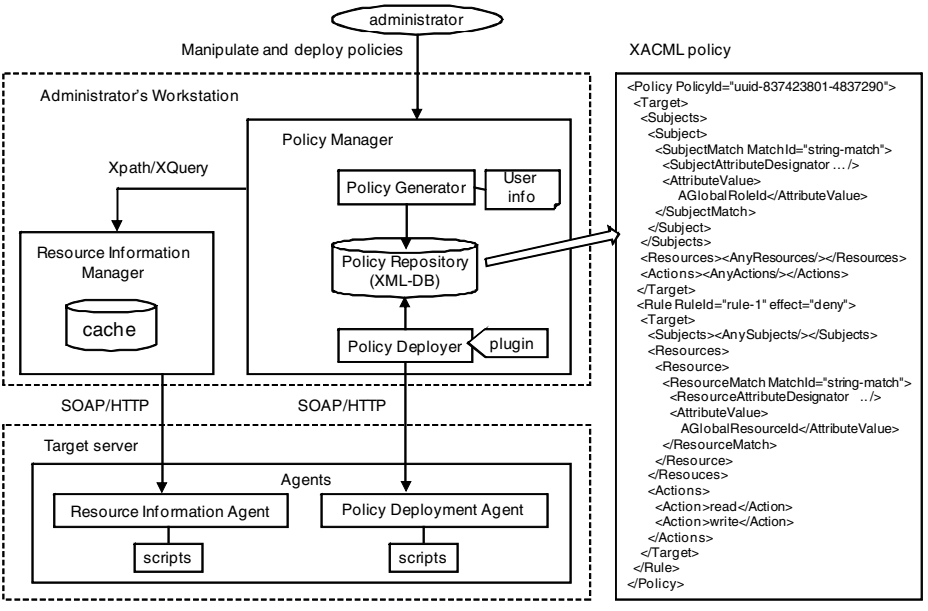


Fig. 5. Overview of the IAM implementation

Java instance cache. CIM-XML format is used for an expression of CIM-based resource information. Policy Manager queries resource information by using XPath/XQuery. Since the current implementation does not include ID Manager, the user information is configured in Policy Manager.

4.2 Agents

Agents are implemented as web services on Apache Axis containers. When Resource Information Agent receives a request to collect resource information, the Agent runs scripts corresponding to requested CIM classes to get information and generate CIM-XML formatted resource information. On the other hand, Policy Deployment Agent executes the scripts to overwrite the ACLs when Policy Manager sends the new ACLs.

4.3 Resource Information Manager

Resource Information Manager supports two types of resource information collection: synchronous and asynchronous. Synchronous collection gets resource information on demand according to the query from Policy Manager. Most volatile resource information, such as an instance of file information, is collected by synchronous. Asynchronous collection gets resource information automatically by collection schedule and stores it in the cache. And nonvolatile resource information, such as an instance of a computer system, is collected by asynchronous collection.

Resource Information Manager provides an interface to query resource information by XPath/XQuery. The implementation of Resource Information Manager is

extensible to handle the non-CIM-based resource information because the implementation accepts schema-independent queries.

4.4 Policy Manager

Policy Manager consists of Policy Generator, Policy Deployer, and Policy Repository. Policy Generator assists administrators in describing role-based policies with browsing resource information and user information, and it stores the descriptions in the form of XACML in Policy Repository. Policy Deployer translates the XACML policy descriptions to ACLs automatically and sends them to Policy Deployment Agent. Policy Deployer takes a plug-in architecture to support access control modules having different capabilities and ACL formats. Each plug-in has policy translation rules for a corresponding access control module. An appropriate plug-in is picked up in accordance with the information of the access control modules modeled in Section 3.3.

5 Experimental Evaluation

This section demonstrates the user interfaces of the IAM implementation and experimental results of the performance evaluation in our test bed.

5.1 Policy Manipulation on GUI

The GUI for the IAM has a Resource Group Editor, shown in Fig. 6, to make groups of files and directories that are target resources of access control. Administrator creates a new group on the left frame, which shows the list of defined resource groups, and then chooses the files and directories to be grouped by exploring the tree on the right frame. Owing to the additional property of SPF_Directory, Resource Group Editor is able to reduce lots of queries for showing the contents list in a directory.

With these resource groups, instead of individual resource IDs, administrators can easily describe abstract policies in Abstract Policy Editor (Fig. 7).

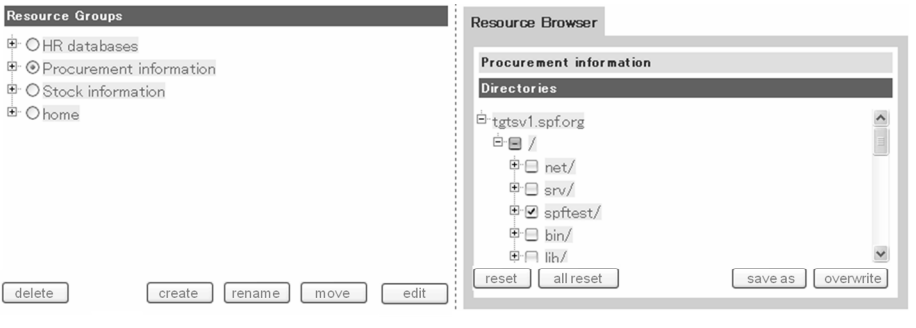


Fig. 6. Making resource groups on the Resource Group Editor

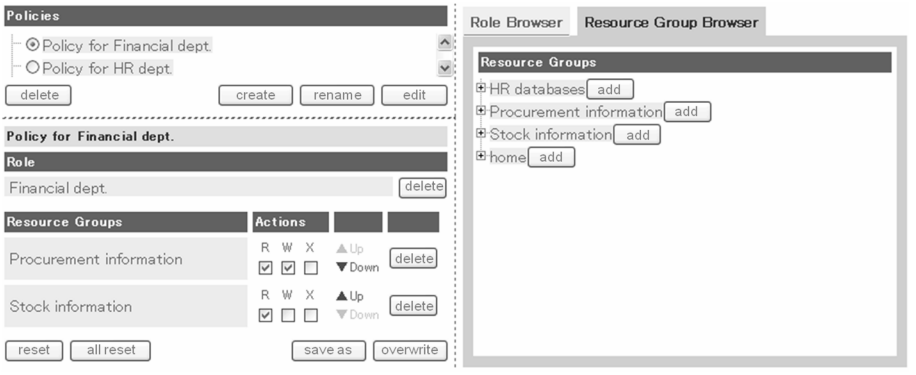


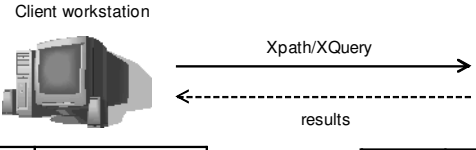
Fig. 7. Describing abstract policies on Abstract Policy Editor

Created policies can be deployed to the target servers by modifying a deployment schedule or by button click on demand. When the policy deployment process starts by button click or by the arranged schedule, the abstract policy is translated to ACLs automatically according to the translation rules stored in the plug-in of Policy Manager, and the ACLs are distributed to the target servers through communications with Policy Deployment Agent.

5.2 Query Performance

Query response time is an important factor in the usability of the IAM. The process of policy manipulation, especially the operations on Resource Group Editor, invokes lots of resource information queries. The efficiency of policy generation as well as the usability of the UI depends on the performance of resource information query.

In light of these facts, we evaluated the performance of Resource Information Manager by measuring the response time of each XPath/XQuery which is used in the policy manipulation scenario. Figure 8 summarizes the testing configuration, and Table 1 lists the queries invoked in the scenario and corresponding results.



	Client
CPU	Pentium4 3.2GHz
Memory	1 GB
OS	Fedora Core 6

	Virtual Machine	Host Machine
CPU	Intel Xeon 3GHz	Intel Xeon 3GHz
Memory	512 MB	2 GB
OS	Cent OS 5	Windows Server 2003 VMware Player

Fig. 8. Configuration of the test bed for measurements of response time

Table 1. Response time of each query for Resource Information Manager

Query target	XQuery	Response time (sec)
Instances of computer systems	for $instance in //INSTANCE[@CLASSNAME="CIM_ComputerSystem"] return {$instance}	2.493
An instance of root directory	for $instance in //INSTANCE[@CLASSNAME="SPF_Directory"] let $value := $instance/PROPERTY[@NAME="Name"]/VALUE where $value = "/" return {$instance}	2.435
All instances just below root directory	for $instance in //INSTANCE[@CLASSNAME="SPF_Directory"] let $value := $instance/PROPERTY[@NAME="Name"]/VALUE where $value = "/etc" or $value = "/bin" or $value = "/lib" or $value = "/OPT" or $value = "/dev" or …	5.770
An instance of file access capabilities	for $instanceFSC in //INSTANCE[@CLASSNAME="SPF_FileSystemCapabilities"] ….	2.523
An instance of setting data for reference monitor	for $instanceRMTSD in //INSTANCE[@CLASSNAME="SPF_RMTargetSettingData"] ….	2.532

Most of queries take only 2.5 seconds to get the results, except for the query to get all directory instances just below the root directory. Unless the “FileList” of the SPF_Directory is used, it is necessary to invoke this inefficient query to show the list of the contents under the root directory on the Resource Group Editor instead of getting only an instance of root directory. As a result, we conclude that the "FileList" property of SPF_Directory contributes to improving the performance of each directory exploration more than twice.

6 Conclusion

A suitable IAM architecture for an integrated access control management system in consolidated server environments is proposed. The proposed architecture employs CIM standards for managing various types of access control modules. In a pilot implementation for integrated file access control, we apply CIM to model the file and directory information, reference monitor, and capabilities of file system. To explore the directories without retrieving redundant CIM instances, we propose an extension of the current CIM_Directory class definition. As a result of this extension, the efficiency of lookup for the contents of each directory significantly improves.

The future work includes further integration with access control modules for virtual machines, different OS and application servers. The IAM architecture has a potential to adapt some different access control modules by adding the agents and the plug-ins. Moreover, scalability of the IAM architecture is another important issue. We improve the IAM implementation to manage lots of servers more efficiently.

Acknowledgments. This work is a part of the Secure Platform project supported by Japanese ministry of Economy, Trade and Industry, and Association for Super-Advanced Electronics Technologies. The IAM architecture and the information retrieval interfaces for access control modules were developed with much help of T. Hatakeyama, M. Yuhara, T. Aizawa, and T. Kurita of Fujitsu Limited.

References

1. Loscocco, P., Smalley, S.: Integrating Flexible Support for Security Policies into the Linux Operating System. In: Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference, pp. 29–42 (2001)
2. Wright, C., Cowan, C., Morris, J., Smalley, S., Hartman, G.K.: Linux Security Modules: General Security Support for the Linux Kernel. In: Proceedings of the 11th USENIX Security Symposium, pp. 17–31 (2002)
3. AppArmor, `http://www.novell.com/linux/security/apparmor/overview.html`
4. Sailer, R., Jaeger, T., Valdez, E., Caceres, R., Perez, R., Berger, S., Griffin, J.L., Doorn, L.: Building a MAC-Based Security Architecture for the Xen Open-Source Hypervisor. In: Proceedings of the 21st Annual Computer Security Applications Conference (CSAC), pp. 276–285 (2005)
5. Spencer, R., Loscocco, P., Smalley, S., Hibler, M., Anderson, D., Lepreau, J.: The Flask Security Architecture: System support for diverse security policies. In: Proceedings of The Eighth USENIX Security Symposium, pp. 123–139 (1999)
6. Jing, J., Gail-Joon, A.: Towards Secure Information Sharing and Management in Grid Environments. In: Proceedings of Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2006), pp. 1–7 (2006)
7. Ryutov, T., Neuman, C.: Representation and Evaluation of Security Policies for Distributed System Services. In: Proceedings of DARPA Information Survivability Conference and Exposition, pp. 172–183 (2000)
8. Common Information Model (CIM) Standards, `http://www.dmtf.org/standards/cim/`
9. eXtensible Access Control Markup Language (XACML), `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`

Scenario-Based Distributed Virtualization Management Architecture for Multi-host Environments

Fermín Galán[1], David Fernández[2], Miguel Ferrer[2], and Francisco J. Martín[2]

[1] Telefónica Investigación y Desarrollo (TID)
[2] Departamento de Ingeniería de Sistemas Telemáticos (DIT),
Universidad Politécnica de Madrid (UPM)
fermin@tid.es, {david,mferrer,fjmartin}@dit.upm.es

Abstract. Traditional virtual scenario management tools (VNUML, NetKit, MLN, etc.) normally consider mono-host deployment (i.e. the entire virtualized scenario deployed in the same physical host). In this paper, the work carried out in the EDIV project, dealing with the development of a multi-host evolution of the Virtual Network User Mode Linux (VNUML) tool, is presented. Following an overview of virtual scenario-based management, the distributed deployment management architecture, detailing the main components, its interfaces and operations, is described. Additionally, details on the actual implementation and the results achieved so far are provided, assessing the feasibility and advantages of the system. Finally a proposal to evolve the tool towards DMTF virtualization related standards (CIM-based virtualization management and OVF) is outlined.

Keywords: Distributed management, virtualization, VNUML, CIM management, OVF, User Mode Linux.

1 Introduction

Scenario-based virtualization management tools (e.g. VNUML [1], NetKit [2] or MLN [3]) are essential in research and education. In such environments the ability to provide experimentation scenarios involving arbitrary and complex topologies composed of virtual machines in a quick and inexpensive way is dramatically useful. These tools are quite different from "conventional" virtualization management infrastructure tools (e.g., VirtualCenter [4]), which are oriented to production data center environments and more focused on achieving optimal performance for final users than in topological flexibility during development and testing stages.

In this context, a *scenario* can be defined as the formal specification (e.g. using eXtended Markup Language –XML) of a set of virtual machines along with their interconnection within a given topology. Furthermore a *scenario-based virtualization management* tool is one aimed at processing scenario specifications in order to be deployed in a physical infrastructure and provide further management. Traditional tools (such as the ones cited in the previous paragraph) normally consider mono-host deployment alone (i.e. the entire scenario deployed in the same physical host).

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 49–60, 2008.

Although this is the straightforward approach, it imposes severe scalability problems, making it very difficult to build large scenarios composed of tens or hundreds of virtual machines. In this case, the only possibility is to split the large scenario in smaller pieces, breaking the management integrity (because of each sub-scenario having to be manually managed independently) and thus increasing complexity.

In order to avoid this limitation, the EDIV (Spanish acronym for *Distributed Scenarios with VNUML*) project, a partnership between Telefónica I+D and UPM starting in January 2008 and based upon our previous work [5], has developed an evolution of VNUML to allow the management of virtualization scenarios deployed on multi-host distributed infrastructure. One of the main design requirements is transparency, so users accustomed to use basic VNUML (i.e. mono-host) will be able to work with EDIV in minutes.

This paper provides a thorough description of the EDIV architecture and its scenario management operations, as well as practical details of a fully functional prototype implementation and its first experimental results: several reference scenarios have been tested over a simple cluster made of three hosts, allowing to analyze how they are split depending on the segmentation algorithm used, and showing how EDIV improves the basic mono-host deployment model of VNUML. Additionally, considering that aligning EDIV-based distributed management with recent virtualization-related standards from DMTF (Distributed Management Task Force) will be a very interesting approach to improve its interoperability, a possible evolution towards CIM (Common Information Model)-based virtualization management and Open Virtualization Format (OVF) is also analyzed in the paper.

The remaining sections of the paper are structured as follows. Firstly Section 2 describes the EDIV architecture, detailing its main components and the interfaces among them, paying special attention to VNUML tool installed in deployment hosts. Section 3 describes the different management operations. Section 4 provides the implementation details and Section 5 includes the subsequent experimental results. Finally, Section 6 presents the proposal to align EDIV with DMTF standards and Section 7 closes the paper with the main conclusions and future working lines.

2 Architectural Design

This section describes the architecture in which the EDIV multi-host deployment system is based. The physical layout is shown in Fig. 1 and consists of a cluster of *N* physical deployment hosts plus a deployment controller. The latter is a logical entity that can be collocated in one of the deployment hosts or located in a physically separated element, as it is shown in the figure. All elements are connected to a common programmable switch-based backplane through *trunk* physical interfaces able to multiplex 802.1q VLANs [6].

In the following subsections the different elements in the architecture (deployment hosts, deployment controller and interconnection backplane) are described. Later, in Section 5, the example configuration made of three hosts and one controller that has been used to develop and validate the prototype is presented.

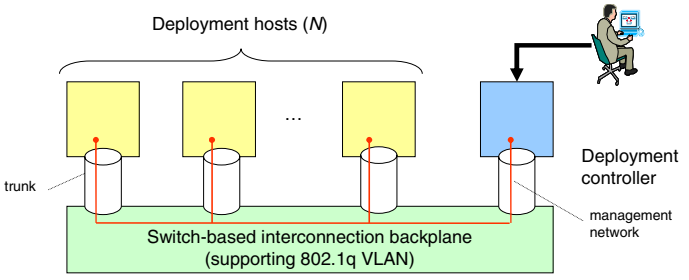


Fig. 1. EDIV physical layout

2.1 Deployment Hosts

Deployment hosts are GNU/Linux systems in which virtual machines belonging to deployed scenarios are run. They have the VNUML tool installed, along with the additional components required for its operation (UML kernels and root filesystems for virtual machines).

VNUML [1] is a general purpose open source tool designed to help building virtual network scenarios that are automatically deployed in a single host. It is based on User Mode Linux (UML) [7], a virtualization technique to run virtual Linux machines over a Linux host. The tool is made of two main components: the XML based VNUML language that enables to describe the virtual network scenario; and the interpreter of the language (actually a shell command) that parsers the description and builds and manages the scenario, relieving the user of all complex UML details.

The tool implements three scenario management modes, which are applied on scenario specifications (either written by users themselves in conventional VNUML utilization or provided through the H interface described in Section 2.2): *deploy*, to start the virtual network scenario in the host; *execution*, to execute programmed command sequences on scenario virtual machines and copying files to them; and *undeploy*, to dismantle the scenario, releasing the host resources.

Regarding VNUML specifications, they are composed of two main sections: virtual networks and virtual machines (a complete reference of the VNUML language can be found in [1]). The virtual networks that interconnect virtual machines among them or with the host network interfaces are defined using the <*net*> tag. Virtual networks are implemented by means of virtual software bridges running on the host machine. The <*vm*> tag is later used to describe each virtual machine. The language enables to describe their main characteristics in terms of: the kernel and the filesystem that are run (<*kernel*> and <*filesystem*> tags), the network interfaces used (<*if*> tag) and the addresses associated with them, either IPv4 or IPv6 (<*ipv4*> and <*ipv6*> tags), as well as the IP routes (<*route*> tag). Besides virtual machine specifications can include <*filetree*> and <*exec*> tags to define files to be copied from the host to the virtual machines and user defined commands to be executed on them.

Additionally to VNUML tool, some other software packages are also required in the deployment hosts (e.g., `vconfig`, to create VLAN specific sub-interfaces in the physical trunk). Finally note that in order to implement the H interface some specific

services or process could be needed (in particular, a SSH/SCP based interface requires a SSH daemon).

2.2 Deployment Controller

The deployment controller manages the distributed virtualization scenario, providing the interface for the user and implementing the operations to deploy, execute commands and undeploy (described in Section 3). It is implemented in a modular way (Fig. 2), composed by several functional entities (coordinator, segmentator and database), described below along with their inner and external interfaces.

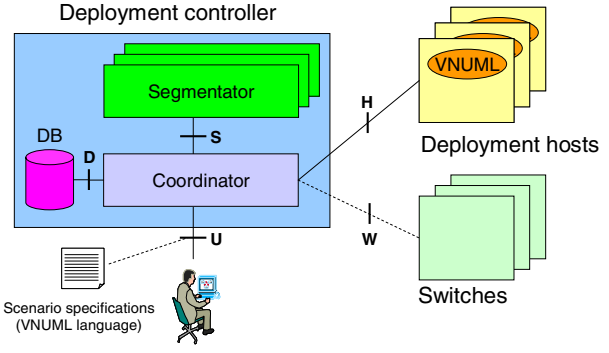


Fig. 2. Deployment controller architecture and logical interfaces

Coordinator. It is the main module of EDIV. It implements the user interface (U) and coordinates the rest of the modules in order to perform the operation requested by the user. As already stated in Section 1, one of the main design requirements for EDIV is transparency for VNUML users. This means that the U interface must be as close as possible to conventional (i.e. mono-host) VNUML utilization. Thus the invocation of the deployment controller is performed through a shell command, using the same XML scenario specification language and management operations (deploy, undeploy and execute) than VNUML does (see Section 2.1).

The coordinator also interacts with external hosts and switches (using the H and W interfaces, respectively). Both interfaces are based on remote command and file copying protocols (e.g. SSH, SCP and rsync are the usual ones for GNU/Linux based hosts, while switch vendors typically use SNMP, telnet or TFTP), over a dedicated network (named *management network)*, which can be associated to a preconfigured VLAN (as shown in Fig. 1) or being a physically separated network (as used in the experimental setup described in Section 5). It is assumed that IP connectivity in the management network is preconfigured in all hosts and deployment controller before starting operation.

Segmentator. This module performs the segmentation of the virtual scenario, deciding in which host each virtual machine will be deployed. Using the S interface, the coordinator provides the VNUML specification to the segmentator, which processes it

according to the segmentation algorithm selected (opaque from the point of view of the coordinator module), returning a table in which each <*vm*> in the specification is assigned to one of the hosts.

The main advantage of the deployment controller modular design is that it enables to easily add new modules implementing new segmentation algorithms, as long as they meet the S interface (consisting in just one primitive, SPLIT, as detailed in Section 3). Moreover several segmentator modules could be available at the same time, allowing the user to choose the most convenient among them when invoking the deployment controller (e.g. with an algorithm selector argument in the U interface command line).

Database. It stores persistent information between controller executions. In particular, it stores the per-host sub-specification fragments running in each host (associated to the scenario they belong to), the list of available VLAN IDs to implement inter-host virtual networks and arbitrary parameters that segmentation algorithms might need (e.g. host statistical load information).

Fig. 2 shows the database as an inner component to the deployment controller, although actually it could be set outside (e.g. in order to take advantage of centralized backup policies). Regarding the D interface, it depends on the database implementation (e.g. SQL for a SQL-based database).

2.3 Switches

The programmable switch backplane provides physical interconnection among deployment hosts and controller. It is composed of a set of Ethernet switches but, as long as it provides end-to-end 801.2q trunks among each deployment host and controller, the internal switch backplane interconnection topology is not relevant (even just one switch could be used if the number of hosts is low).

If the VLANs used by EDIV to interconnect virtual machines running on different hosts need to be explicitly managed in the switches by the deployment controller, the W interface (between coordinator and switches) has to be implemented, so the coordinator can enable (or disable) specific VLAN IDs in the switches. Alternatively it can be assumed that all VLAN IDs in the operational range of the controller are always enabled in the switches. In this case, the W interface is unnecessary (this is the simplest approach, but it might not be desirable due to security concerns).

3 Detailed Operation Sequences

This section details the two basic EDIV scenario management operations: deploy and undeploy, considering a simple setup made of two deployment hosts. The third operation (execution) is not shown for the sake of briefness, although it is very close to undeploy.

The deployment of the scenario (Fig. 3) starts with a user command through the U interface, specifying the desired scenario (*SC*) as argument. As a preparatory step, the coordinator accesses the database in order to get pre-deployment information (e.g. algorithms parameters, list of free VLANs, etc.). Then *SC* specification is provided to

the segmentator in the SPLIT primitive. The segmentator returns a *mapping*, which basically is a table in which each virtual machine in SC is assigned to one of the deployment hosts (in this case, either H1 or H2).

Next the coordinator splits the *SC* XML into per-host fragments (*SC1* and *SC2*), each *one* containing the <*vm*> tags of the virtual machines assigned to each host (H1 and H2) and the <*net*> tags corresponding to intra-host and inter-host virtual networks (inter-host virtual networks required a slight modification in <*net*> tags, see [5] for details).

In the next step the H interface is used to create inter-host networks. Basically, for each network a free VLAN ID is chosen, creating the needed subinterfaces in each host (by issuing the CREATE_VLAN_SUBIF primitive which maps to `vconfig` command), and, optionally, enabling the VLAN in the switches through the W interface (issuing an ENABLE_VLAN primitive). Later a VNUML deploy operation on each host is invoked using the corresponding scenario fragment (*SC1* and *SC2*).

The coordinator issues operations in H and S interfaces in parallel, waiting for all operations to finish. Once finished, it performs some post-deployment operations (mainly storing *SC1* and *SC2* and the newly used VLANs IDs in the database, associating them with the name of the distributed scenario as key) and finishes reporting the outcome of the deployment operation ("OK" in case of success) to the user.

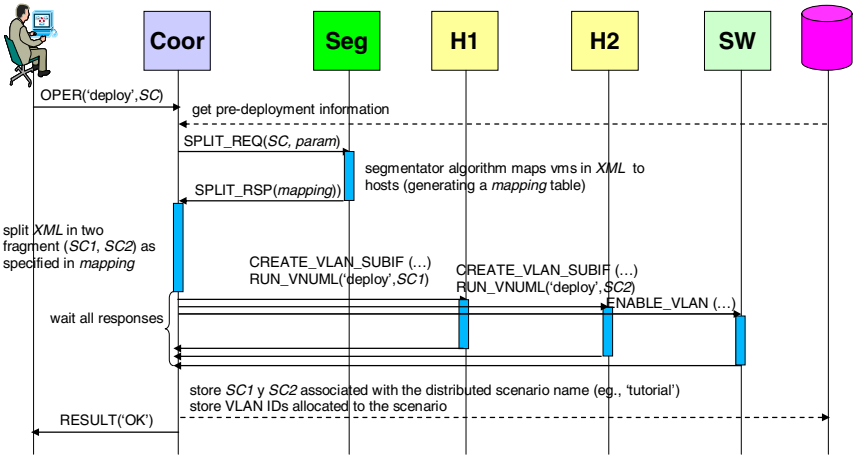


Fig. 3. Deployment operation sequence

Regarding undeployment (Fig. 4), the user only needs to provide the name of the scenario, because the fragments of the XML specifications (*SC1* and *SC2*) are stored in the database, as described above, at deployment time. In addition, the coordinator also obtains the list of VLAN IDs used by inter-host networks in the scenario. With all this information gathered, the coordinator:

- executes a VNUML undeploy operation on each host through the H interface
- releases inter-host VLAN IDs (DESTROY_VLAN_SUBIF) and, if needed, disables the VLANs in the switches through the W interface

Once all the previous parallel operations have finished, it cleans the database information associated with the undeployed scenario (i.e. removing SC1, SC2 and the inter-host VLAN ID list) and reports the final result to the user.

It is important to note that, as the undeployment (and execution) operation does not require providing the original scenario XML filename, the risk of incoherence (e.g. trying to undeploy an scenario using a different specification than the one used to deploy it) is eliminated, improving in this way the original VNUML behaviour.

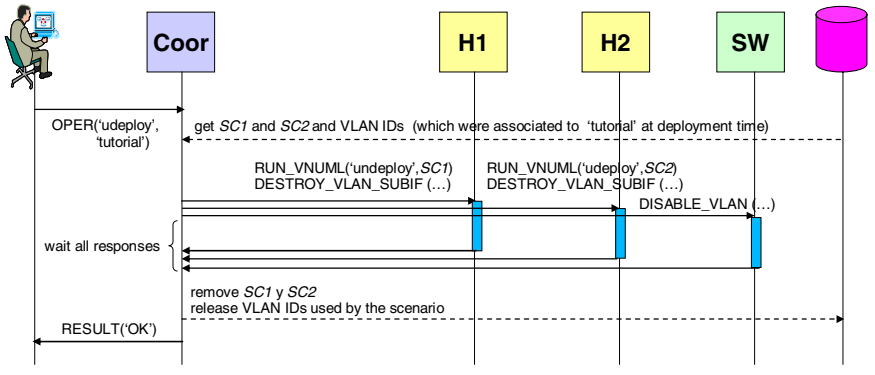


Fig. 4. Undeployment operation sequence

4 Implementation

A fully functional prototype of the deployment controller (implementing all the modules and interfaces described in Section 2.2) has been developed; details are provided in this section.

Both the coordinator and segmentator modules are implemented in Perl (as the original VNUML is), being the S interface based on the Perl module API (i.e. the coordinator acts as the main program and each segmentator algorithm is encapsulated in a corresponding .pm file). Two different segmentation modules have been implemented, each one involving a different algorithm: round robin (sequentially allocates each virtual machine to one of the deployment hosts) and weighted round robin (a variation of the previous, in which the host CPU load measures are used to weight the allocation, so more loaded deployment hosts would get lower share of virtual machines). Segmentation algorithms can be complemented using explicit allocation rules. This feature allows users to provide indications on how the deployment should be performed (e.g. "virtual machines A and B must be deployed in the same host"), acting as restrictions to the segmentation process.

The deployment controller uses a configuration file with the following information:

- deployment hosts cluster characteristics (e.g. number of hosts, CPU, memory, management network IP address, etc.),
- range of VLAN IDs available for inter-host virtual networks in the interconnection switches
- default segmentation algorithm

Regarding the database, a MySQL database implementation has been selected, so the D interface is based on the standard modules used in Perl to access MySQL databases.

Regarding external interfaces, the H interface is based on SCP (to copy the VNUML sub-specifications) and SSH (to issue commands). The optional W interface has not been implemented, assuming that the range of VLAN IDs specified in the configuration file is enabled in the interconnecting switched.

Finally it is worth mentioning that the deployment controller not only implements the management operations described in Section 3, but also provides centralized access to the consoles of all the virtual machines in each scenario, by using SSH port redirection mechanisms. In this way, the user can access text-based or xterm-based virtual machine consoles in the same way he is accustomed to use with basic VNUML.

5 Results

In order to validate the EDIV deployment controller implementation described in the previous section and to show that the behavior of the segmentation algorithms is as expected, several experiments have been performed.

The experimental setup (shown in Fig. 5) is composed of three deployment hosts (H1, H2 and H3), consisting on Xeon 3.4 GHz dual core 4GB RAM machines, interconnected through a single switch supporting VLANs. The management network is physically independent, using different physical interfaces. Both deployment hosts and deployment controller run in Ubuntu 7.10 GNU/Linux operating system.

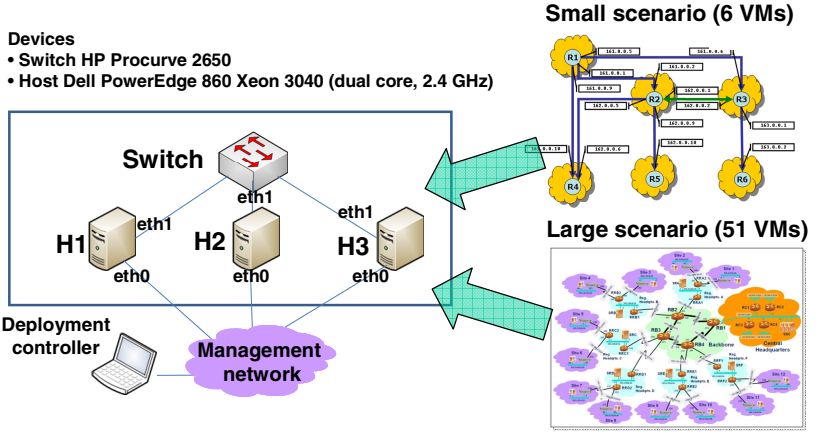


Fig. 5. Experimental setup

In order to perform the tests, two complementary scenarios (small and large) have been used, both taken from the VNUML examples library at [1]. The small scenario is a BGP routing testbed used to test routing daemons (specifically bgpd) and it is composed of 6 virtual machines. The large scenario is a networking laboratory for

OSPF dynamic routing tests, used in grade courses at DIT UPM, resembling the topology of a distributed organization and involving 51 virtual machines.

The deployment of both scenarios has been performed using round robin and weighted round robin algorithms, considering several host load distribution situations (the results are shown in Table 1). In the table, *low* (L) implies CPU is free almost all the time (around 0% use); *medium* (M), around 50%; and *high* (H), used almost all the time (around 100%). In order to set the proper load value, hosts have been stressed using `cpuburn` tool [8].

Table 1. Experiments results

	Deployment host loads			**Round-Robin**			**Weighted Round Robin**		
	L1	**L2**	**L3**	**H1**	**H2**	**H3**	**H1**	**H2**	**H3**
Small scenario (6 VMs)	L	L	L	2	2	2	2	2	2
	M	M	M	2	2	2	2	2	2
	H	L	L	2	2	2	1	3	2
	H	M	M	2	2	2	2	2	2
Large scenario (51 VMs)	L	L	L	17	17	17	17	17	17
	M	M	M	17	17	17	17	17	17
	H	L	L	17	17	17	7	22	22
	H	M	M	17	17	17	13	19	19

It can be demonstrated that the round robin algorithm always performs a uniform distribution, so each host gets the same number of virtual machines (2 in the case of the small scenario, 17 for the large one). In the case of weighted round robin a uniform distribution is also obtained when the relative load value is the same in all hosts (i.e. L/L/L and M/M/M). However, with dissimilar loads, the more loaded host gets the lowest number of virtual machines, and the difference between the number allocated to the most loaded and the less loaded is proportional to the load difference. Therefore, in the large scenario, the most loaded hosts (H1) get 7 virtual machines in H/L/L case, while obtaining 13 in the H/M/M (because in the latter the load difference is smaller). It is worth noting that, for the small scenario, this effect is almost negligible (only in the H/L/L case), given it is composed only of 6 virtual machines.

Finally the deployment time (i.e. the time elapsed since the user invokes EDIV deployment controller until the scenario is finally deployed and ready for use) has been analyzed. Using conventional VNUML mono-hosts deployment, the small scenario takes around 42 seconds to boot, while the larger one takes around 282. However, using weighted round robin, this time can be lowered to 17 and 130s respectively (in the L/L/L, M/M/M and H/L/L cases), thus achieving a 50-60% significant saving. In the H/M/M the time still being short in the large scenario case (135s), while it rises to 43s in the small scenario. This can be explained because in the large scenario the most loaded host is running a significant smaller number of virtual machines than in the other hosts (so deployment time is almost the same), while for the small scenario the most loaded host runs the same number of machines as the others, which implies a higher deployment time, close to the one in VNUML mono-host case.

In summary, the tests carried out have proved that both algorithms behave as expected, being the weighted algorithm more adequate when virtual machine deployment has to compensate the CPU load in each host.

6 EDIV Alignment with DMTF Standards

Although the EDIV implementation described so far provides a sound approach to scenario-based distributed virtualization management (as proven by the results shown in Section 5), it could be enhanced by aligning it with some DMTF standards. In this line, possible modifications to the current architecture and the expected advantages are described in this section.

One of the current EDIV limitations is that only UML-based virtual machines can be deployed, due to the use of VNUML as the tool in charge of per-host local deployments. Given that DMTF is developing a set of standards in order to provide an interoperable management of virtualized infrastructure based on CIM, one possible EDIV improvement could be to consider CIM-based deployment hosts in addition to VNUML-based hosts (as shown in Fig. 6). Instead of using SSH/SCP in the H interface (which is quite opaque from a management point of view) the interface would be based on WBEM standard protocols, particularly CIM-XML or CIM-SOAP [9]. In addition, a WBEM server would be run in these deployment hosts, including in the repository the CIM virtualization model described in the CIM virtualization profiles [10]. A provider able to manage CIM-based virtual machines would be required, such as libvirt-cim [11], which could be integrated in any standard CIMOM (due to it implements the Common Manageability Provider Interface –CMPI) and would allow managing virtual machines based on other technologies different from UML, such as KVM [12], Xen [13] or OpenVZ [14]. Note that this enhancement could imply small modifications in the VNUML language, mainly in the <*vm*> tags, using a new *type* attribute in order to specify the virtualization technology (e.g., <*vm type="xen"*>).

Another DMTF standard that would be interesting to consider from the EDIV point of view is OVF [15], which provides a common hypervisor-neutral open virtual machine format, fostering the adoption of virtual appliances (i.e. pre-configured software stacks comprising one or more virtual machines to provide self-contained services) as new software release and management model (e.g. through the development of virtual appliance lifecycle management tools). Using OVF-based virtual machines in EDIV scenarios could provide two benefits. Firstly it would enable to design scenarios in which the virtual machines are actually virtual appliances highly optimized for specific purposes (e.g. dynamic routing stacks, firewalls, etc.) provided by third-parties, relieving users from much of the installation and configuration effort. Secondly the use of the same packaging format in EDIV scenarios than in pre-production and production environments would reduce the evolution gap through the development lifecycle. Thus, once OVF virtual machines have been tested and tuned in the EDIV scenario, packages can be smoothly migrated to pre-production (in order to finish the tests in an environment that mirrors the production layout) and finally production, dramatically reducing the time-to-market of new architectures and services.

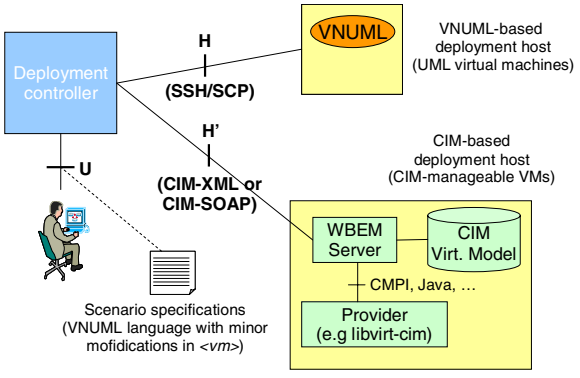


Fig. 6. CIM-based deployment hosts in the EDIV architecture

The EDIV evolution towards OVF consists on including OVF-encoded virtual machines in EDIV scenario specifications and is aligned with the CIM-based deployment hosts enhancement described above. Thus a new type of virtual machines would be considered (e.g. <*vm type="ovf"*>) including a reference to the virtual machine in the OVF file and a CIM-based provider able to manage OVF packages. In this way, the power of OVF virtual appliances would be integrated in EDIV scenarios without involving severe modifications in the deployment controller logic. Moreover, in case of reusing multi-VM OVF appliances involving topology descriptions (it is worth noting that OVF descriptors allow to define arbitrary topologies equivalent to VNUML ones, although in a less intuitive way), a translator from OVF descriptors to VNUML language (both XML-based) could be developed.

Note that the aforementioned modifications would extend the EDIV management capabilities (from UML to any virtualization technology supporting CIM management, with a special interest in OVF virtual appliances), but keeping the two main EDIV assets: the VNUML language (with small modifications) as high level descriptive scenario specification language and the segmentation modules (where the virtual machine allocation logic is encapsulated). Moreover these improvements could co-exist with the currently existing VNUML-based deployment hosts, considering two variants of the H interface (as shown in Fig. 6).

7 Conclusions and Future Work

The present paper has described EDIV: an scenario-based distributed virtualization management architecture based on a deployment controller. Apart from the detailed description of the architecture, its modules, interfaces and operation sequences, the paper provides several results that assess the feasibility of the approach and its advantages compared to the mono-host traditional deployment.

As main conclusions, firstly the EDIV distributed management approach allows to deploy very large and complex scenarios (e.g. hundreds of virtual machines), which would be very difficult or impossible to build in a single host due to the resources it would require. Secondly deployment times can be dramatically reduced (around 50-60% saving), as shown in Section 5.

Another advantage of EDIV is its transparency for the user: the deployment controller offers a user interface very similar to the one offered by VNUML, allowing a VNUML user to switch to EDIV in minutes. Indeed users only have to worry about the high-level scenario design, because the low-level deployment details are hidden by both EDIV and VNUML.

Another important feature of EDIV, derived from its modularity, is the clear separation between the deployment controller logic and the segmentation algorithm, thanks to the S interface. This fact allows segmentator developers to focus on providing highly optimized algorithms without any concern about the controller deployment logic which uses such algorithms, and, on the contrary, to optimize and extend the deployment logic (e.g. new operations or new user interfaces) without worrying about the segmentation step.

As future working lines, the main one is to align EDIV with DMTF standards as described in Section 6 in order to take advantage of interoperable virtualization management. Other secondary goals are to fix some issues in the current prototype (e.g. implement the optional W interface) or implement filesystem and kernel management (currently, they are preinstalled in deployment hosts, only scenario specifications are copied through the H interface).

Acknowledgments. This work is supported by the Business Oriented Infrastructure (BOI) group within the Business Support Systems (BSS) unit at Telefónica I+D.

References

1. Virtual Network User Mode Linux (VNUML), `http://www.dit.upm.es/vnuml`
2. Pizzonia, M., Rimondini, M.: Netkit: Easy Emulation of Complex Networks on Inexpensive Hardware. In: TridentCom 2008, Innsbruck, Austria (March 2008)
3. Begnumm, K.M.: Managing Large Networks of Virtual Machines. In: Proc. of 20th Large Installation System Administration Conf (LISA 2006), USENIX, December 2006, pp. 205–214 (2006)
4. VMware VirtualCenter, `http://www.vmware.com/products/vi/vc`
5. Galán, F., Fernández, D.: Distributed Virtualization Scenarios Using VNUML. In: First System and Virtualization Management Workshop (SVM 2007), Toulouse, France (October 2007)
6. IEEE, Virtual Local Area Networks. IEEE standard 802.1q (2001)
7. Dike, J.: User Mode Linux (UML). Prentice-Hall, Englewood Cliffs (2006)
8. CPU Burn-in Homepage, `http://users.bigpond.net.au/CPUBURN`
9. DMTF, CIM Operations over HTTP, DSP0200 v1.3.0c (September 2007)
10. DMTF, CIM System Virtualization Model White Paper, DSP2013 v1.0.0, (November 2007)
11. Libvirt-cim, `http://libvirt.org/CIM`
12. Kivity, A., et al.: KVM: the Linux Virtual Machine Monitor. In: Proc. of the 2007 Linux Symposium, pp. 225–230 (June 2007)
13. Barham, P., et al.: Xen and the Art of Virtualization. In: Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), pp. 164–177. ACM, New York (October 2003)
14. OpenVZ, `http://openvz.org`
15. DMTF, Open Virtualization Format Specification, DSP0243, v1.0.0b (July 2008)

Web Service Distributed Management Framework for Autonomic Server Virtualization

Bogdan Solomon[1], Dan Ionescu[1], Marin Litoiu[2], and Mircea Mihaescu[3]

[1] NCCTLab, University of Ottawa, 800 King Edward Avenue,
Ottawa ON K1N 6N5, Canada
[2] IBM CAS, Toronto
[3] IBM USA, Sommers

Abstract. Virtualization for the x86 platform has imposed itself recently as a new technology that can improve the usage of machines in data centers and decrease the cost and energy of running a high number of servers. Similar to virtualization, autonomic computing and more specifically self-optimization, aims to improve server farm usage through provisioning and deprovisioning of instances as needed by the system. Autonomic systems are able to determine the optimal number of server machines - real or virtual - to use at a given time, and add or remove servers from a cluster in order to achieve optimal usage. While provisioning and deprovisioning of servers is very important, the way the autonomic system is built is also very important, as a robust and open framework is needed. One such management framework is the Web Service Distributed Management (WSDM) system, which is an open standard of the Organization for the Advancement of Structured Information Standards (OASIS). This paper presents an open framework built on top of the WSDM specification, which aims to provide self-optimization for applications servers residing on virtual machines.

1 Introduction

Server virtualization is not a new technology, it was first invented in the 1960s by IBM in order to deal with multitasking and time sharing for the very expensive mainframes of that age. In 1998 virtualization was "rediscovered" for the x86 platform by VMWare, but due to instruction translation the approach taken was slow when compared to bare metal performance. More recently, advances in the form of paravirtualization and direct hardware support for both AMD and Intel processors in the form of AMD-V and Inter-VT technologies have allowed virtual machines to reach near bare metal performance. With such performance, virtual machines offer a number of benefits over running one operating system with one application server on one server machine in the form of better utilization of the hardware at all times, less power consumption, better security and fault tolerance through the use of sandboxed virtual machines. The push for virtualization is accompanied by a shift to Hardware as a Service (HaaS) seen in projects like Amazon's Elastic Compute Cloud (EC2) [1] and FlexiScale. Both

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 61–71, 2008.

EC2 and FlexiScale provide developers with a virtual machine in which they can set up their execution system, in terms of operating system, application server and applications. After the virtual machine is deployed new virtual machines from the base image can be created when they are needed and removed when they are unused. The process of addition and removal however has to be done by a human being. Through the use of these services organizations can pay only for the CPU time and bandwidth that they use, and at the same time ensure the availability of their application.

Recent initial public offering of VMware Inc. shows clearly that virtualization is increasingly important to businesses. Two major players in the industry, VMware and Microsoft Corp., are offering their previously for-pay server products for free, in order to market their more-expensive and more-capable enterprise systems. As expected, the server virtualization literature is quite large and discusses architectures, implementations and performance issues in regards to various server virtualization architectures.

Operations support system (OSS) for enterprises and service providers were devised to offer automation of the configuration, provisioning, and management mechanisms such that a service plan can be implemented with less human efforts. However, little has been investigated in the area of automating and optimizing the delivery of server-based applications and resources.

With the introduction of the Autonomic Computing Manifesto [2], a new field of research was introduced and a series of products started to be marketed. The research efforts aimed to build intelligent systems capable of self-configuring, self-healing, self-optimization, and self-protecting themselves. Such systems have to be able to analyze themselves, determine their state and if needed take the most appropriate action in order to maintain the Service Level Agreement (SLA). Critical applications provide guaranteed response times by ensuring that enough servers are running for the worst possible scenario. While this is extremely inefficient since at non peak times most servers will be underutilized, it also does not guarantee a good response time if the worse case scenario changes. The underutilization of servers also causes extra costs, as all the servers have to be powered. According to a survey done by the International Data Corporation (IDC) servers in data centers are on average utilized only 10% to 15% of total capacity. Self-optimizing systems can be thus applied to clusters of servers in order to provide a guaranteed response time by adding new servers when more clients arrive, and removing those servers when the number of clients decreases. While it would be possible to consider autonomic systems and virtualization as competing systems with the same goals - improving utilization of servers in datacenters, autonomic systems can be used to provide the automation of creating, moving and destroying virtual machines.

In the Autonomic Computing Manifesto [2], Paul Horn identified eight key elements of an autonomic system. One of these elements was the use of open standards. Since an autonomic system is meant to be used in a heterogeneous world, various vendors must be able to create their own systems which should

be managed and interoperate out of the box. This means that the autonomic solution must be independent of the various system it manages, from both hardware and software perspective. For example there should be no difference between managing a WebSphere cluster and a Weblogic cluster in order to obtain a certain response time. This is where the Web Service Distributed Management (WSDM) [5] specification comes into play, by providing an open standard framework which defines a web service architecture which can be used for the management of distributed resources.

As the process of addition and removal of servers in virtual server environment is presently done by system operators, there is therefore, a need to provide a software platform capable of providing self- configuration, self-provisioning, self-optimization and self-protection mechanisms for a complex IT infrastructure. Thus, an autonomic computing control technique for the provisioning of virtual servers on a given hardware platform will be introduced and developed in this paper. The paper provides a solution for the dynamic load control on VMM based virtual computing platforms.

The remainder of the paper is structured as follows: Section 2 introduces the autonomic solution for server virtualization, section 3 shows the architecture of an autonomic based system for management and optimization of the virtual cluster. Section 4 describes the implementation of the architecture while section 5 shows some experiments run. Finally section 6 presents the conclusions.

2 Autonomic Server Virtualization

2.1 Challenges

An autonomic self-optimizing cluster has to solve a number of problems. First of all, such a system must be able to utilize only the server instances that it requires, and not more. In the case of a virtual machine environment, the system must use only the minimum number of virtual machines that allow it to provide the required SLA. At the same time, the virtual machines should be configured in such a way as to not over utilize the available CPU/memory of the hardware machine. Second of all, the system must be able to guarantee a certain SLA. This translates into being able to start the addition of new instances before the instances are required. Preferably the instances will be up and available just in time to take the new load. Since creating, starting and configuring a new instance takes on the order of minutes, the system must be able to predict the behavior of the system's future. Third of all, and possibly most important, the system must be able to deal with a scarcity of resources, provide the required resources to the critical applications and allow the non critical applications to degrade gracefully. For a virtual machine system, this means at the same time allocating more virtual machines for the critical applications and redistributing the ratio of resources allocated to the various virtual machines to favor the critical applications.

2.2 Autonomic System Solution

The solution taken in this paper is based on previous work [3], [6], [8] which sees the autonomic system as a control loop with sensors, filters, controllers and actuators. The goal of the system is to provide a certain guaranteed response time to a number of web applications running inside WebSphere 6.1 application servers, which run on top of Xen virtual instances. In order to achieve its goals, the system makes a decision regarding what new VMs need to be deployed or how to modify the resources used by existing VMs. The decision of adding resources is reached based on a number of factors.

If there are enough resources free in the data center for the required VM, the decision of what to do is based on where the free resources are. If there are enough free resources on a machine that already has a VM running the application needing more resources than the VM is given more resources. If no such machine exists, than a new VM is created on a machine which has enough free resources. In the case where none of the above two can be accomplished, the system calculates a way to add resources to one VM and create a new VM with the remainder resources. The approach of modifying the CPU usage of various VMs is similar to the idea in [4]. The approach is extended in this paper, to include multiple VMs.

If there are not enough resources in the data center, then the system recalculates the resources required by each application, and shifts the resources in such a way as to provide the best possible response time for all applications based on how critical each application is. Figure 1 shows the logic for decision making.

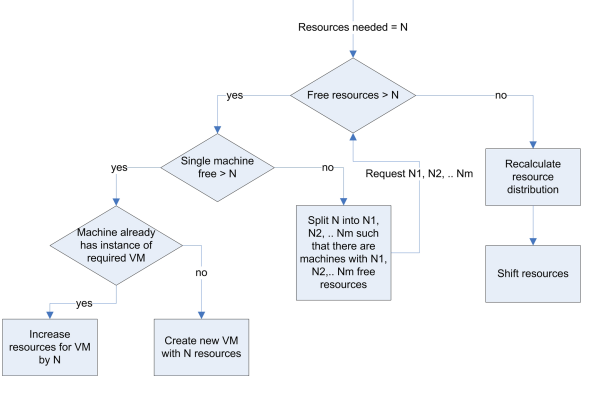


Fig. 1. VM Decision Making

The above solution results in a number of requirements for the autonomic system:

1. The system must be able to shift resources at run time between various VMs residing on the same hardware

2. The system must be able to predict how many virtual resources an application will need
3. The system must maintain a map of available hardware and free resources on hardware
4. The system must provision only as many resources as needed, and add more resources if required in the future
5. Between adding a new instance and adding more resources to an existing instance, the second will be preferred

3 An Architecture for Server Virtualization Using a Web Service Based Autonomic Computing System

In order to build the above system, the architecture used is similar to the one described in [8] and [7] and is composed from the following seven components:

1. Sensors - two types of sensors are used: one which retrieves data from each of the application servers running on top the virtual machines. This data contains information like response time provided by the server, number of clients and what percentage of the CPU available in the VM the application server uses. The second sensor measures data from the VM itself in the form of CPU and memory used by each VM.
2. Filters - in order to modify the data coming from the sensors into data usable by the rest of the components a filter manager implementing a filter chain is used. The filter chain calculates information that is not directly available from the sensors like throughput and think time for the application server. The filter chain also calculates the average metrics across all servers in an application server cluster.
3. Coordinator - the coordinator's role is to provide the main logic of the control loop and drives the communication between the other components. It implements an iterative approach, where the state of the system is repeatedly estimated until two successive estimates are within a certain difference.
4. Model - the model provides a view of the system's state and is used by the other components to provide estimates of the future state of the system.
5. Estimator - the estimator uses the model and the current received data from the filter in order to predict the future state of the system based on a non linear system.
6. Decision Maker - two decision makers are used and the system switches between them seemlesly based on the available resources. When there are available free resources, the decision maker simply decides where to create a new VM and how much of the underlying system's resources the VM will take. The same decision maker decides when to free up VMs that are no longer needed by the applications. The second decision maker is used to decide how to repartition the used resources when no free resources are used,

or when insufficient free resources are available. The same decision maker repartitions resources when a VM's resources are released, if the system was previously strained for resources.

7. Actuator - similar to the decision maker, two actuators are used. The first actuator is responsible for creating new VMs and removing VMs when they are no longer needed. This actuator is also responsible to configure the VMs and start the application server and any other necessary applications. The second actuator takes care of redistributing the CPU time and memory allocated to each VM.

4 Architecture Implementation

Figure 2 displays the architecture described previously as applied to a cluster of WebSphere application servers running on top of Xen VMs. In the figure two VMs, each with an instance of a WebSphere application server are managed. Each instance has two sensors - one from the WebSphere server and one from

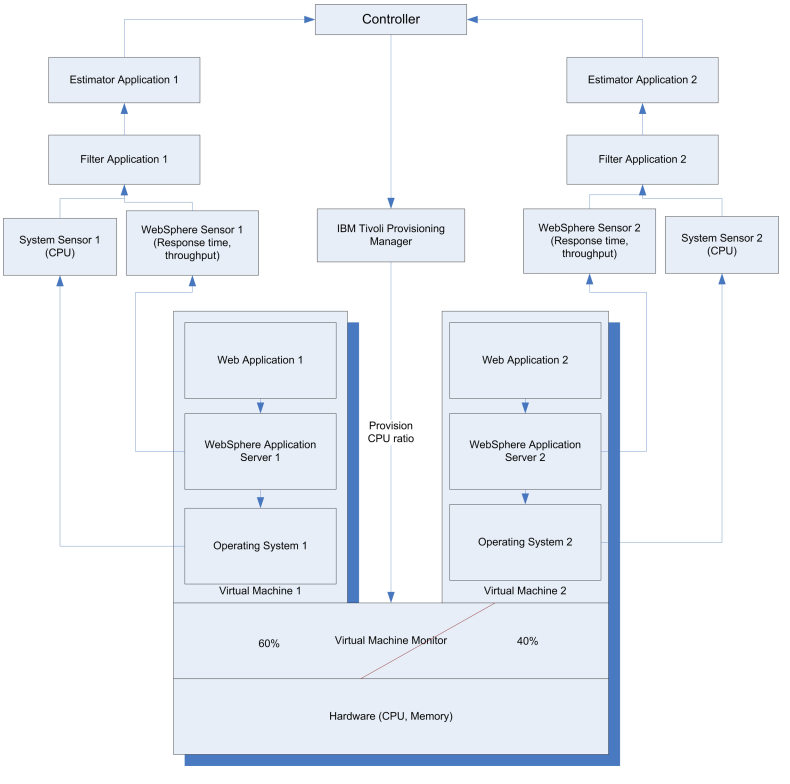


Fig. 2. VM Control Loop

the Xen virtual machine. The data from the sensors is then aggregated by the filters and then passed to the estimators to determine the future state of each of the two applications running on top of the VMs. The estimated data is then passed to the controller which decides how to modify the resources each of the two VMs receive. The actuation is done through the use of IBM Tivoli Provisioning Manager, which runs scripts to adjust the resources received by the VMs.

4.1 Web Service Distributed Management Framework

The implementation of the components described above was done by extending the Web Service Distributed Management (WSDM) open standard, and the Apache Muse implementation of the WSDM standard was used for runtime. Each component is formed from three WSDM resources: a factory component, a management component and a runtime component. Since each WSDM web service endpoint is able to support a large number of resources, each with its

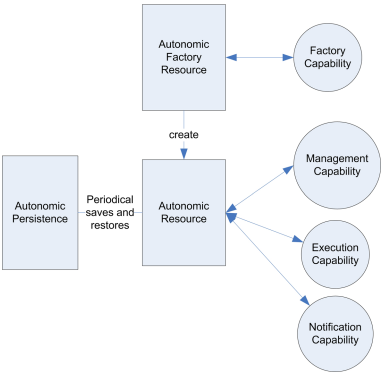


Fig. 3. Resource Capabilities

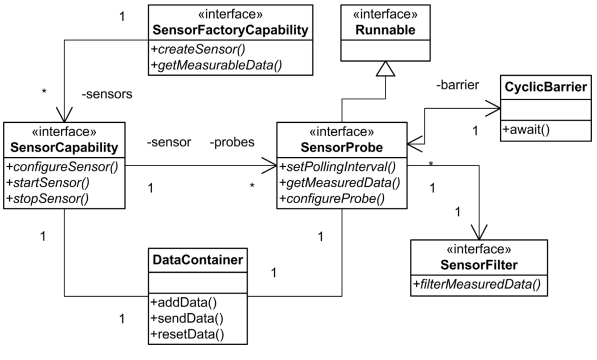


Fig. 4. Sensor Interfaces

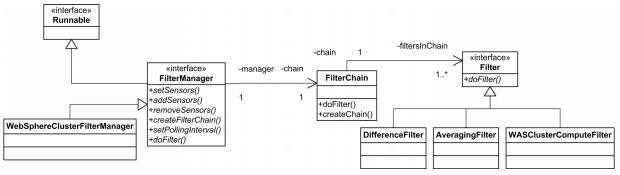


Fig. 5. Filter Interfaces

own unique address provided through the use of Web Service Addressing (WS-A) a factory pattern is used in order to create the components. For example, the same filter web service application could run two filter managers, each of them for a different cluster. The factory has two responsibilities: to reply to queries requesting the available component implementations that can be created at the endpoint, and to create new components based on a predefined type. Upon creation of a new component instance, the two resources for the component are created, the management resource and the execution resource. The management resource allows the configuring and reconfiguring of the resource as well as start, pause and stop commands to be sent to the component. The execution resource performs the runnable logic of the resource. It also sends messages to the other components that it communicates with. Figure 3 shows the resource architecture of a web service endpoint for a component. All messaging between components is done in the form of notifications, through the use of the Web Service Notifications (WS-N) standard.

Each of the seven components implements the interfaces described in [8]. Figure 4 shows the structure of the sensor component. The sensor component has one goal, to retrieve data from the monitored entity and process the data to be passed forward to the filter. One sensor component can be composed of multiple probes, where each probe monitors and retrieves data for one specific part of the system. In the case of an application server, there could be one probe monitoring the response time, number of clients, throughput, etc. of a certain application, and another probe monitoring the underlying system performance like CPU and memory utilization. Due to the composition of the sensor from multiple probes, the probes must be synchronized such that the data returned by the probes represents almost the same moment in time. In order to achieve this a cyclic barrier is used to ensure the synchronization of data gathering by the probes.

Like the sensor component, the filter component has only one role - to filter the measured data into data that can be used by the rest of the control loop. The structure can be seen in figure 5. In order to provide a certain degree of reusability and modularity, the autonomic filter implements a filter chain pattern, where the data is passed successively through a number of filters, each filter taking one action to modify the data. For example, in the WebSphere Application Server cluster example, the data coming from each of the sensors represents the total values for a certain metric since the server had started. What is required by the rest of the control loop is an incremental value for the previous pooling

period. Furthermore, the control loop requires a number of metrics that are not directly available from the sensor, but which can be calculated from the measured metrics, and also require an average value for the whole cluster and not values per server. Each of the above calculations can be abstractized into three separate filters, which can then be chained.

The estimator, model, coordinator and actuator structure is very simple and not shown in this paper. Each of these components implements one function related to its role in the control loop. For example, the coordinator has one function visible outside - coordinate - which takes care of coordinating the control logic and communications between the other components.

5 Experiments

Some simple experiments were run to see the behavior of the system. The testing system was built using Xen Server 3.1 as the host system, with Fedora 8 guests

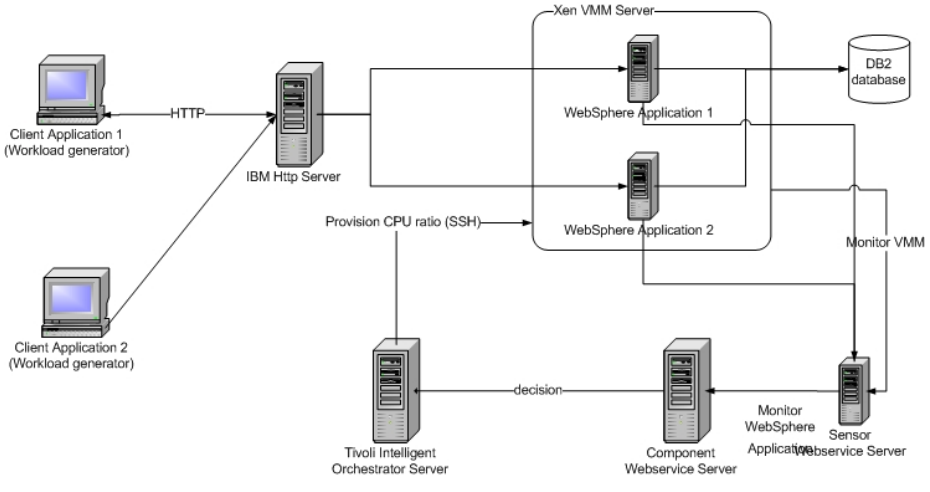


Fig. 6. Test bed

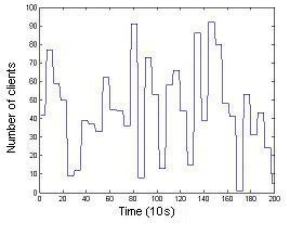


Fig. 7. Clients

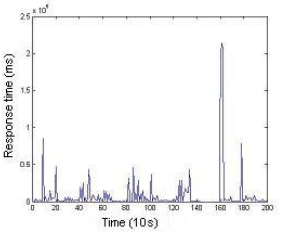


Fig. 8. Response time

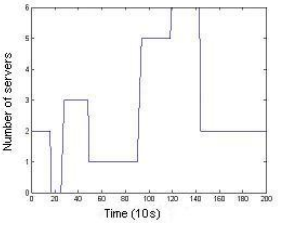


Fig. 9. Predicted server instances

systems running WebSphere Application 6.1 servers. The Trade 6 benchmarking application was then run on top of WebSphere servers and was monitored. Figure 6 shows the test-bed used. The two WebSphere Application Servers reside on one Xen VMM, and have a front end Http server which distributes the requests to the two backend serves. Since the Trade6 application is a three tier application a DB2 backend stores the persistent information for the application. The sensors are then deployed on one server, and the rest of the components are deployed on another server. Tivoli is also set up on a separate machine from the control loop logic.

Figures 7 through 9 show the way the number of clients, response time and number of predicted server instances vary for one of the two applications running on VMs. From these figures it can be seen that the average response time, as expected, is correlated with the number of clients. It can also be seen that the control loop smooths the high variability of the response time values, resulting in a relative small number of changes to the VMs.

6 Conclusions

This paper has presented a solution for combining an autonomic computing infrastructure for self-optimization with server virtualization in order to achieve optimal server usage in data centers. The autonomic infrastructure is based on open standards and uses web services for management of the virtual server resources. Both the addressing of the components that form the autonomic system and the messaging between the components is based on web service standards and form a SOA, where the components represent services that can be created, started, stopped and deployed on the fly. By using a two layered approach to optimization, based on whether there exist free resources for new VMs or not, the system described here can guarantee that the resources are used in such a way as to both provide adequate SLAs, and optimize the critical applications first. At the same time, this solution ensures that only the required resources are used and not any more. By combining server virtualization and self-optimization only the required resources of a hardware server are used for each VM instance, and as more resources are needed by various applications the autonomic infrastructure can provide them based on SLAs. The autonomic infrastructure can also resolve resource conflicts between VMs.

7 Future Work

Future work will include building a more thorough test-bed for the solution described in this paper, that will optimize a number of Java based web applications running on top of multiple VMs. The test-bed will then be used to measure the performance of the proposed solution in terms of resources used, power consumption and meeting of SLAs for the web applications.

References

1. Amazon. Amazon Web Services: Elastic Compute Cloud, http://aws.amazon.com/ec2
2. Horn, P.: Autonomic Computing: IBM's perspective on the State of Information Technology (October 2001), http://researchweb.watson.ibm.com/autonomic/
3. Litoiu, M., Woodside, M., Zheng, T.: Hierarchical model-based autonomic control of software systems. In: ACM ICSE Workshop on Design and Evolution of Autonomic Software (2005)
4. Menasce, D.A., Bennani, M.N.: Autonomic virtualized environments. In: ICAS 2006: Proceedings of the International Conference on Autonomic and Autonomous Systems, Washington, DC, USA, p. 28. IEEE Computer Society, Los Alamitos (2006)
5. OASIS. OASIS Web Services Distributed Management (WSDM), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm
6. Solomon, B., Ionescu, D., Litoiu, M., Mihaescu, M.: Decentralized predictive control of autonomic computing environments. In: 4th International Information and Telecommunication Technologies Symposium, pp. 94–103 (December 2006)
7. Solomon, B., Ionescu, D., Litoiu, M., Mihaescu, M.: An autonomic computing approach to server virtualization. In: 5th International Information and Telecommunication Technologies Symposium, pp. 87–94 (December 2007)
8. Solomon, B., Ionescu, D., Litoiu, M., Mihaescu, M.: Towards a real-time reference architecture for autonomic systems. In: 29th International Conference on Software Engineering and Workshops (2007)

Virtual System Environments

Geoffroy Vallée, Thomas Naughton, Hong Ong, Anand Tikotekar,
Christian Engelmann, Wesley Bland, Ferrol Aderholdt, and Stephen L. Scott*

Oak Ridge National Laboratory, Oak Ridge, TN 37830, USA
{valleegr,naughtont,hongong,tikotekaraa,engelmannc,blandwb,
aderholtwf,scottsl}@ornl.gov
http://www.ornl.gov

Abstract. Distributed and parallel systems are typically managed with "static" settings: the operating system (OS) and the runtime environment (RTE) are specified at a given time and cannot be changed to fit an application's needs. This means that every time application developers want to use their application on a new execution platform, the application has to be *ported* to this new environment, which may be expensive in terms of application modifications and developer time. However, the science resides in the applications and not in the OS or the RTE. Therefore, it should be beneficial to *adapt the OS and the RTE to the application instead of adapting the applications to the OS and the RTE.*

This document presents the concept of Virtual System Environments (VSE), which enables application developers to specify and create a virtual environment that properly fits their application's needs. For that four challenges have to be addressed: (i) definition of the VSE itself by the application developers, (ii) deployment of the VSE, (iii) system administration for the platform, and (iv) protection of the platform from the running VSE. We therefore present an integrated tool for the definition and deployment of VSEs on top of traditional and virtual (*i.e.,* using system-level virtualization) execution platforms. This tool provides the capability to choose the degree of *delegation* for system administration tasks and the degree of protection from the application (*e.g.,* using virtual machines).

To summarize, the VSE concept enables the customization of the OS/RTE used for the execution of application by users without compromising local system administration rules and execution platform protection constraints.

1 Introduction

The architecture for modern distributed and parallel execution platforms differ from single head node/multiple compute nodes Beowulf clusters to distributed Grids and large-scale system with specialized nodes (*e.g.,* I/O nodes). Several tools are available for the management of such platforms [6,8,10,11].

Furthermore, as different system solutions emerge on top of traditional computing platforms, such as system-level virtualization, system management tools have also been extended [15]. While these enhancements allow for the deployment of environments on

* ORNL's research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC for the U. S. Department of Energy under Contract No. DE-AC05-00OR22725.

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 72–83, 2008.

new platforms like virtual machines, they are lacking in terms of customizability – specifically from the perspective of application developers.

Because of that, with current system management solutions, application developers do not gain any flexibility. Applications still have to be *ported* every time developers want to use a new execution platform. However, the science resides in the applications, not in the system software for the execution platforms. Based on this contrast, it is critical to provide a solution that allows application developers to customize their execution environment that will then be deployed on top of the execution platforms. In other words, *the operating system (OS) and the runtime environment (RTE) have to be adapted to the application and not the application adapted to the OS and the RTE of a specific platform.*

To address this issue we propose the concept of a *virtual system environment* (VSE), which decomposes these challenges into two different aspects: (i) the definition of the environment needed to run the application, both according to application developers and system administrators perspective – this high-level description is actually very agnostic about the system configuration of the target system for application execution, and (ii) the deployment of a defined VSE on a target environment. We currently support disk-full/disk-less and physical/virtual systems; and also system partitioning (*e.g.,* I/O nodes versus login nodes versus compute nodes).

The remainder of this paper is organized as follows: Section 2 presents how a VSE can be defined by both application developers and system administrators. Section 3 presents a tool for the deployment of VSEs on top of various system configurations (*i.e.,* physical/virtual, disk-less/disk-full). Section 4 presents VSE benefits for system administration. Section 5 presents the effect of the VSE concept on system protection. Section 7 concludes.

2 Virtual System Environment Definition

An application is typically designed to be executed with a specific version of an OS and RTE. For instance, an MPI application can be designed to run on top of RedHat Enterprise Linux 4.0 with LAM/MPI 7.1.3. This kind of information is decided by developers in order to simplify development. It also means that every time another environment has to be used, most likely the application will have to be modified, ported.

On the other hand, computing centers today provide different execution platforms: clusters, shared memory systems, or even large-scale high-performance systems such as Cray XT or IBM BlueGene systems. Each of these systems typically provide a different execution environment and applications have to be "adapted" to each of them. However, the science resides in the applications and therefore application developers should not have to deal with such porting issues, and should be able to focus on the science.

It is important to decouple the definition of the application's needs in term of RTE and what components system administrators want to have in each environment used by applications.

System-level virtualization provides a first step in that direction, decoupling the environment used for the execution of the applications and the environment used on top of the hardware (virtual machines versus host OS). For instance, it is possible to create

a *virtual appliance*, *i.e.,* a specialized virtual machine for the execution of a given application. However, the concept of appliance does not ease the definition of the application environment; the system within the VM still need to be more or less manually installed. Furthermore, because of the lack of meta-data defining what the application environment is, it is not possible to deploy an existing virtual appliance on top of a standard system (*i.e.,* disk-less or disk-full system).

Additionally, as discussed in Section 6, system management tools have been extended to support virtual environments but suffer from some significant limitation. OSCAR-V [15] is such a tool, managing virtual machines and creating images for virtual machines with a minimal system footprint (only needed software is included into the image), but it is not possible for administrators and users to easily express their definition of execution environments (OSCAR-V recognizes only Beowulf clusters [14], and not Grids or large-scale systems) and it is still difficult for users & application developers to define their execution environment[1].

The VSE concept aims to address these challenges and has been implemented as an extension of the OSCAR-V prototype: (i) the VSE fits application needs, no unnecessary system footprint in included; (ii) the OS type & version and the RTE are chosen by application developers and not by system administrators; (iii) system administrators can check the VSE before deployment; (iv) application developers can define their VSE off-line from the execution platform; and (v) the VSE can be deployed automatically by system administrators. Because the VSE implementation is actually a non-intrusive extension of OSCAR-V, performance for VSE creation and deployment is actually similar to OSCAR-V performance. Thus, we do not present performance results in this document, only the VSE architecture and implementation is described in details. Application developers define their RTE needs using a high-level language based on XML, which describes a set of software packages (*Package Sets*). A package is an abstraction for the local management of software that aims at easing the installation, configuration and removal of software in a given local system. More details are presented in Section 2.2. In mathematical terms, our notion of sets follows the Zermelo-Fraenkel set theory, with the axiom of choice (ZFC). It means that a collection of "operations" are available for the package set mechanism. From the usage point of view, only a subset of "operations" are important: it is possible to combine package sets and get the intersection of package sets. These operations provide a very flexible method for the definition of complex VSEs.

2.1 Package Sets Definition

Package Set Combination. It is possible to combine package sets together:

$$PackageSet_A \cup PackageSet_B$$

This enables the combination of VSE definitions from application developers and system administrators (see Figure 1).

[1] This may lead to conflicts between applications' needs and system administrators' needs; we do not provide an automatic solution to manage these conflicts since most of the time there are policy issues.

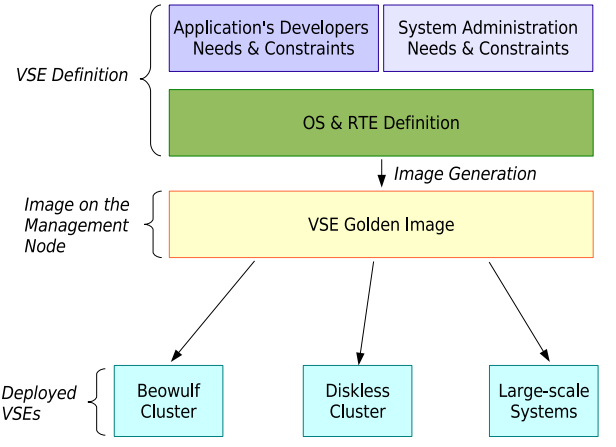


Fig. 1. VSE Definition and Management

For instance, if system administrators, based on local policy, want to include the Moab software [9] in all VSEs because it is the chosen workload manager used by the computing center, they can create a VSE definition that will be combined with an application's developers definition. The resulting specification incorporates the constraints from both the application and system administrators.

Package Set Intersection. It is also possible to define the intersection of package sets:

$$PackageSet_A \cap PackageSet_B$$

The intersection operation is more suitable for advanced capabilities rather than the strict definition of a new VSE. For instance, the intersection operation can be used to identify common software components between several VSEs. That can be used later on by system administrators in order to identify current needs of application developers in term of settings of the execution environment, and therefore try to address more efficiently present needs and anticipate future needs.

Package Set Validation. The package set mechanisms also include basic validation capabilities in order to ensure that package sets can be correctly combined. This validation tool is based on a versioning mechanisms (comparison of software version), and a dependency mechanism (set A depends on set B but conflicts with set C).

If users combine several package sets together the system also checks that the Linux distribution from the different package sets are the same.

Versioning. Users can specify the version of each package within a package set. This allows for fine grain software management. For instance, application developers can specify that their application needs a specific version of a library. We provide standard operators to deal with versioning: *equal (eq)*, *superior to (gt)*, *inferior to (lt)*, *superior or equal to (gte)*, and *inferior or equal to (lte)*.

2.2 Package Sets Usage

Package sets define a VSE and are used to create a "golden image" which is agnostic of the target platform execution configuration. The current VSE implementation relies on OSCAR [10], a system management software.

The package sets implementation actually directly relies on *OSCAR Packages* (OPKGs) which allow one to define a software package for software installation in distributed or parallel systems, including information such as a list of binary packages, configuration scripts and versioning information. It typically extends the standard notion of binary packages, adding information about what has to be done to have the software setup at the global level of the distributed or parallel systems. Note that we assume application developers provide their application via an OSCAR package.[2]

The current implementation supports the definition of single package sets and the combination of package sets. The package set intersection operation has not yet been implemented.

Package Set Analysis. In order to ease system administration tasks and to track modifications, information about package sets are stored in a database, the OSCAR Database (ODA).

The first step for the creation of a VSE is therefore the parsing of the VSE's XML file, its validation via XML tools and the update of the OSCAR database. We will see later that information in the database is used to update a basic image, which ultimately results in a golden image that matches the VSE definition.

The validation is composed of two phases: (i) the basic validation of the XML file using standard XML tools, (ii) the validation of the list of OPKGs from the package set. OSCAR provides a tool (OSCAR Package Downloader - OPD) for managing OPKG repositories, which can be used to download OPKGs. OPD2, especially developed for the VSE support, allows us to get the list of all the available OPKGs, for all supported Linux distributions. OPD2 also saves information about OPKGs into the database. Based on this list, it is possible to validate package sets (*e.g.,* checking if OPKGs are available).

Creation of a Basic Golden Image. Based on package sets, a "golden image" [2,10] can be created and used for the actual deployment of a given VSE. For that, we (i) analyze the package set(s), (ii) create a basic golden image for the target Linux distribution, and (iii) install the different OPKGs based on the package set definition.

The creation of basic golden image relies on the OSCAR version of the SystemInstaller software [2]. The creation of the basic image is based on the Linux distribution and the architecture specified in the package set(s). During the image creation, the OSCAR database (ODA) is updated in order to initialize information about the new image.

Once the basic image is created, and based on information about the package set from ODA, it is possible to finalize the golden image installing the OPKGs associated to the package set.

[2] OSCAR packages are based on binary packages (*e.g.,* RPMs or Debian packages), the creation of new OSCAR packages is fairly simple if application developers already provide binary packages for their application.

OSCAR did not have a stand alone tool for OPKG management, initially all OPKGs were installed directly into the image, using SystemInstaller, without using information in ODA. We therefore implemented the *OSCAR Package Manager* (OPM) tool. This tool queries the database to know the exact status of images, OPKGs and nodes. Based on this information, OPM installs OPKGs into images but also on remote nodes if nodes have already been deployed. The image is then ready to be deployed and the database up-to-date for management purpose.

3 Virtual System Environment Deployment

We target three different platform architectures: (i) Beowulf clusters, (ii) disk-less clusters, and (iii) large-scale systems (*i.e.,* platforms with specialized nodes).

We saw in Section 2 that it is possible to have an XML file which describes the VSE that has to be deployed and to create the associated golden image. Because we do not want to have to recreate the image every time we deploy it (for instance to have image persistence), we take care to separate tools for image creation and mechanisms to deploy them. In other words, the VSE's XML description is used to generate a "golden image" on the management node. Then, this golden image is "adapted" to fit the platform architecture. This adaptation is based on the description of the target platform.

3.1 Machine Sets

In order to express the topology of the target system, we introduced the notion of node sets (also called node groups). This concept, like the package sets concept, follows the Zermelo-Fraenkel set theory, with the axiom of choice (ZFC). This includes the support of union and intersection operations on machine sets:

$$MachineSet_A \cup MachineSet_B$$

$$MachineSet_A \cap MachineSet_B$$

A key characteristic of node sets is the need to express dependencies between the machines in the set. To address this issue, we assume that all relationships between nodes can be expressed as server/client dependencies, *i.e.,* more precisely as a one-to-one dependency (set A is dependent upon set B). For instance, for I/O nodes where three types of nodes are used (meta-data server, data storage server and compute nodes), two different node sets can be defined: (i) a set for the meta-data server and the data storage server where the meta-data implements the notion of server for the machine set, the data storage server being the client; and (ii) a set for the data server (both meta-data and data storage server) and the compute nodes. To define complex systems (*e.g.,* a large-scale system composed of login nodes, compute nodes, and I/O nodes), different machine sets can be defined, one for each type of nodes.

3.2 Image Deployment

Figure 2 shows the overall architecture for VSE deployment mechanism for the support of Beowulf clusters, disk-less systems and large-scale systems. Note that the figure includes the relationship with the mechanism for the creation of the golden image.

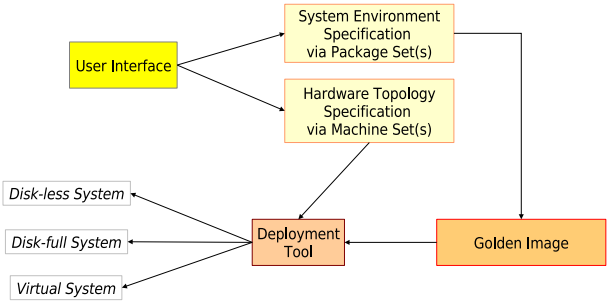


Fig. 2. Overall Architecture for the Deployment of VSEs

Beowulf Clusters. Beowulf clusters are still the standard architecture for clustering: a headnode provides all clustering services and compute nodes do the application computation using services from the headnode. This is a standard client/server architecture.

To describe this architecture only two node sets are needed: (i) the headnode, and (ii) the compute nodes.

Disk-less Clusters. Disk-less clusters may be deployed in many different ways. Currently, we use the standard NFS-ROOT [3] or RAMFS solutions, which is sufficient for small to medium sized clusters (we assume that for large-scale systems, the standard design for such systems is used).

In this case, the system may be categorized into two groups, like for Beowulf clusters: (i) the headnode, or server; (ii) the compute nodes.

The difference with Beowulf clusters is that the image is not "deployed": the image is copied on the headnode, making the difference between shared data and modified data. Then compute nodes are booted and use their own image.

OSCAR did not initially support disk-less clusters. We developed an extension of OSCAR for the support of NFS-ROOT and RAMFS based disk-less support. This support is based on the tuning of images on the headnode. A golden image is divided into two parts: (i) a *shared image* for the part of the file system that can be shared between nodes (read-only), and (ii) a *private image* for the part of the file that needs to be in read/write mode. It is also possible to fall back to a disk-full solution (logically merging the two images for deployment).

Large-Scale Systems. For large-scale systems, the situation is different because this kind of architecture is no longer based on the idea of one single server and many compute nodes. Typically, for this kind of architecture, nodes are grouped into different sets: compute nodes, "service nodes" (*e.g.,* for the parallel file system), and "login nodes".

It is possible to describe a server/client dependency relationship between the different nodes involved in a single service (for instance the I/O subsystem), combining node groups together. For instance, PVFS [12] has three kinds of nodes in order to implement the parallel file system: meta-data server, storage nodes and clients. It is possible to say that the meta-data server has a server/client dependency and then combine these two

into a single node group and create a dependency between this group and the compute nodes.

The current implementation allows one to describe different node groups and to combine them together. Based on this mechanism and the assignment of one specific image to a group of nodes, it is possible to deploy complex large-scale systems.

Virtual Systems. Another solution for the deployment of VSEs is the usage of virtual machines. In this context, the VSE can be instantiated via virtual appliances that can be viewed as a minimal system configuration specialized for the execution of a given application. Thus, VSEs can be considered as a specification tool for virtual appliances.

We previously extended OSCAR, creating OSCAR-V [15], to support system-level virtualization. One of the benefits of OSCAR-V is the ability to support several system-level virtualization systems via the V2M abstraction layer. This allows one to switch between virtualization solutions without re-deploying virtual machines.

Combining the concept of VSEs and features from OSCAR-V, users can take full advantage of virtualization, simplifying the management of virtual systems and improving the customizability of execution environments.

4 System Management

The administration of computing systems must strike a balance between the required system aspects and those which are strictly end-user specific. The ultimate goal being to support users and their computational needs. However, the responsibility of maintaining the system typically does *not* lie in the hands of the individual(s) most familiar with the applications using the resources.

VSEs offer an interesting means by which system administration tasks can be delegated to the end-user who is most aware of the application's needs. The extent to which these system administration tasks are delegated may differ based on the approach used for implementing the VSE, *e.g.,* node partitions, disk-less nodes, virtual machines. The extent of delegation must be commensurate with the selected protection scheme. For example, the VSE might be a common system image that users customize and deploy on a set of disk-less nodes or could be entirely user generated based upon virtual machine platform specifications. In either case, the proper level of privileges is matched with the degree of customization by the system administrator.

This provides a basis to use VSEs to improve user control for specialization, which can be used for systems research testbeds or to simply provide a consistent platform environment for scientists. A VSE also provides a good basis to empower user expertise, which is commonly found on large scientific systems. In many situations these users may require older libraries/compilers or even operating system kernels, which are easily supported through the use of a VSE.

As presented in Section 2, both the system administrators and the users can define their own package set and machine sets. These sets are then merged to describe both the system environment and the hardware partitioning that fits both system administrators and users needs. In summary, the VSE concept allows more flexible management of the environment used by applications without compromising the local system administration policies.

5 Protection

Because the VSE concept enables the customization of different types of systems (*e.g.,* disk-full & disk-less Beowulf clusters, virtualized systems), we provide a sliding scale of protection. In other words, based on the system configuration described via package sets and machine sets, it is possible to increase or decrease the degree of protection for both the user and system administrator: (i) system administrators can protect the execution platform from malicious applications or trust application and give them direct access to the hardware for performance purpose; and (ii) the application developers can choose to run the application directly on top of the bare hardware, with the risk to have to modify the application, or to run in a virtual environment in order to ensure a similar execution environment on all the VSE enabled platforms.

The protection mechanism is typically tied to the degree of customization supported by the system, *i.e.,* the more you can change/customize the more likely you may want to dial the protection level up, ultimately using virtual environments for maximum isolation (see Table 1).

Table 1. System Characterization According to Protection and System Administration Delegation Capabilities

System Type	Protection Level	System Administration Type
Disk-full Beowulf Cluster	Low	Central system administration
Disk-less Beowulf Cluster	Medium	Central system administration
Virtual System	High	Delegation possible

6 Related Work

The HARNESS project [5] studies the launch of a virtual environment at job start, this virtual environment being installed by the runtime environment for a particular application. This capability enables the deployment of virtual RTEs that fit application's needs. However, this study suffers of limitations in term of flexibility for the creation of a complete virtual RTE, especially in term of "virtual hardware" (it is not possible to support solutions based on system-level solutions) and HARNESS does not provide integrated tools for system deployment.

The *Modules* system provide environment customization at the level of a user's command interpreter (shell) [4]. The Modules system is responsible for managing the differences between command shells, *e.g.,* bash, csh. A system administrator provides the available software and configuration setting via a Tcl file that may be loaded at shell invocation. The users customize their execution environment by loading the appropriate "modules", *e.g.,* `module load mpi/lam-7.0.6`. These command can be made persistent using a higher level tool like Env-Switcher [10]. While the Modules system is widely used and quite useful, it is limited to changes that can be made at the command interpreter level. Therefore, alternate kernel versions or entirely different operating systems are not an option with this approach.

VMPlants [8] is a solution for the management of *virtual execution environments* in a grid context. A virtual execution environment is defined by a graph which allows users to customize their virtual execution that can be then deployed within virtual machines (using VMWare [16] or User Mode Linux [7,1]). However, VMPlants assumes that a system already exists on each machine the user will use. VMPlants does not provide any solution for the management of this system but also does not provide tools and methods for the interaction between the site system administrator and the application's users. It is therefore not possible to enforce the use of specific software within the virtual machine (for instance the use of a checkpoint/restart solution); it is not possible to check if the virtual environment defined by the application users is compliant with local system usage policies, and the management flexibility offered to users is not available to system administrators. Finally, it is not possible to deploy various type of execution platforms based on VMPlants, the use of virtual machines is mandatory.

The *Collective* project [13] is based on the idea of *virtual appliances*, which are application specific bundles that an author (vendor) maintains and end-users use with limited or no administration responsibilities. They employ a virtual machine monitor (VMM) to provide a trusted computing base and effectively a hardware abstraction layer (HAL) to ease appliance portability. An appliance is a specialized single purpose system, *e.g.,* word-processing-appliance, that a user may use but does not maintain (administration is done by the appliance author). This concept is similar to that of a VSE, but differs in scope and how composition is achieved. The VSE is primarily targeted at HPC environments, whereas the virtual appliances are focused on general purpose desktop environments. The appliances are not extended or changed by the user to build up their environment, instead a collection of separate appliances are used in concert (each administered independently). Both approaches use virtualization to assist with portability and enhance usability. However, the Collective assumes multiple appliances (virtual machines) could be run simultaneously where the VSE would typically encompass a single virtual machine.

7 Conclusion

This document presents the concept of a *Virtual System Environment* (VSE), which has been implemented via extensions and/or modifications of the OSCAR and OSCAR-V system management suites. A VSE decouples the definition of the execution environment from the actual deployment method. Users can therefore define application needs and constraints in a generic way. On their side, system administrators, depending on the degree of trust and local management policies, can deploy the VSE into various system types (*e.g.,* disk-less versus disk-full systems, physical versus virtual systems) on specific system partitions. Therefore, VSEs introduce a high degree of customization for application developers, end-users and system administrators. For that, we introduce the notion of *package sets* and *machine sets* which provide a flexible way to define the execution environment and the hardware topology, respectively.

To summarize, through the VSE concept, it is possible to revisit the traditional system administration rules without compromising the platform protection or increasing

the number of administration tasks. In fact, the system protection can even be increased when using virtual environments, which can be done without a great deal of management effort (system-level virtualization is natively supported by OSCAR-V). System administration tasks can also be decreased, delegating some tasks to application developers.

This is especially useful for virtual systems: the system administrator can define what is the host OS, without paying attention to the system-level virtualization solution; and application developers can focus on the definition of the VSE that will be deployed into the virtual machines.

References

1. Brockmeier, J.: The Definitive Guide to User Mode Linux. APress (2004)
2. Dague, S.: System Installation Suite Massive Installation for Linux. In: The 4th Annual Ottawa Linux Symposium (OLS 2002), Ottawa, Canada, June 26-29 (2002)
3. de Goede, H.: Root over nfs clients & server howto, http://www.clusterresources.com/pages/products/moab-cluster-suite/workload-manager.php
4. Furlani, J.L., Osel, P.W.: Abstract Yourself With Modules. In: Proceedings of the 10th Large Installation Systems Administration Conference (LISA 1996), Chicago, IL, September 29–October 4, pp. 193–204 (1996)
5. Geist, G.A., Kohl, J.A., Scott, S.L., Papadopoulos, P.M.: HARNESS: Adaptable virtual machine environment for heterogeneous clusters. Parallel Processing Letters 9(2), 253–273 (1999)
6. Georgiou, Y., Leduc, J., Videau, B., Peyrard, J., Richard, O.: A tool for environment deployment in clusters and light grids. In: Second Workshop on System Management Tools for Large-Scale Parallel Systems (SMTPS 2006), Rhodes Island, Greece (April 2006)
7. jorg, H., Oxer, H., Hoxer, H., Buchacker, K., Sieh, V.: Implementing a user mode linux with minimal changes from original kernel (2002)
8. Krsul, I., Ganguly, A., Zhang, J., Fortes, J.A.B., Figueiredo, R.J.: Vmplants: Providing and managing virtual machine execution environments for grid computing. In: SC 2004: Proceedings of the 2004 ACM/IEEE conference on Supercomputing, Washington, DC, USA. IEEE Computer Society, Los Alamitos (2004)
9. Moab workload manager, http://www.clusterresources.com/pages/products/moab-cluster-suite/workload-manager.php
10. Mugler, J., Naughton, T., Scott, S.L., Barrett, B., Lumsdaine, A., Squyres, J.M., des Ligneris, B., Giraldeau, F., Leangsuksun, C.: OSCAR Clusters. In: Proceedings of the 5th Annual Ottawa Linux Symposium (OLS 2003), Ottawa, Canada, July 23-26 (2003)
11. Papadopoulos, P.M., Katz, M.J., Bruno, G.: Npaci rocks: tools and techniques for easily deploying manageable linux clusters. Concurrency and Computation: Practice and Experience 15(7-8), 707–725 (2003)
12. PVFS: Parallel virtual file system, http://www.parl.clemson.edu/pvfs
13. Sapuntzakis, C., Lam, M.S.: Virtual Appliances in the Collective: A Road to Hassle-free Computing. In: Proceedings of HotOS 2003: 9th Workshop on Hot Topics in Operating Systems. USENIX (2003)

14. Sterling, T., Savarese, D., Becker, D.J., Dorband, J.E., Ranawake, U.A., Packer, C.V.: BEOWULF: A parallel workstation for scientific computation. In: Proceedings of the 24th International Conference on Parallel Processing, Oconomowoc, WI, pp. I:11–14 (1995)
15. Vallée, G., Naughton, T., Scott, S.L.: System management software for virtual environments. In: Proceedings of ACM Conference on Computing Frontiers 2007, Ischia, Italy, May 7-9 (2007)
16. VMware, Inc, http://www.vmware.com

Authentication in Virtual Organizations: A Reputation Based PKI Interconnection Model

Ahmad Samer Wazan, Romain Laborde, Francois Barrere, and AbdelMalek Benzekri

IRIT Institute
Toulouse, France
{wazan,laborde,barrere,benzekri}@irit.fr

Abstract. Authentication mechanism constitutes a central part of the virtual organization work. The PKI technology is used to provide the authentication in each organization involved in the virtual organization. Different trust models are proposed to interconnect the different PKIs in order to propagate the trust between them. While the existing trust models contain many drawbacks, we propose a new trust model based on the reputation of PKIs.

Keywords: Virtual Organization, PKI, X.509 Certificate, Reputation, Trust Model.

1 Introduction

The Concept of virtual organization (VO) is a direct consequence of the network evolution. VO offers a framework to make people belonging to different organizations to collaborate, share experiences, resources, data as if they were in a single enterprise. This collaboration means that employee with specific skills from an organization will access resources provided by another organization in order to fulfill the objectives of the common project. VO organization being multi administrative brings security management issues, especially concerning the authentication process.

X.509 certificates and PKI management is widely used by organizations as the authentication technology for identifying their assets, which could be either employee or machines/servers (Fig.1). They hold an existing PKI infrastructure and skilled administrators to act as the certification authority (CA). The CA has its own practices for controlling the validity of the identity specified in the certificates it issues. An entity, called relying party (RP), within the organization authenticate another entity that belongs to the same organization, using the X.509 certificate because it trusts its CA and its CA's practices.

In the context of VO, where entities belongs to different PKIs managed by different CAs, the problem of trust becomes more complex. To facilitate the management of trust, the authorities of certification can define trust relations between them. Interconnecting CAs is a huge problem which is manually performed by security experts and organizational people, trying to understand if the PKIs have similar quality of certificate management practices. Indeed, a CA that does not take care of the real

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 84–95, 2008.

identity of the employees when it creates certificates could reduce the overall security of the partners that have better practices who won't get assurance about the identity of people from this organization. Today, the evaluation of the quality is based on different documents (the Certificate policies and the Certification practice statement) that are published by each CA. These documents present the procedures that the CA claims to follow.

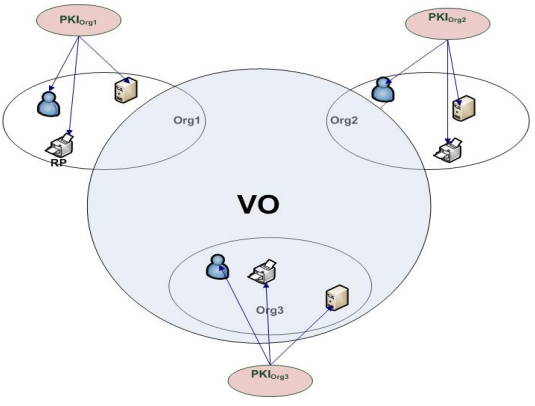


Fig. 1. Virtual Organization

In this article, we propose a new trust model based on the reputation of PKIs and their practices. We use reputation as a quantitative indicator provided to PKI administrators to determine the trustworthiness of foreign certificates. Our model includes both an architectural model that presents the different entities involved in the calculation of the PKI's reputation and a calculation process based on the combination of two formal models.

The rest of this paper is structured as follows. Section2 summarized the different PKI interconnection approaches. Section 3 analyzes the concepts of trust and reputation, and presents some computational models. In section 4, an architectural model is developed. Section 5 exposes how to apply reputation in the context of VO and illustrates the model through an example. Finally, section 6 concludes.

2 Trust Models

A relying party (RP) is ensured about the identity of an entity because it does trust the electronic certificate that contains the name of the entity. In the X.509 certificate framework, the trust on a certificate is based on the trust the RP has on the certification authority (CA) that creates/manages the certificate.

The trust domain of an RP is the set of CAs it does trust. Managing the trust domain when it is confined to the organization is not a hard task. When a RP has to make a transaction with another entity from the same organization, it does trust the certificate because both have the same CA that they trust. However, in the context of VO, each organization has its own department or team that plays the role of authority

of certification and issues/manages the certificates of all the organization's assets, which can be employees and machines or servers as well. This implies that the RP has to trust the CA of another organization when it makes a transaction with an entity from another organization. Thus, the trust domains should be extended by interconnecting the different organization managed PKIs involved in the VO.

We present in this section the main PKI interconnection models that have been proposed. Although they propose technical approaches for extending trust domains, RP or its PKI administrator has no quantifiable indicator to decide if extending the trust domain is secure or not. In addition, trust is total and static in these models.

2.1 Hierarchy Model

In a strict hierarchy, there is one trusted root CA which issues a certificate to subordinate CAs. Those CAs may, depending on the relevant policy, certify other CAs. Each CA is trusted because the higher CA certifies it can be trusted. Only the root CA must be trusted on its own. The hierarchical structure means there is a short and definite path to trace a certificate back to a trusted source.

However, in context of VO, this model suffers the following limitations:

- The root CA cannot constitute a single responsibility point. It cannot take on all the liabilities of errors caused by the leaf CAs in the different organizations. In practice, the liability rests an issue between the RP and the individual CA in the hierarchy,
- The root CA has no authority on the certificates for all the assets of an organization. The root CA has authority only on the certificates of the assets involved in the VO but not on the other assets that do not belong to the VO.

2.2 Certificate Trust List Model

The Certificate Trust List (CTL) is a list of CAs' certificates from a trusted authority. The list itself is electronically signed to ensure its integrity. While CTLs are simple, they provide a very useful device for communicating trust.

Trust lists have also given rise to the 'browser' model - the most widespread interoperable PKI by virtue of web browser applications (such as Internet Explorer, Netscape or Firefox). These browsers use a list of pre-loaded certificates from several dozen of the largest and most reputable CAs such as Verisign, RSA and Thawte. Almost all e-commerce websites display a certificate issued by those particular CAs. When a browser visits their site, their certificates are automatically recognized and the user has some assurance that the web site is from the organization it claims to be.

In context of VO, the management of trust lists is a problem. When a VO is created, the administrator must add the CA of each organization involved in the VO to the list of the trusted CAs for each asset of his/her organization related to the VO. When the project is over these CAs must be deleted. The situation becomes more complicated when assets of organizations are part of multiple VOs.

2.3 Cross-Certification Model

Cross-certification is a different approach to interoperability. Instead of a hierarchy, CAs deal with each other as peers and choose whether or not to trust each other.

If they do, the CAs issue cross-certificates to each other. A relying party can then trace a certificate from an unknown CA back to a local trusted CA. This may be implemented by allowing users to contact their trusted CA's repository of certificates or by including a chain of signatures on the certificate itself.

However, achieving interoperability through a mesh of certifications is technically and logistically challenging. It is not easy for a single pair of CAs to co-ordinate their policies and technical systems, and as the mesh grows, the number of cross-certifications grows even faster. If every pair of CAs cross-certifies (to create a fully meshed network), the number of cross-certifications required is almost n^2 (if n is the number of CAs). Cross certification is most suited when the number of partners in the VO is limited.

In context of VO, this model requires that all CAs has the same level of quality. That means the two CAs' services are regarded as equal with respect to quality of their certificate management practices (practices to identify the subjects, to store the keys, to manage revoked certificates, etc.). The complexity involved in the policy mapping depends on the differences between the policies. However, not all the CAs can have the same quality.

2.4 Bridge CA Model

The bridge CA model is based around a central (bridging) CA which cross-certifies with each CA. It functions as a communication channel between each of the CAs.

This combines aspects of both the root model and the cross-certification model. It provides much of the administrative simplicity of the root model, because it only requires one pair of cross-certifications for each CA, rather than n2 certifications in a completely meshed system.

It also provides some of the flexibility of the mesh model because it is not conceived as imposing strict technical requirements on complying CAs. Nevertheless the bridge must set certain minimum standards for CAs to participate. For example, in USA, the Federal Bridge CA specifies several different levels of assurance that a CA can be certified at. The bridge itself is focused on the task of providing interoperability.

A bridge CA playing the role of an administrative root CA, this model suffers from the same problems as the hierarchy model.

3 Trust, Reputation

Trust and reputation are related, but distinct, concepts. Trust represents an agent's individual assessment of the reliability, honesty etc. of another, while reputation is a social notion corresponding to a group assessment of such issues. In this section we will explain these two concepts.

3.1 Definitions of Trust and Reputation

Defining trust is quite difficult because it appears in many different forms [9]. In this article, we consider the definition given by Gambetta [8] that states "*Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends*".

Reputation is generally built by combining trust assessments (or recommendations) given by a group of agents to obtain a single value representing an estimate of reputation [9]. Consequently, reputation is public information which depends on the recommendations of the society where the entity is located.

The difference between trust and reputation can be illustrated by the following perfectly normal and plausible statements [9]:

(1) "I trust you because of your good reputation."

(2) "I trust you despite your bad reputation."

The statements (1) and (2) indicate that the reputation is a *quantitative indicator* that allows an entity to decide if it will grant trust or not on another entity. This indicator provides information to evaluate the associated risk. In addition, reputation being based on external recommendations, this indicator value can evolve all the time including the period of the life of the VO.

3.2 Computational Models of Trust and Reputation

There are several approaches to calculate reputation, such as:

- Simple Summation or Average of Ratings [1, 2]: the simplest way of computing which is simply to sum the number of positive ratings and negative ratings separately, and to keep a total score as the positive score minus the negative score.
- Bayesian Systems [3]: Bayesian systems take binary ratings as input (i.e. positive or negative), and are based on computing reputation scores by statistical updating of beta probability density functions (PDF).
- Discrete Trust Models [4]: in this model, the votes are not continuous measures but they are discrete verbal.
- Belief Models [5]: this model is related to probability theory, but where the sum of probabilities over all possible outcomes not necessarily adds up to 1, and the remaining probability is interpreted as uncertainty.
- Fuzzy Models [6]: Trust and reputation can be represented as linguistically fuzzy concepts, where membership functions describe to what degree an agent can be described as e.g. trustworthy or not trustworthy. Fuzzy logic provides rules for reasoning with fuzzy measures of this type.
- Flow Models [7]: Systems that compute trust or reputation by transitive iteration through looped or arbitrarily long chains.

The calculation model of reputation must integrate a function:

- To collect the different testimonial estimates from other entities,
- To modulate the impact of each type of information on the final score of reputation depending on different criteria (date, reliability of recommendation, etc.).

In addition, the entities that are involved in the calculation of the reputation score should be determined. The next section presents the architectural model. Then the calculation model is proposed.

4 The Proposed Architectural Model

Several entities in X.509 PKI are involved in the calculus of reputation. We propose an architecture which includes the various entities that take part in the calculation of the reputation of X.509 PKI and their interactions (Fig. 2).

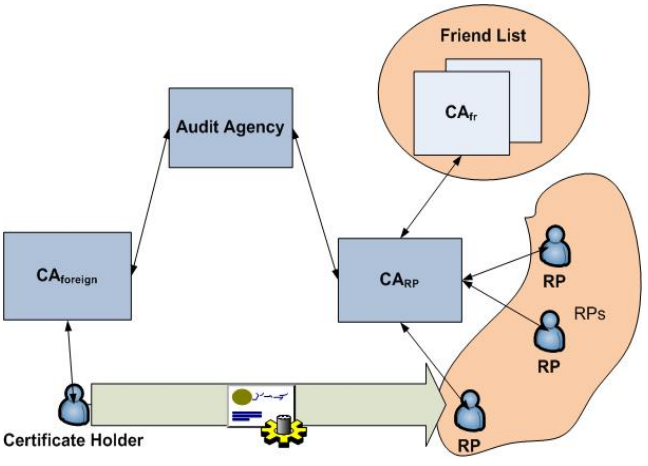


Fig. 2. The Proposed Model

4.1 Relying Party (RP)

Relying party represents an offered service or any other type of entity who wants to verify the identity of another entity holding a certificate signed by an authority doesn't belong to its trust domain. In this model, we suppose that the RP gets the authorization from the administrators of its CA in order to accept or not the certificate, i.e., the RPs delegate its CA's administrator to verify the appropriateness of the certificate.

At the end of each session, the relying parties must vote if the certificate content was valid and there were not any errors caused by the certifying authority of the certificate. There are many issues in the work of PKI could be voted by the relying party such as the availability of the OCSP server used to check the certificate status, the liabilities taken on by the CA in case of problems, the identity information is meaningful or not, the clarity of the relying party agreement, etc. This vote is communicated to the administrator of the RP's CA.

4.2 Audit Agency (CPS Quality)

The determination of reputation is an important factor but it is not sufficient. In fact, RP needs to know also the quality of the procedures followed by the PKI in order to publish the certificate (CPS quality) [13].

PKI is not only a technical project but also a legislative economic project which is related to different domains. Consequently the determination of quality requires a good knowledge on various fields which is impossible to have for a normal person

and even sometimes for experts. For example, this person cannot estimate the force of cryptographic algorithm used to build the public keys.

The most detailed and relevant suggestion for the formal presentation of certificate policies was described in [10]. Chokhani and Ford [10] enumerated a list of factors CAs must take into account in order to describe the procedures used to issue and manage the certificates. These factors include the practices followed by CA to authenticate the users, the policy of the CA, the procedures of safety, the engagements of the user (for example, the protection of his private key), and legal CA engagements (for example, guarantees and limitations on the responsibility). PKIs should follow this list and publish a report called Certificate Practice Statement (CPS). By consulting this file, a qualified person can decide if the announced procedures in this report are suitable or not.

To face this problem, Ball and al [11] have proposed to build an expert system. This system gives a score between 0 and 1, partly based on this CPS, which represents quality of the CPS. The system works in two separate stages. The first stage, called the static stage of calculation, evaluates quality depending on information that a CA has published about its practices of work. This information is obtained two documents: the Certificate Policies that the how to use the certificate, and the CPS, which describes how the CA manages its certificates. A server CPS answers the questions raised by the expert system depending on the CPS, which is published by CA in the form of a file XML. The importance of each question was determined by questioning experts in the field of PKI [12]. The second stage evaluates quality depending on the real behavior of CAs. This checking must be carried out by a third entity. This is obtained from an external audit entity that tests the PKI and publishes a signed certificate of audit [11].

4.3 The Friend CAs (Recommendations)

In order to enrich the computation of reputation of a foreign CA, CA of RP takes into account the recommendations of theirs friend CAs. These CAs might have collaborated in previous projects with the foreign CA and share their experiences with it. Consequently CA must maintain a list of friend CAs so that it can calculate the reputation of the foreign CA. This list can be built manually by the administrator of CA (for example protocols of co-operations between the two CAs).

4.4 Certificate Authority (CA) of RP (Calculating Reputation)

We have decided to make the CA of RP to play a central role in the calculation of the reputation for several reasons:

- There is already a trusted relation between a RP and its CA.
- RP may have no personal experience with foreign CA.
- CA of RP is more qualified than RP on the activity of certification and thus it can better evaluate other CAs.
- CA of RP can obtain scores provided by its clients who already worked with CA foreign.
- CA of RP can set up protocols of co-operation with other CAs, that we call friend CAs, who can give their feedback about the foreign CA.

A CA of RP has a more global view than the RP. It is able to determine any change of behavior of foreign CA more quickly than the RP itself.

Thus, there are two information sources from which the CA can provide a value of reputation to its RP.

1. Values provided by the other RPs who have already worked with the foreign CA,
2. Recommendations from friends CAs.

According to the criticality of the service, the administrator authorizes or not the RP (service) depending on the reputation and the quality score. That means the administrator must define thresholds for each service it has.

5 Reputation in the Context of X.509 PKI

Reputation being based on recommendations given by a group of entities, we should be able to calculate the score of reputation using one of the model mentioned above.

5.1 Calculation of Reputation

We have chosen belief model to calculate the reputation. Reputation depends on the recommendations from third parties and these recommendations could be more or less true. So we have doubt about the recommendation we receive. Belief model calculates reputation by considering the uncertainty of the received recommendations.

This model define a new term which is named opinion $w = (b, d, u, a)$ where b, d, u, and a represent respectively belief, disbelief, uncertainty and atomicity.
Where:

$$b + d + u = 1 \qquad where \qquad b, d, u \in [0,1] \tag{1}$$

According to the belief model, the value of atomicity is 0.5 when there are two possible output values (trustworthy, not trustworthy), consequently in our study the value of the atomicity is always 0.5.

When an entity *A* asks the recommendation of entity B about entity *X*, entity *A* computes the reputation of entity *X* depending on the recommendation of entity *B* using the recommendation operator as following:

The opinion $w_X^{AB} = (b_X^{AB}, d_X^{AB}, u_X^{AB}, a_X^{AB})$ expresses the opinion about *X* as result of recommendation from *B* where:

$$\begin{aligned} b_X^{AB} &= b_B^A b_X^B, \\ d_X^{AB} &= b_B^A d_X^B \\ u_X^{AB} &= d_B^A + u_B^A + b_B^A u_X^B \\ a_X^{AB} &= a_X^B \end{aligned} \tag{2}$$

Supposing that entity A receives many recommendations, in this case, it must be able to consensus all these recommendations using the consensus operator of the belief model as following:

The opinion $w_X^{A,B} = (b_X^{A,B}, d_X^{A,B}, u_X^{A,B}, a_X^{A,B})$ represents the opinions of *A* and *B* about *X* where:

$$
\begin{aligned}
b_X^{A,B} &= (b_X^A u_X^B + b_X^B u_X^A)/k \\
d_X^{A,B} &= (d_X^A u_X^B + b_X^B u_X^A)/k \\
u_X^{A,B} &= (u_X^A u_X^B)/k \\
a_X^{A,B} &= \frac{a_X^B u_X^A + a_X^A u_X^B - (a_X^A + a_X^B) u_X^A u_X^B}{u_X^A + u_X^B - 2u_X^A u_X^B}
\end{aligned}
\tag{3}
$$

$$k = u_X^A + u_X^B - u_X^A u_X^B, u_X^A = u_X^B \neq 0, u_X^A = u_X^B \neq 1$$

In our case A and B represent the same entity because entity *A* will have many opinions about the entity *X* and it will consensus these opinions by using this operator.

According to the belief model the probability expectation is given by the following formula:

$$E(w) = b + au \tag{4}$$

By using this equation we can compute the final result which will represent the reputation value.

5.2 Controlling the Influence of Old Experiences

When an administrator of a CA evaluates a foreign CA, it records the evaluation in its own database. When the organization reestablish a collaborative work with the same organization the old experience must affect also the final reputation, but this effect must be less important than the recent evaluation.

Tajeddine et al [14] have proposed an approach to combine the reputations with considering the time parameters.

The reputation value for each category of recommenders degrades with time and this value converges to a neutral value as time passes. Before using the reputation values, they must be adjusted as following:.

$$REP(t) = NV + (REP - NV) * e^{-(t-t_0)/\tau} \tag{5}$$

Where:

- NV: is neutral value.
- t_0: the of reputation computing.
- τ: decay factor that determines how quickly and slowly reputation value becomes invalid.

5.3 Example

We explain through an example how the reputation score is computed using belief model and how the administrator authorizes or not the RP to accept or to refuse the certificate.

Supposing that the relying party RP1 receives a certificate signed by $CA_{foreign1}$. The certificate authority of RP_1, RP_2, and RP_3 is CA_{RP}. CA_{RP} has two friend CAs, named

CA_{fr1} and CA_{fr2}. Because $CA_{foreign1}$ doesn't belong to the static trust domain of RP1, RP1 asks its CA if it can trust or not the certificate sign by $CA_{foreign1}$.

CA_{RP} calculates the reputation depending on the recommendations that it has already collected about $CA_{foreign1}$. Before collecting the recommendation the CA_{RP} must set up a structure containing the values of belief, disbelief, and uncertainty for all its recommenders as following:

Table 1. The recommenders list

Recommender Id	Belief	disbelief	uncertainty
RP_1	0.8	0.2	0
RP_2	0.85	0.15	0
RP_3	0.65	0.20	0.15
CA_{fr1}	0.65	0.35	0
CA_{fr2}	0.60	0.40	0

Supposing that recommenders have given the following recommendations, a triple (belief, disbelief, uncertainty) on $CA_{foreign1}$ and $CA_{foreign2}$ as following:

Table 2. Scores given by the recommenders

CA ID	Recommender ID	Given Score	date
$CA_{foreign1}$	RP_1	(0.72, 28, 0)	23-05-2007
$CA_{foreign2}$	RP_2	(0.5, 0.5, 0)	21-05-2007
$CA_{foreign1}$	RP_2	(0.6, 0.4, 0)	20-05-2007
$CA_{foreign1}$	RP_3	(0.75, 0.25, 0)	23-05-2007
$CA_{foreign1}$	CA_{fr1}	(0.56, 0.44, 0)	23-05-2007
$CA_{foreign1}$	CA_{fr2}	(0.75, 0.25, 0)	23-05-2007

CA_{RP} must revaluate the recommendations according to the belief, disbelief and uncertainty which are assigned to recommenders by using the recommendation operator of belief model (equation 2).

Table 3. Evaluation of given score by recommenders using belief model

Recommender ID	Result
RP_1	(0.576, 0.224, 0.2)
RP_2	(0.425, 0.425, 0.15)
RP_3	(0.4875, 0.1625, 0.35)
CA_{fr1}	(0.364, 0.286, 0.35)
CA_{fr2}	(0.45, 0.15, 0.4)

In order to calculate the reputation of CAforeign1 from these recommendations we use the consensus operator of the belief model (equation 3).

By the application of the consensus operator we obtain the following result:

$$w = (b=0.53, d=0.41, u=0.06)$$

Consequently, the value of reputation which will be returned to the RP will be calculated depending on equation 4 as following:

$$E(w) = 0.53 + 0.5 * 0.06 = 0.56$$

According to the criticality of service, the administrator of CA can decide to authorize or not the RP to use the certificate. The administrator can define set of policy rules that have the form:

If the reputation (<,>,=) X and quality (<,>,=) Y then authorize or not

6 Conclusion and Future Works

The objective of our work is to create a new trust model based on the reputation systems to interconnect efficiently PKIs in the context of virtual organizations. In this paper, we have presented the main drawbacks of the actual interconnection PKI models. We have analyzed the concepts of trust and reputation and summarized the calculation models that have been developed.

The reputation of a PKI is a quantifiable trust indicator with a discrete value that is related to the risk taken by an entity when it accepts a certificate from this PKI. Thus, trust on certificates is no longer binary and it becomes possible to differentiate the use of certificates according to service to access. For example, transactions on sensitive data imply higher trust indicators than transaction on public data.

Our future work will focus on formalizing the policy of acceptance of certificates that represents the risk accepted by the organization when the entities accept a certificate. For example, we have to provide user friendly mechanisms to control the impact of the reputation on the final decision.

From a technical point of view, it will be necessary to specify communications protocols between the entities implied in the calculation of the reputation to exchange their information (recommendations, reputation scores, etc). Finally, an important work will be to determine how these entities will vote and how to incite them to do it.

References

[1] Resnick, P., Zeckhauser, R.: Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System. In: Baye, M.R. (ed.) The Economics of the Internet and E-Commerce. Advances in Applied Microeconomics, vol. 11. Elsevier Science, Amsterdam (2002)

[2] Schneider, J., et al.: Disseminating Trust Information in Wearable Communities. In: Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K (September 2000)

[3] Jøsang, A., Ismail, R.: The Beta Reputation System. In: Proceedings of the 15th Bled Electronic Commerce Conference, Bled, Slovenia (June 2002)

[4] Abdul-Rahman, A., Hailes, S.: Supporting Trust in Virtual Communities. In: Proceedings of the Hawaii International Conference on System Sciences, Maui, Hawaii, January 4-7 (2000)

[5] Jøsang, A.: A Logic for Uncertain Probabilities. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 9(3), 279–311 (2001)
[6] Manchala, D.W.: Trust Metrics, Models and Protocols for Electronic Commerce Transactions. In: Proceedings of the 18th International Conference on Distributed Computing Systems (1998)
[7] Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project (1998)
[8] Gambetta, D.: Can We Trust Trust? In: Gambetta, D. (ed.) Trust: Making and Breaking Cooperative Relations, pp. 213–238. Basil Blackwell, Oxford (1990)
[9] Jøsang, A., Ismail, R., Boyd, C.: A Survey of Trust and Reputation Systems for Online Service Provision. Decision Support Systems (to appear, 2005)
[10] Chokhani, S., Ford, W.: Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. RFC 2527 (March 1999)
[11] Ball, E., Chadwick, D., Basden, A.: The Implementation of a System for Evaluating Trust in a PKI Environment. Evolaris, vol. 2, pp. 263–279. Springer, Wein (2003)
[12] Chadwick, D.W., Basden, A.: Evaluating Trust in a Public Key Certification Authority. Computers and Security 20(7), 592–611 (2001)
[13] Ølnes, J.: PKI Interoperability by an Independent, Trusted Validation Authority. In: 5th Annual PKI R&D Workshop. NIST, Gaithersburg (April 2006)
[14] Tajeddine, A., Kayssi, A., Chehab, A., Artail, H.: PATROL: a comprehensive reputation-based trust model. Int. J. Internet Technology and Secured Transactions 1(1/2), 108–131 (2007)

Business Service Management in a Service Oriented, Virtualized World

Vincent Kowalski

BMC Software, 2101 CityWest Blvd., Houston, TX 77042
Vincent_Kowalski@bmc.com

Abstract. Business Service Management (BSM) is now recognized as one of the most important attributes of a comprehensive systems management solution. Delivering relevant information to business decision makers is now a priority for management tool and application vendors. Recent trends in enterprise software have provided enterprises an opportunity to realize greater efficiencies and cost savings from their software investments. Additionally, these same opportunities have created new challenges for systems management software developers to keep pace with this dynamic environment. In particular, Service Oriented Architecture (SOA) and Virtualization are having a tremendous impact on how enterprises construct their production environments to leverage these efficiencies. SOA encourages enterprises to create more agile, reconfigurable and responsive software solutions, and Virtualization allows these same enterprises to realize cost savings by reducing the amount of hardware required, better utilization of existing hardware and reduction of energy consumption. SOA and Virtualization are now mainstream in the corporate computing environment and systems management tools must account for them in their complete solutions offering. These management solutions must provide substantial value to a given business and, SOA and Virtualization must be a part of any BSM solution. This paper provides the reader with the key ingredients for creating compelling BSM solutions in a Service Oriented, Virtualized enterprise.

Keywords: Business Services Management (BSM), Service Oriented Architecture (SOA), Virtualization.

1 Business Service Management

Traditionally, software companies that develop and market systems management tools have focused more on the traditional IT infrastructure than with the actual business constituency it serves. Vendors are now responding to more recent demands from the business for systems management to provide information that supports the actual business decision-making process. This is an important evolution for IT organizations. It is difficult to conceive of many, and in some cases any, business processes that do not make use of IT infrastructure. For example, if an Application Server that hosts a company's online shopping cart application is not available, the company would like to know in dollars, the cost of this downtime. Additionally, understanding

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 96–101, 2008.

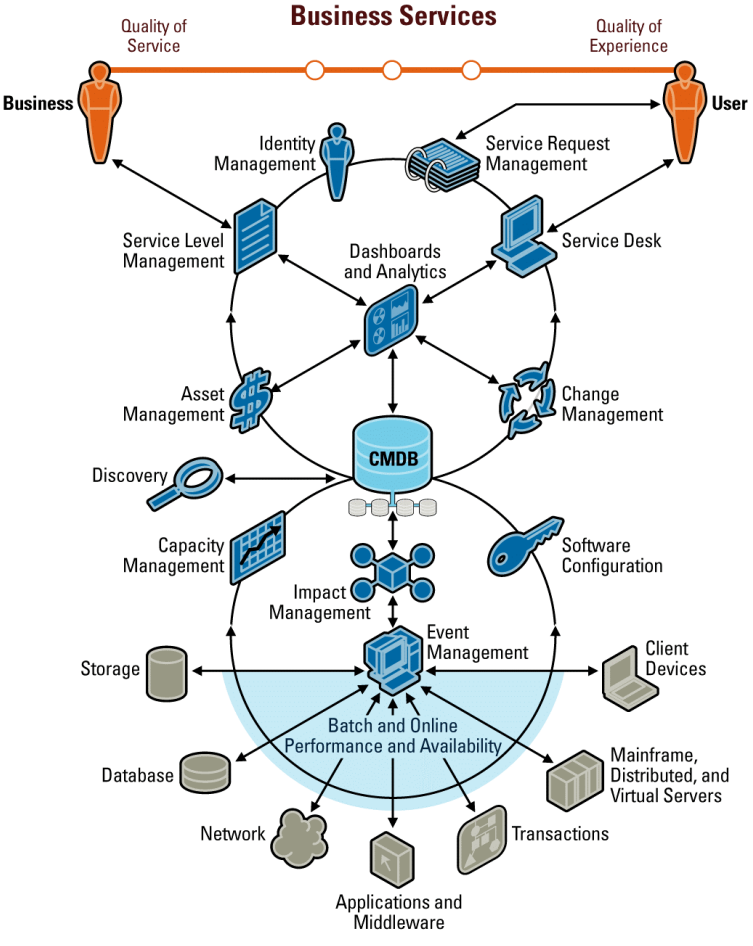


Fig. 1. Business Service Management (BSM)

which servers are most critical in their business, will allow decision makers to determine what additional investments (such as redundant servers, monitoring software, additional headcount) should be made to support the infrastructure.

Figure 1 above is a high-level, but comprehensive depiction of an end-to-end BSM Solution. A few key things to note about this diagram are the following. First, the heart of BSM is a Configuration Management Database (CMDB). As mandated by ITIL, a CMDB is the repository for Configuration Items in a given enterprise. Next, it is important to note where traditional systems monitoring and management tools fit into the overall scheme. Such tools depicted at the bottom of the diagram feed into tools that support ITIL-based activities such as Capacity Planning and Management, Impact Assessment and Management and Configuration Management. In the upper half of the diagram (the circle above the CMDB) are the tools that support Service Management tasks such as Service Desk, Change Management and Service Level

Management. These all are presented in an integrated Dashboard based view that, along with Business Analytics ultimately support the business decision-making that is the objective of BSM. Again, although we need an accurate view into the health and status of the underlying infrastructure, the main goal of BSM is to provide actionable data related to the business. One final thing to note: Figure 1 implies a high level of integration among the various components participating in a BSM solution. Such integration is well served by the use of Service Oriented Architecture (SOA) as will be discussed in the next section.

2 Service Oriented Architecture (SOA)

2.1 Conceptual Framework

As depicted in Figure 2[1], Service Oriented Architecture (SOA) is generally the use of Web Services technology to support the interactions of Service Providers (the blue octagon in the lower right hand vertex of the triangle) being discoverable and invokeable by Service Requestors (the red octagon in the lower left hand vertex of the triangle). To provide this discovery mechanism, a Registry (or Discovery Agency in the yellow cloud at the top of the diagram) may be used to publish descriptions of services. The technology to support Service Oriented Architecture includes:

- Web Services Stack
- Web Services Registry

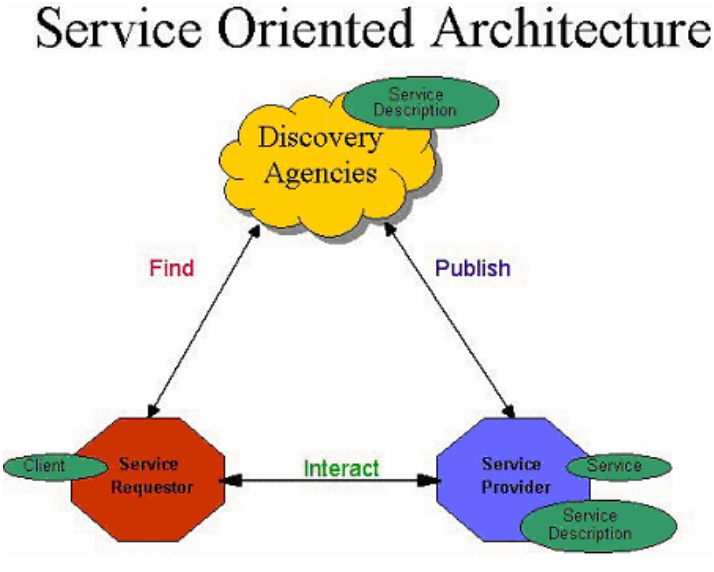


Fig. 2. Service Oriented Architecture (SOA)

The Web Services Stack provides the *interact* relationship between Requestor and Provider. The Web Services Registry enables Service Providers to *publish* services descriptions that can be discovered by potential Service Requestors using the *find* operation depicted on the line between Requestor and Discovery Agency. The technology to support these components is based on the following well-known standards:

- XML (the ASCII protocol upon which all Web Services are built)
- SOAP (for invoking services—the interact operation in the diagram)
- WSDL (for defining the interfaces of services)
- UDDI (for publishing and querying a Registry of services)

2.2 SOA in the Enterprise

Initial efforts related to SOA were in the area of Business-to-Business (B2B) applications. The thinking was that public Web Services Registries based on UDDI would expose a number of publicly available services that could be invoked by Web applications and brokers that would mediate for such applications. Conceivably highly dynamic applications could be constructed where providers could come on or off stream and be bound to applications based on dynamically determined criteria such a cost of service, quality or availability of service, content provision and so on.

This model found only limited adoption. Instead SOA found a home in corporate environments where standards-based and interoperable Web Services were seen as a key enabler to creating integrated solutions that include Web-based access to legacy applications and data sources and application-to-application communications. Enterprises that standardized on Web Services technology and SOA internally found they could achieve greater agility and higher degrees of data visibility and integration than were previously attainable.

As use of SOA in the corporate environment has increased so has the complexity of applications based on SOA. Thus, the need for managing SOA infrastructure has increased dramatically. Certain aspects of managing SOA infrastructure borrow from areas with well-known solutions such as Systems Monitoring and Service Level Management. SOA-based applications present two fundamental challenges that are not often addressed very well in the Systems Management arena:

- SOA applications are highly distributed
- SOA applications may be orchestrated (based on a representation of a workflow or process)

The first of these challenges is further complicated by the fact that Web Services technology can (and should) be implemented in such a way that you can make no assumptions about implementation language, underlying infrastructure or even provider of the Web Services components themselves. This characteristic of being *implementation agnostic* is a huge advantage to creating interoperable SOA applications, but complicates the management of those applications. Orchestration (when used) is a powerful tool for creating SOA applications that are configurable along the path of a given workflow. Such a workflow can be developed by a Business Analyst, not a Web Services specialist. Though such applications provide significant advantages to

developers and users of such applications, the management implications of such applications is only now starting to be understood. In summary, to get the requisite comprehensive and holistic visibility into SOA applications, both their highly distributed and process-oriented natures must be taken into account by any solution purporting to manage them.

2.3 SOA in BSM Solutions

Interestingly, SOA has become a tool for integration within the BSM space as well as by the Enterprises that BSM is intended to serve. Re-consider Figure 1. The set of components depicted in such a diagram would indicate a level of complexity that could be difficult to manage. In fact, with such a set of tools involved it is easy to see that this could degenerate to the N^2 phenomenon where the number of integrations is on the order of the square of the number of applications being integrated. SOA provides us a solution to this where we can move from a large set of point-to-point integrations to a more tractable hub-and-spoke architecture as depicted in Figure 3.

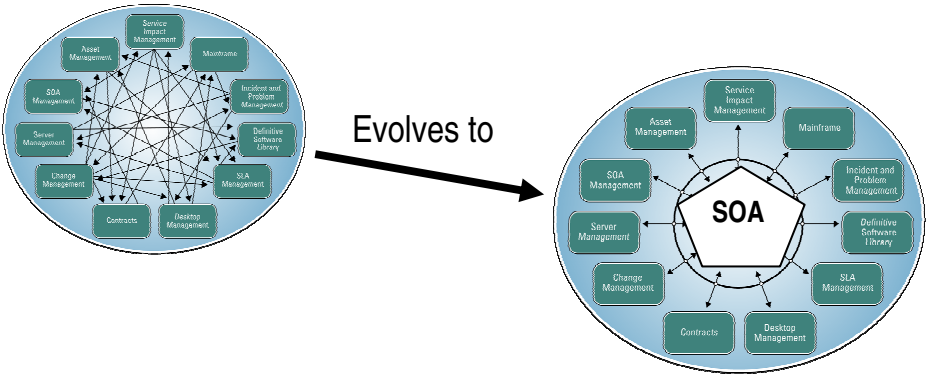


Fig. 3. Service Oriented Architecture (SOA) in BSM

3 Virtualization

Most people do not realize that virtualization has been a part of computing for many years. As early as 1969, IBM had produced the VM/CMS operating system intended to be a system able to run multiple concurrent operating systems. Despite this long history, virtualization has only recently become highly prominent part of IT infrastructure. A number of factors have contributed to virtualization realization, including the following requirements:

- Consolidation of IT hardware assets
- Maximization of under-utilized IT hardware resources
- Reduction of electrical energy consumption (i.e., Green Data Centers)

Virtualization presents a number of challenges in the management space that include:

- How to know the performance impact of virtualization before you virtualize

- Anticipating capacity issues and proactively responding to them before service is disrupted
- Reducing the risk of virtualizing
- Rapidly provisioning virtual servers without violating compliance with corporate policy or government regulations
- Eliminating over-provisioning and uncontrolled virtual server sprawl
- Managing and ensuring end-user performance of applications running on the virtual infrastructure

Virtual computing resources have some inherent differences from "hard" or physical computing resources in that they always have a relationship to a hard asset. Although virtualizing may not incur any hard cost other than disk space, their creation and destruction tends to be more dynamic and less disruptive than actual computing assets. Additionally, since virtual assets are not purchased and expensed as are physical assets, finding data about them in accounting, inventory or asset management databases is difficult. Given these differences, BSM solutions must make accommodations for virtualized resources in the following components:

- Discovery (to be able to discover virtual resources)
- CMDB (to model virtual configuration items)
- Capacity Planning and Management (to account for virtual resources in capacity calculations)
- Systems Monitoring and Management (to monitor virtual resources in capacity calculations)
- Service Level Management (to account for virtual resources in determinations of Service Level Agreement compliance)
- Change Management

4 Summary and Conclusion

It is vital for Systems and Service Management solutions to take into account the special requirements of SOA and Virtualized computing environments. To engineer and deliver comprehensive Business Service Management solutions that are relevant to today's Data Center, these additional requirements need to be incorporated into the designs of new versions of existing Systems and Service Management tools as well as serve as a catalyst for developing tools not yet in the marketplace. Finally, approaches such as SOA may provide challenges and opportunities with respect to the environments they support and should be leveraged to provide better-integrated total BSM solutions.

Acknowledgments. The author wishes to acknowledge Lisa McDowell of BMC Software who reviewed this paper and provided valuable feedback.

Reference

1. Champion, M., Ferris, C., Newcomer, E., Orchard, D.: Web Services Architecture W3C Working Draft, November 14, W3C (2002), http://www.w3.org/TR/2002/WD-ws-arch-20021114

A Control of a Mono and Multi Scale Measurement of a Grid

Imene Elloumi[1,2], Sahobimaholy Ravelomanana[1], Manel Jelliti[1], Michelle Sibilla[1], and Thierry Desprats[1]

[1] IRIT UMR 5505, Université Paul Sabatier, 31062 Toulouse cedex 9, France
[2] Higher School of Communication of Tunis, City of Communication Technologies 2083 Ariana, Tunisie
{elloumi,raveloma,jelliti,sibilla,desprats}@irit.fr

Abstract. The capacity to ensure the seamless mobility with the end-to-end Quality of Service (QoS) represents a vital criterion of success in the grid use. In this paper we hence posit a method of monitoring interconnection network of the grid (cluster, local grid and aggregate grids) in order to control its QoS. Such monitoring can guarantee a persistent control of the system state of health, a diagnostic and an optimization pertinent enough for better real time exploitation. A better exploitation is synonymous with identifying networking problems that affect the application domain. This can be carried out by control measurements as well as mono and multi scale for such metrics as: the bandwidth, CPU speed and load. The solution proposed, which is a management generic solution independently from the technologies, aims to automate human expertise and thereby more autonomy.

Keywords: CIM, Grid, A Dynamic Monitoring, QoS.

1 Introduction

The efficient use of grid resources by and for large applications is a challenging field of studies currently tackled by several research projects (EGEE1 project, the LHC project (LCG)2, GRID50003, etc). Indeed, during the development phase, controlling the intrinsic complexity of the applications and their suitability to grids and/or clusters is much needed. Within this phase, we will focus on the problems encountered in the grid implementation process. These are related to monitoring the communication network and its impacts. The monitoring component plays an important role in increasing reactivity and availability. Seen in this regard, it behooves to monitor a communication network, which enables the raising of network information towards lodge services by the grid and thus to get a pertinent analysis (traceability, reproducibility). However, some fundamental criteria for the surveillance of these large-scale

[1] EGEE: Enabling Grids for E-sciencE.
[2] LCG: grid project for the Large Hadron Collider (LHC) by CERN (the European Laboratory for Particle Physics).
[3] GRID5000: a French national wide experimental grid.

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 102–112, 2008.

distributed systems require a new generation and a revision of the software infrastructure of these grids, namely:

- The tolerance to failures, that is, how to react to different situations altering the expected requirements. Revising this criterion can continuously ensure the availability of the service and how to achieve this in a preventive and/or active manner [1].
- Ensuring a real or pseudo real time diagnostic.
- Optimizing applications and making infrastructures.
- Moving to a larger scale.

Putting aside the availability of the grid, inductive surveillance is the focal point in our present study. To this end, we must be able to observe, analyze, manage and control the evolution of state of all network components.

This paper falls into five sections. In section one, we introduce the related work on the capabilities of monitoring tools. Specifically, we will focus on the effectiveness and suitability and capabilities of present day monitoring tools in gathering real-time data required by grid environments. We will also demonstrate why the current monitoring tools are insufficient in grid environments, and delineate the non-existence of tools to combine multiple monitoring sources. It should be noted that the number of metric network handled has increased. This corroborates the argument that supervision can not be based on a single metric independently, hence, the need to combine several network metrics. In Section two, we present a standard model-based monitoring subsuming an integration issue and a multi-dimensional monitoring pattern. In Section three, we introduce and discuss our contribution regarding mono and multi-scale monitoring in the grid. In Section four, we present an application illustrating our proposed monitoring. In the last section, we conclude and present the future work.

2 Related Work

2.1 Shortcomings of the Existing Monitoring Tools

There are several tools that have been applied specifically to grid monitoring, which meets different needs such as the NWS, Fast, GloPerf, PingER, NetPerf, Performance Co-Pilot, etc [2] [3] [4] [5] [6]. No tool, however, exists for a rigorous and automatic real-time analysis, able to recover and rehabilitate incidents. These monitoring tools are not efficient when applied to grids. Measurement system must be able to:

- Identify and detect the availability of the grid in real time without altering its services,
- Allow for measurement, integration and aggregation of information, and insure information pertinence at all levels (from hardware to application levels) in real time,
- Not only does it collect the charge of the different resources, but rather it predicts their values in the near future by virtue of statistical methods.

System measurement could be done through the development of few intrusive methods of measurement. It stands midway between the quality of monitored information and the resources needs of the method used for measuring information.

2.2 Advanced Monitoring Needs

Four major needs were identified:

1. Masking heterogeneity and refine measures of information management, that is, to cope with the heterogeneity of the information format relative to monitoring the management entity. A more generic solution is mandated which might consist a view of information management. Such information model covers the fields of networks, homogeneous systems and applications. Thus, to get more pertinence, information aggregation is in order [7].
2. Inductive expertise: an inductive expertise will give the values of a metric network in any time in order to be informed about their evolution before a breakdown [8].
3. Prevention and Adaptation: one (or several) network component(s) in a critical state for some time (Time Outs) should be reported (trigger alarms) in order to sidestep any potential problem and thereby mitigate the likelihood of blackouts [9].
4. Diagnosis: diagnosis is a crucial component in monitoring system management. Put differently, it permits us to track and pinpoint the source of the malfunction of any network component. Further, it probes the nature and the impact of the malfunction on the entire system. The implication of this is to adequately correct or remedy any identified problem [10].

3 A Standard Model-Based Monitoring

3.1 Integration Issue

The integration of heterogeneous management data is fraught with problems: (1) communication protocol for collecting and retrieving management data is quiet insufficient and (2) the structure of collected data is heterogeneous (for examples SNMP SMI, Object oriented). The initiative WBEM[4] specified by the Distributed Management Task Force (DMTF) provides solutions to these two problems. First, CIM model[5] [11] allows data processing to be based on a unified data model (UML compliant). Then, the architecture of the CIM servers makes possible the storage of details about integration into the CIM repository which contains descriptions of classes.

Further, seen the implementation of CIM models by some platforms such as Open Pegasus [12] proves to be successful, our approach will endorse such to tackles the above-mentioned problems.

[4] Distributed Management Task Force, WBEM: Web services for Management (http://www.dmtf.org/standards/wbem/wsman).

[5] Distributed Management Task Force, Common Information Model (CIM) Standards (http://www.dmtf.org/standards/cim).

Through the core CIM classes we have three areas (Nework, Application and System) on which we worked. Fig.1 is an extract of the grid node from the CIM models. All the classes needed for monitoring each Grid components are present in CIM. The Processor class is a LogicalDevice representing the processor. It contains attributes CurrentClockSpeed and LoadPercentage. The LogicalPort class is also a LogicalDevice which represents the point of connection to a device. It contains attribute Speed and MaxSpeed which represent the Bandwidth (bandwidth port in bits per second).

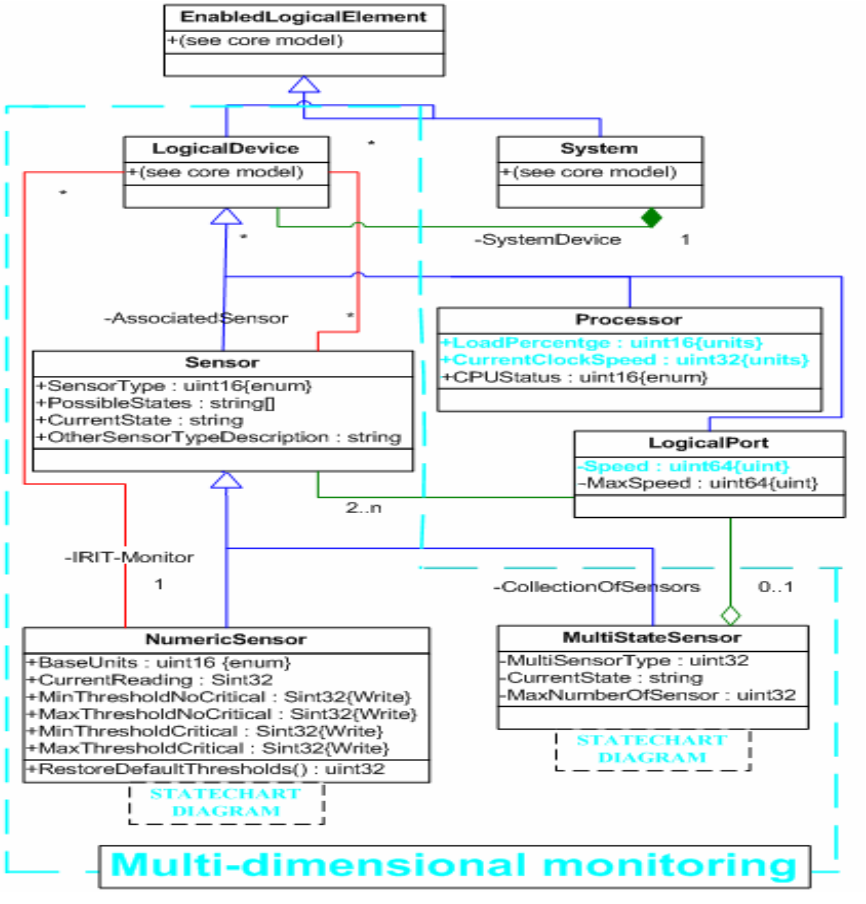


Fig. 1. Outsourcing Supervision of the Grid Network Nodes

3.2 A Multi-dimensional Monitoring Pattern

The DMTF has specified the concept of measuring equipment by the Sensor class. This class has the following properties (Fig.1): Sensor Type, Current State, and Possible States and Other Sensor Type-Description. A Sensor is associated with a Logical Device (for which the measure is undertaken) by the Associated Sensor dependency

[11]. In this standard specification a subclass Numeric Sensor contains some information regarding the levels of criticality, including: Min Threshold No-Critical, Max Threshold No-Critical, Min Threshold Critical, and Max Threshold-Critical. These levels represent the threshold limits of each state where the administrator will have the freedom to configure with respect to his own values (Fig.1). Thus, we have five states values for the properties.

In addition, the class has the Numeric Sensor Restore Default Threshold () method which makes it possible to fix or restore all default thresholds defined by the administrator (Fig.2).

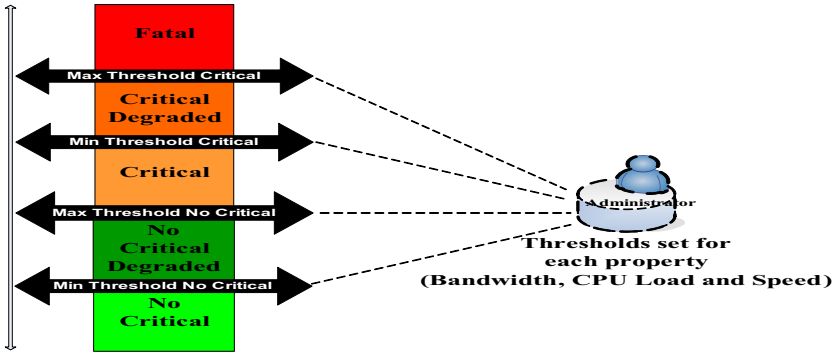


Fig. 2. Criticality Scale

The subclass Multi State Sensor of Sensor has been identified as a collection of Sensors. But the DMTF has not specified proprieties in the subclass Multi State Sensor. It may contain two or more Sensors symbolized by the aggregation Collection Of Sensors (Fig.1). It is also possible to add other thresholds.

3.3 Issues

The class Sensor is a way to monitor the evolution of any property of measurement type. The association AssociatedSensor cannot specify the property of Logical Device to which it relates. Moreover, the behavior associated with the Sensor is not clearly expressed. The class Multi State Sensor allows us to model a multi-scale measurement gathering sensors of the same type. For example, Sensors which measure the speed of CPU node in a grid computing or the bandwidth Logical Port of the grid, or a combination of different Sensors types (i.e., CPU load and speed CPU, etc). The class Sensor can meet the requirement needed to outsource the expression of the influence of some metrics on the state of a managed entity. To monitor a logical device property, the class Sensor will be deployed. More precisely, we will dwell on Logical Port and Processor. Related to this, the Speed properties (bandwidth) of a Logical Port and CurrentClockSpeed (CPU speed) and Load Percentage (CPU load) of a processor will be the main foci.

4 Dynamic Specification of a Multi-dimensional Monitoring

4.1 Expressing Influence and Control

The use of the previous management information model was specific to monitor Grid components, but it represents a static view of the managed entities [13]. The influences of changes (behavior) are not determined. In order to specify such behavior, the UML State chart diagrams are associated to the CIM classes. In our approach [13] [15], a State chart diagram can be associated with any object class derived from the CIM Managed Element class and the transitions triggered by events obtained from the CIM Indication class. States are situations where invariant conditions are checked.

4.2 Control of a Mono Scale Measurement

To specify the property monitored by a Numeric Sensor, the IRIT_Monitor dependency is added between the Logical Device and Numeric Sensor classes with the type and name of the monitored property (Fig.1). This dependency stems from the Associated Sensor dependency that already exists between the Logical Device and Sensor classes.

Our approach entails that a state chart diagram is associated to the Numeric Sensor class in order to specify the generic behavior of each Numeric Sensor associated to any property (Fig.3) [16]. We have identified a combination of possible states to describe a behavior that can be applied on any monitored property, namely: No Critical, No Critical Degraded, Critical, Critical Degraded and Fatal. A set of conditions (Table.1) controls the transitions from a state to another. Each (monitored) property change is reported by a CIM_Indication instance which triggers the evaluation of current state conditions. This applies to properties such as "gauge" to provide greater temporal logic to the behavior of devices [17].

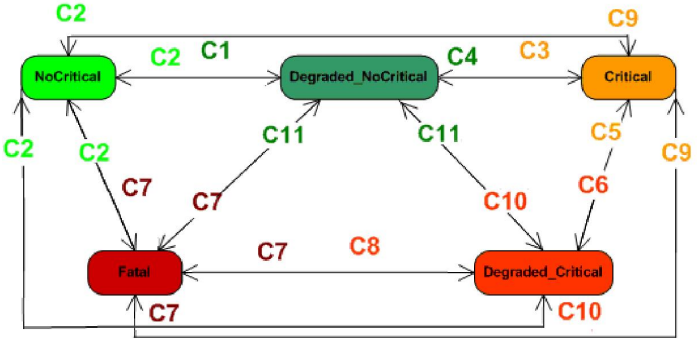


Fig. 3. Behavior Modelling: A StateChart Diagram Associated to the CIM_NumericSensor class (See conditions in the Table. 1)

Table 1. Condition Specified in the StateChart Diagram Associated to the Class CIM_NumericSensor

Symbol	Condition
C1	(Monitor.PropertyName.value) > (MinThresholdNoCritical)
C2	(Monitor.PropertyName.value) <= (MinThresholdNoCritical)
C3	(Monitor.PropertyName.value) > (MaxThresholdNoCritical)
C4	(Monitor.PropertyName.value) <= (MaxThresholdNoCritical)
C5	(Monitor.PropertyName.value) <= (MinThresholdCritical)
C6	(Monitor.PropertyName.value) > (MinThresholdCritical)
C7	(Monitor.PropertyName.value) > (MaxThresholdCritical)
C8	(Monitor.PropertyName.value) <= (MaxThresholdCritical)
C9	(MaxThresholdNoCritical) < (Monitor.PropertyName.value) <= (MinThresholdCritical)
C10	(MinThresholdCritical) < (Monitor.PropertyName.value) <= (MaxThresholdCritical)
C11	(MinThresholdNoCritical) < (Monitor.PropertyName.value) <= (MaxThresholdNoCritical)

4.3 Control of a Multi-scale Measurement

We have associated a state chart diagram with the Multi State Sensor class (Fig.4), which does not depend on the type of "N" Sensors. We have assigned quantifications to amplify the most critical in comparison with the least critical. The following section provides further explanatory information:

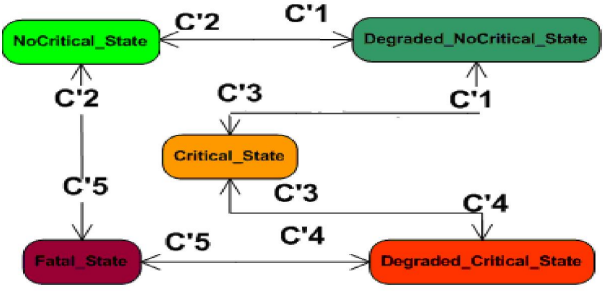


Fig. 4. Behavior Modelling: A StateChart Diagram Associated to the CIM_MultiStateSensor class (See conditions in the Table. 2)

These conditions are presented in the Table.2.

The values Ni ($0<i\leq5$) correspond to the percentages of having a state No-Critical or Degraded No-Critical or Critical or Degraded Critical or Fatal (for further details, see the following section).

Table 2. Condition Specified in the StateChart Diagram Associated to the CIM_ MultiState-Sensor Class

Symbol	Condition
C'1	(N2>=N1) and (N2>N3) and (N2>N4) and (N2>N5)
C'2	(N1>N2) and (N1>N3) and (N1>N4) and (N1>N5)
C'3	(N3>=N1) and (N3>=N2) and (N3>N4) and (N3>N5)
C'4	(N4>=N3) and (N4>=N2) and (N4>=N1) and (N4>N5)
C'5	(N5>=N4) and (N5>=N3) and (N5>=N2) and (N5>=N1)

4.4 Quantification

With respect to the state chart diagram associated to the Multi State Sensor class, we will try to quantify the states according to their degree of criticality. That is, we will cope with a collection of Sensors. To provide greater formalized behavioral logic, it is strongly recommended that this diagram be able to interpret the significance of the influence of the state. To this end, the scale of criticality should be used a basis for interpreting, a case in point, Fatal statement compared to less critical statements. In practice, this may be reflected, for example, by the followed expertise:

If the statements of Sensors that monitor the bandwidth of the three routers are: Critical, Critical Degraded and Fatal

Then the state of MultiStateSensor is Fatal.

$$fi: \leftarrow \begin{cases} a*x;\ i=1 \text{ if } x=Nnc \\ b*x;\ i=2 \text{ if } x=Nncd \\ c*x;\ i=3 \text{ if } x=Nc \\ d*x;\ i=4 \text{ if } x=Ncd \\ e*x;\ i=5 \text{ if } x=Nf \end{cases} \quad (1)$$

With:

Nnc: Number of no critical state (No-Critical State).

Nncd: Number of degraded no critical state (Degraded No-Critical State).

Nc: Number of critical (Critical State).

Ncd: Number of degraded critical state Degraded Critical State).

Nf: Number of fatal state (Fatal State).

And **(a + b + c + d + e) =100** with **a <b <c <d <e**, the function F(x):

$$F(x)= [\sum_{i=1}^{n} fi(x)]/100 \quad (2)$$

As a result, we find the rates relative to each state with predefined quantifications. They are calculated by the following reports (with (1) and (2)).

5 Application Example in a Local Grid

Suppose that we have a local grid containing 50 nodes.

So, we will have: 50 Processors, 50 Numeric Sensors to monitor the load (Load Percentage) CPU, 50 Numeric Sensors to monitor the speed (Current Clock Speed) CPU; and for Logical Ports, 100 Numeric Sensors associated to monitor the bandwidth (Bandwidth).

Hypothesis: For the (predefined) thresholds, it is assumed that the states of the associated Sensors are as follows (for example):

For Load CPU, current State of Sensors: (Nnc=10, Nncd=7, Nc=14, Ncd=15, Nf=4).

For CPU Speed, current State of Sensors: (Nnc=7, Nncd=11, Nc=16, Ncd=3, Nf=13).

For Bandwidth, current State of Sensors: (Nnc = 12, Nncd=27, Nc=32, Ncd=15, Nf=14).

And for example weight: a=10, B=15, C=20, D=25, E=30, however, a more specific study can be made for these weight values. After interpreting the statechart diagram and quantification, the resulting state analysis is presented in Table.3.

Table 3. A Multi-Dimensional State Analysis

	MSS (Bandwidth)	MSS (Speed CPU)	MSS (Load CPU)	MSS (MMS Load and MMS Speed CPU)	MSS (MMS Load and MMS Speed and MMS Bandwidth)
Fi(x)=	f1(x)=120 f2(x)=405 f3(x)=640 f4(x)=375 f5(x)=420	f1(x)=70 f2(x) =165 f3 (x)=320 f4(x)=75 f5(x)=390	f1(x)=100 f2(x)=105 f3(x)=280 f4(x)=375 f5(x)=120	f1(x)=0 f2(x)=0 f3(x)=0 f4(x)=1 f5(x)=1	f1(x)=0 f2(x)=1 f3(x)=0 f4(x)=1 f5(x)=1
F(x)=[f1(x)+f2(x)+f3(x)+ f4(x)+F5(x)]/100	19.6	10.2	9.8	0,02	0,03
N1=f1(x)/F(x)	6,122	6,862	10,204	0	0
N2=f2(x)/F(x)	20,663	16,172	10,714	0	**33,333**
N3=f3(x)/F(x)	**32,653**	31,372	28,571	0	0
N4=f4(x)/F(x)	19,132	7,352	**38,265**	50	**33,333**
N5=f5(x)/F(x)	21,428	**38,235**	12,244	50	**33,333**
∑Ni, 0<i<6	100	100	100	100	100
Multi-State Sensor	N2>Ni, (i !=2) Then his State is « **Degraded No-Critical** »	N5>Ni, (i !=5) Then his State is « **Fatal** »	N4>Ni, (i != 4) Then his State is « **Degraded Critical** »	N5=N4 Then his State is « **Fatal**»	N5=N4 and N5=N2 Then his State is « **Fatal** »

With,

N1: Percentage of having Sensors in states "No-Critical".

N2: Percentage of having Sensors in states "Degraded No-Critical".

N3: Percentage of having Sensors in states "Critical".

N4: Percentage of having Sensors in states "Degraded Critical".

N5: Percentage of having Sensors in states "Fatal".

Thus, an automatic update of composite elements according to the results of the analysis interpretation presented above in Table.3.

6 Conclusion

In this paper we have presented a solution to identify an inductive and dynamic monitoring for the multi views and multi purpose grid. The major advantage of our contribution is that the administrator will have relevant information on the state of composite network components of the grid (the five states of criticality: No-Critical, Degraded No-Critical, Critical, Degraded Critical and Fatal). This proposal can dynamically control of the CPU resources and the bandwidth to a cluster (with a control of a single measurement scale), a local grid and an aggregation of grids (with control of a multi-scale Measurement). These results could carry out different management operations such as lifting of alarms, repairing the system, etc., hence the inductive aspect of the monitoring. Therefore, whenever statements Sensors and Multi State Sensors are archived, it is necessary to analyze them to identify critical states that persist in setting Time Outs. The latter, once exceeded, needs to be triggered by

specific alarms either as a warning of the criticality state or a probability of a failure. Subsequently, we should give much attention to the control of every situation and the implementation of the necessary and appropriate corrections relative to each type of problem, for example run the intrinsic sequence of actions, when solved, will allow the system to acquire more autonomy management components. Hence, we can detect during the execution of a service "when" a resource has deteriorated and determine its state.

Fig.5 illustrates three cases of Forward alarme_1 on the state Fatal after a Time Out = T, alarme_2 on the state Degraded Critical after a Time Out=T and alarme_3 on the state Critical after a Time Out = T. SNMP Traps (SNMP agent sends an alarm to the Manager) are examples of these alarms.

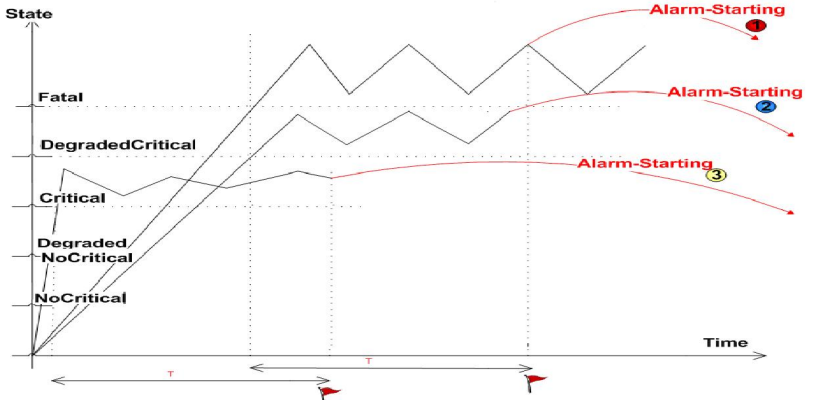


Fig. 5. Trigger Alarm

7 Future Work

The present study seems to meet the needs and requirements stated earlier. This study was implemented to control CPU resources and the bandwidth of the cluster, local grid and aggregate grid. However we think that we can apply this method to the other network resources. This can be done following the same approach by limiting the latency to a specified QoS and adjusting according to demand. We want to explore the behavior of the approach under different workload conditions and investigate the appropriate parameters (e.g. controller-Period) for better responses. Much work indeed needs to be undertaken to check other performance metrics such as network jitter and latency.

References

1. Andreozzi, S., Antoniades, D., Ciuffoletti, A., Ghiselli, A., Markatos, E.P., Polychronakis, M., Trimintzios, P.: Issues about the Integration of Passive and Active Monitoring for Grid Networks, Heraklion, Greece, Bologna, Italy (2005)

2. Wolski, R., Spring, N.T., Hayes, J.: The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. University of California San Diego and Tennessee Knoxville (2004)
3. Quinson, M.: Automatic Discovery of the characteristics and capabilities of a distributed computing platform. Ph.D. thesis. Ecole Normale Supérieure de Lyon (December 11, 2003)
4. Foster, I., CarlKesselmany, G.: A Metacomputing Infrastructure Toolkit (2004)
5. Blum, R.: Network Performance Open Source ToolkitUsing Using Netperf, Tcptrace and NISTnet (2003)
6. Performance Co-Pilot for IA-64 Linux User's and Administrator's Guide. Original publication. Supports the Performance Co-Pilot 2.3 release. Version 2.3 (December 2006)
7. Salehie, M., Tahvildari, L.: Autonomic Computing Emerging Trends and Open Problems. In: DEAS 2005, Dept. of Elect and Comp. Eng., University of Waterloo, St. Louis, Missouri, USA, May 21 (2005)
8. Bianchi, S.C.S., Jocteur-Monrozier, F., Sibilla, M., Marquié, D.: Improving behaviour control of complex systems. In: 12th Workshop of HP OpenView University Association, Porto, July 11-13 (2005)
9. Adi, A., Gilat, D., Ronen, R., Rothblum, R., Sharon, G., Skarbovsky, I.: Modeling nd Monitoring Dynamic Dependency Environments. In: IEEE International Conference on Services Computing SCC 2005, Orlando, July 11-15 (2005)
10. Geihs, K.: Towards Model-Driven Management. In: 12th Annual Workshop of HP Software University Association, Porto, pp. 237–245 (2005)
11. DMTF Application Working Group, CIM Schema and Specifications, Version 2.16 (2006)
12. Ravelomanana, S., CristinaSordela Bianchi, S., Joumaa, C., Sibilla, M.: A contextual GRID monitoring by a Model Driven Approach. In: Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, Gosier, Guadeloupe, pp. 37–43. IEEE, Los Alamitos (February 2006)
13. Sibilla, M., Barros de Sales, A., Desprats, T., Marquié, D., Steff, Y., Jocteur-Monrozier, F., Rivière, A.: CAMELEON: A CIM "Modelware" platform for distributed integrated management, Academic Alliance Contest winner. In: The Annual DMTF Developers Conference, USA, June 10-12 (2002)
14. Sibilla, M., Barros De Sales, A., Raynaud, Y., Jocteur-Monrozier, F.: An active CIM_Dependency pattern for a consistence service monitoring. In: Séoul, C., Boutaba, R., Kim, S.-B. (eds.) Network Operations and Management Symposium NOMS 2004, April 19-23, pp. 883–884. IEEE, Los Alamitos (2004)
15. Sibilla, M., Jocteur-Monrozier, F.: The supervision experience: Use models at the heart of systems. In: Dupuis, M. (ed.) Software engineering models and systems, ARAGO. Editions TEC & DOC, vol. 30, pp. 151–175 (May 2004)
16. Jocteur-Monrozier, F., Bianchi, S.C.S., Sibilla, M., Vidal, P.: Contrôle Dynamique des Systèmes Complexes par la Modélisation Comportementale. Revue Génie Logiciel, Hermès 72, 14–21 (2005)

A Resource Management Mechanism and Its Implementation for Virtual Machines

Zhigang Wang[1], Chuliang Weng[2], Yu Wang, and Minglu Li

Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
felixwang@sjtu.edu.cn[1] weng-cl@cs.sjtu.edu.cn[2]

Abstract. Recently, virtualization technology is growing in popularity. The virtualization solution such as Xen is becoming more and more mature, but the virtual machine management tools are not perfect yet, especially when it comes to the virtual machine cluster management.

This paper presents a friendly tool that we are developing for virtual machine cluster management named VE-manager. VE-manager can deploy and manage virtual machines in a cluster environment without considering the differences of the underlying virtualization technology. Benefiting from live migration technology, VE-manager can also achieve load balancing at the level of virtual machines.

1 Introduction

Virtual machines (VMs) can provide high security and user isolation which do not rely on operating systems mechanisms. On the other hand, several different VMs with different operating systems are able to run on a single virtual machine monitor (VMM) at the same time, making it possible to take full advantage of computing resource, as well as satisfy different execution environment requirement.

Virtualization technology also can be used in the cluster environment. Creating several virtual machines in the cluster can expend the physical cluster or create a new virtual cluster [1] [2] with private network. Even the virtual cluster can be created or destroyed dynamically when demanded [3]. Applying the virtualization technology into cluster environment introduces a new challenge for system management. If there are different VMMs such VMware, QEMU [4], Xen [5] and KVM in the same cluster, Centralized management will be even more difficult.

In order to meet previous mentioned demands, we design a solution (VE-manager) which can manage and monitor all the virtual machine in the cluster on the manager node with graphical user interfaces (GUI). It can also manage users and processes within a VM. Furthermore, we will monitor the workload of every VMM and achieve dynamic load balancing through live migration [6] of VMs.

The rest of this paper is organized as follows. Section 2 presents the overall architecture of our VE-manager solution. Section 3 describes the implementation

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 113–118, 2008.

details. Section 4 presents the current progress of our project. The paper ends with some conclusions and comments of future work.

2 Architecture

The overall architecture of VE-manager is presented in Fig.1.

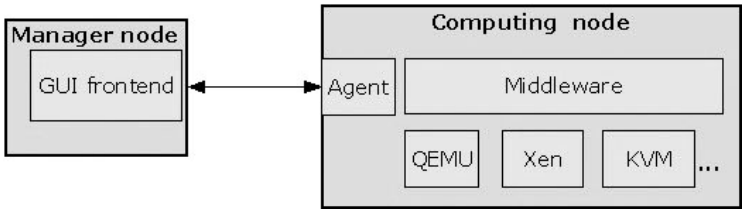


Fig. 1. The architecture of VE-manager

The overall solution is composed by the follow modules: (1) Virtual Machine Monitor (VMM). (2) Middleware. (3) Agent, and (4) GUI frontend.

The VMM, middleware and agent are deployed at each computing node of the cluster while the GUI frontend runs on the manager node.

The underlying VMMs are XEN, KVM or QEMU. In order to hide the difference of the underlying virtualization technology and provide uniform interface for higher levels, we introduce a middleware between the VMM and the agent.

Agent is a daemon which accepts the requests from manager, executes the commands through the interfaces provided by middleware and then returns the results to manager. It is a bridge between the manager and middleware.

GUI frontend on the manager node is the front-end of our solution which provides a friendly and effective GUI to administrator. All the virtual machine management tasks can be carried out on this front-end. It is a cross-platform application which can run on Linux, Windows and other popular operating systems.

3 Implementation Details

3.1 Agent and RPC

Agent is a bridge connecting the GUI frontend on the manager node and the middleware on the computing nodes. In order to send messages between manager and agent, we choose a lightweight but powerful remote procedure calls (RPC) protocol XML-RPC [7] as the communication method.

XML-RPC is a popular protocol which uses XML over HTTP to implement remote procedure calls. It encodes the procedure calls into XML, sends it to a server using HTTP, and then gets back the response as XML.

XML-RPC works in Client/server mode. In our solution, the client is the GUI frontend while the server is the agent. We have defined and implemented 29

function calls in the agent. The number will increase in the future if necessary. The function calls are not the direct map to the VMM middleware interface. Some of them are the combination of several VMM middleware APIs, so as to reduce the number of function calls and the network traffic.

3.2 GUI Frontend

The design of the GUI emphasizes convenience and effectiveness. Manager lists the VMMs in the cluster and the VMs under each VMM on the left side (as Fig.3 shows). This list is organized as a tree.

When user clicks one of the VMs, the details will be showed on the right side. The details are made up by two parts. One includes the running information, e.g. the state (running, booted or paused), the usage of CPU and memory etc. The other is the hardware information, such as the number of vCPUs, the mount of memory, the MAC and IP address of virtual network. Some hardware configuration can also be adjusted on the fly.

Creating a new virtual machine can easily be accomplished. A number of wizard dialogs are used to require the user to set configuration. After all wizard dialogs are finished, the manager will collect all the configure information of the new VM, and convert it to XML as the parameter of createVM() RPC. After RPC finish its job, a new VM will be added to the VMs list.

In order to make it platform independent, the GUI frontend is implemented in python while the UI is constructed with pyGTK. Since the frontend only depends on python and pyGTK (XML-RPC is a part of the standard python library, so there is no especial dependence on the XML-RPC), the manager can work well on both Linux and Windows without making any changes.

3.3 Middleware

It is not surprising that different VMMs provide different management interfaces. What is more, there are some differences in different versions of the same VMM. So a middleware is used to handle these differences.

The middleware of VE-manager is based on the libvirt [8] at present. Libvirt is an excellent library to interact with the different VMMs. It supports Xen, QEMU, KVM, LXC, openVZ currently. It provides a stable API to applications on higher levels.

3.4 System Management

The management in a virtual environment is divided into two parts:

The first part – the system management within a VM, e.g. creating a new user or monitors the running processes in a VM.

The second part – VMM is involved in this part, e.g. getting the usage of CPU, memory of a VM or creating a new VM.

Since the former requires information within a VM while the latter one requires information from a VMM, we discuss them separately.

System management within a VM: the system controlling and monitor within the VM are similar to a normal system management where VM can been assessed via virtual network. So the standard monitor system can been used.

System management within a VM includes: (1) User and user privilege. (2) Processes. (3) Services. (4) Files and file share. (5) System update.

VM management within a VMM: In order to support different kinds of VMMs, a middleware is introduced in VE-manager. VMM and VM level managements are carried out through a uniform interface provided by the middleware, not the VMM directly.

VM management includes follow items: Pause or resume the VM, boot or shutdown the VM, get or set the usage of memory or vCPUs etc.

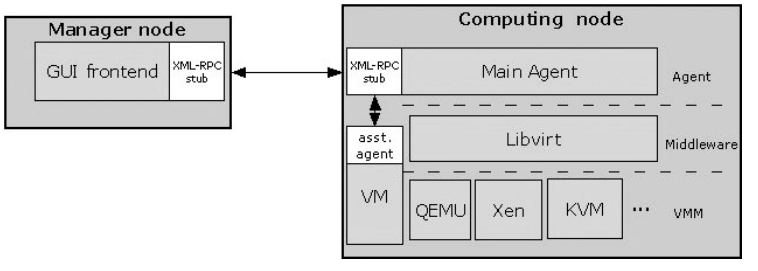


Fig. 2. More detail of VE-manager

3.5 Live Migration and Load Balancing

Live Migration can transfer of a virtual operating system from one VMM to another with minimal down time. This technology makes it possible to do dynamic load balancing in a virtual machine cluster. Manager collects the workload information of each VMM in the cluster, such as the usage of CPU, memory and network. Given these information, the VMs on the overloaded VMMs can be migrated to the idle ones by the manager.

3.6 Security

Because the messages transported over HTTP can be easily sniffed, it is dangerous to connect a VMM in a unsafe environment. So VE-manager has an option to use SSL connections to send XML-RPC messages if want to enhance security.

4 Current Progress

According to the implementation details, we have developed a prototype of VE-manager. The screenshot of current GUI frontend is presented as Fig.3.

The agent has been implemented completely. And the manager has following features currently.

(1) Get the running status of the VMMs and VMs, and show all the information on the GUI.

(2) User can set the number of vCPUs and the amount of memory for a specific VM via the GUI.

(3) User can reboot, pause and resume a VM via the GUI.

(4) Deploy a new VM via the GUI frontend.

(5) System management within a VM, e.g. user and processes management.

The following features are under development: Migration of VM and load balancing, we will accomplish these feature soon.

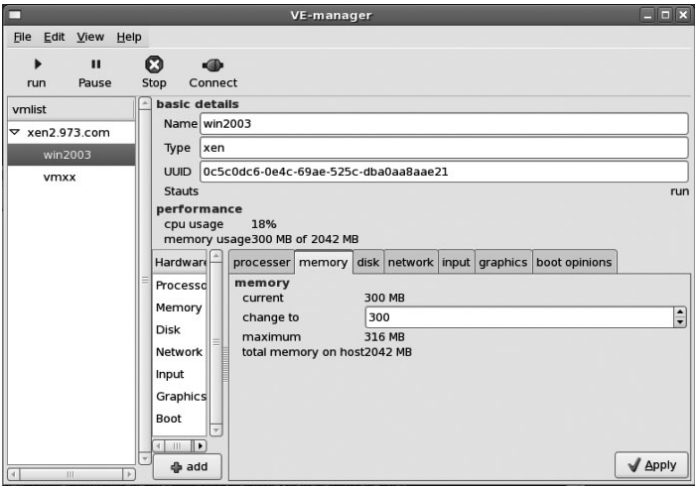


Fig. 3. The interface of current manager

5 Conclusions and Future Work

We have described a solution of virtual machine cluster management named VE-manager in this paper. The goal of VE-manager is to manage all the VMMs, the VMs running on the VMMs and system within the VM effectively and friendly in a cluster environment. We also consider the deploying of VMs, migration of VMs and load balancing.

Comparing with other virtual machines management solution [9] [10], VE-manager has following advantages:

(1) GUI frontend is designed as a thin client. It does not rely on any platform-dependent library. Furthermore, we implement it with platform-independent programming language, so the GUI frontend can run on different platforms without modifying any code.

(2) Because the GUI frontend does not depend on the VMM middleware, if our middleware is based on another library in the future, we only need to implement another agent to co-work with the new middleware. And the GUI frontend will work as usual.

(3) We consider the management solution in the cluster environment at the beginning of our design. So we discuss the deploying, migration of VMs and load balancing in this paper which are import for virtual machines cluster management.

In a cluster environment, fault tolerance is also an import topic, include: (1) How to detect the abnormal on a VMM and migrate all the VMs on this VMM to another VMM before it break down. (2) Failure recovery of VMs.

On the other hand, storage pool, network and desktop management is also the important topic in virtual machines cluster management, we will consider all of these in the future.

Acknowledgements

This work was supported by National 973 Basic Research Program of China under grant No.2007CB310900, Huawei Science and Technology Foundation, and National Natural Science Foundation of China (No.90612018, No.90715030, No.60503043).

References

1. Krsul, I., Ganguly, A., Zhang, J., Fortes, J.A.B., Figueiredo, R.J.: VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. In: Proceedings of the 2004 ACM/IEEE conference on Supercomputing (November 2004)
2. Zhang, X., Keahey, K., Foster, I., Freeman, T.: Virtual Cluster Workspaces for Grid Applications, TR-ANL/MCS-P1246-0405 (April 2005)
3. Emeneker, W., Jackson, D., Butikofer, J., Stanzione, D.: Dynamic virtual clustering with xen and moab. In: Proceedings of ISPA Workshops: Workshop on Xen in HPC Cluster and Grid Computing Environments (XHPC), pp. 440–451 (2006)
4. Bellard, F.: Qemu, a fast and portable dynamic translators. In: USENIX 2005 Annual Technical Conference, Anaheim, CA, USA (April 2005)
5. Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Pratt, I., Warfield, A., Barham, P., Neugebauer, R.: Xen and the art of virtualization. In: Proceedings of the ACM Symposium on Operating Systems Principles (October 2003)
6. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 2005) (May 2005)
7. UserLand Software, Inc.: XML-RPC, http://www.xmlrpc.com/
8. libvirt, http://libvirt.org/
9. Virtual Machine Manager, http://virt-manager.et.redhat.com/
10. Vallee, G., Naughton, T., Scott, S.L.: System management software for virtual environments. In: Proceedings of the 4th international conference on Computing frontiers (May 2007)

VNEC - A Virtual Network Experiment Controller

François Gagnon*, Tomas Dej, and Babak Esfandiari

Carleton University
fgagnon@sce.carleton.ca

Abstract. This paper presents VNEC, a tool to specify and execute network experiments in a virtual environment. The user first formulates the network topology and then provides the tasks that should be performed by the computers together with their execution. Next, VNEC initializes the environment by powering up and configuring the virtual machines to match the desired network topology. Finally, commands are dispatched to the right virtual machines in the specified order. VNEC provides an environment for several types of research experiments such as virus propagation patterns and reactions of different targets against a given attack.

1 Introduction

Many interesting, and sometimes essential, network-related experiments are costly, time-consuming, and risky. For example, unleashing viruses to study their spreading behavior allows to improve defense systems, but building physical test networks is expensive and networks are hard to maintain after a virus infection. Performing a security penetration test on a sensitive bank server can provide good insights on potential risks, but can create a security breach on the server. Deploying a new application in a production environment is usually hazardous, since it is hard to foresee negative impacts, e.g., downtime of the production environment due to an error in the application.

In several cases, running these experiments in a virtual environment is sufficient (e.g., virus study and penetration test) or can serve as a dry-run to avoid costly breakdowns when performing them on the physical network (e.g., application deployment).

There are a few existing tools, like VMWare [10], which offer the necessary components to set up a virtual environment where experiments can be executed. However, one has to either setup and execute the experiment manually or develop his own controller. This controller is responsible for two main operations: configuring the network and dispatching tasks to the virtual machines (VM) in a specific order. Developing such a controller can be quite complex, especially dispatching tasks to the VM.

* Corresponding author.

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 119–124, 2008.

In this paper we introduce *VNEC* (Virtual Network Experiment Controller), a tool that allows the user to easily specify and execute network experiments. An experiment consists of a network topology containing several computers as well as a sequence of tasks to be executed in a particular order by the computers of the network. Once the specification phase is completed, the experiment can be run in a VMWare virtual environment. VNEC provides a graphical user interface to specify the topology and the task workflow for an experiment. Moreover, it takes care of configuring the virtual machines and dispatching tasks to the corresponding VM.

The main applications driving the development of VNEC are risky network experiments (e.g., virus propagation and attack scenarios) and data collection experiments (e.g., operating system fingerprinting and collecting target reactions against attacks). These applications dictate two important requirements on the solution:

- Executing the experiment in a confined environment (to avoid infecting physical machines with viruses).
- Allowing the use of a wide variety of guest operating systems (to collect data across a wide spectrum of targets).

The rest of the paper is structured as follows: Section 2 discusses related work in the area of virtualization controller. Section 3 presents VNEC and its different specification and configuration steps. Finally, a discussion and some pointers towards future work will conclude the paper.

2 Related Work

We decided to use VMWare virtualization technology because it supports a very large number of different guest operating systems (OS). Other alternatives included VirtualPC [2], QEMU and Xen [6] (Chapter 14), UML (User Mode Linux) [5], and Basilisk [1], but they typically support much fewer guest OS.

Some tools similar to VNEC already exist, but none of them are as generic. Of particular interest is the following: VMWare VIX API [9], HP SmartFrog project [8] and the VNUML tool [7].

VIX is an API developed by VMWare to provide some control over the virtual machines (e.g., power up, file copying, command execution). Unlike VNEC, VIX is not a controller; it simply provides an API helping one to build a custom controller. More importantly, the VIX functions allowing communication with a VM work only if the the VM is running a recent Linux or Windows OS[1]. This is a major limitation, since commands cannot be dispatched to all virtual machines.

The SmartFrog project is a general framework to manage distributed systems. To use VMWare technology, SmartFrog uses a Java wrapper on VIX, thus it can act as a controller for virtual networks. Since SmartFrog relies entirely on VIX, it inherits its limitations.

[1] These functions requires the VMWare tools to be installed on the VM, and only recent Linux or Windows VM can install them.

The VNUML tool (Virtual Network User Mode Linux) is a controller for the UML virtualization technology. However, since UML supports only Linux guest OS, VNUML is more limited than SmartFrog.

VNEC is an extension of a custom tool developed at the Communication Research Center [4] in Canada. The earlier version had no graphical interface and commands could only be dispatched to Windows and Linux virtual machines. The need to dispatch commands to all virtual machines was the principal motivation to develop VNEC.

There could be other undocumented ad hoc solutions to the creation of virtual network environments. However, we are not aware of a solution combining both network and task specifications that would work for most guest operating systems.

Since VNEC uses VMWare, every OS supported as a guest by VMWare can be used in the experiments. However, unlike VIX, VNEC is able to dispatch commands to every virtual machine not just to those running Linux or Windows. This is important for many data gathering experiments such as studying the reaction of different OS against a given attack and gathering OS fingerprints.

3 Virtual Network Experiment Controller

VNEC has three modules, each one being detailed in corresponding sections below:

- Network specification (which network topology and VM to use)
- Task workflow specification (which tasks are to be executed by the VM and in which order)
- Experiment execution (configuring the VM and dispatching commands in the desired order)

3.1 Network Specification

Figure 1 depicts the network specification environment of VNEC. The network specification phase consists of three parts:

- Creating the set of components (computers, hubs, and routers) using drag-and-drop.
- Specifying the network topology by connecting the components according to some rules (e.g., computers cannot connect to computers, each computer has at most one connection).
- Associating each computer to a virtual machine (from a set of pre-existing VM) and providing network settings (e.g., IP address) to the computer[2].

In Figure 1, the user connected five machines (in clockwise order from top-right: FreeBSD NetBSD, OpenBSD, Linux, and Windows) using two hubs and a router. Routers are implemented using a dedicated VM, while hubs are the default behavior on a VMWare VMNet. By clicking on a computer icon, it is possible to overwrite the default configuration (ip address, snapshot, etc.).

[2] Default values can be used for network settings.

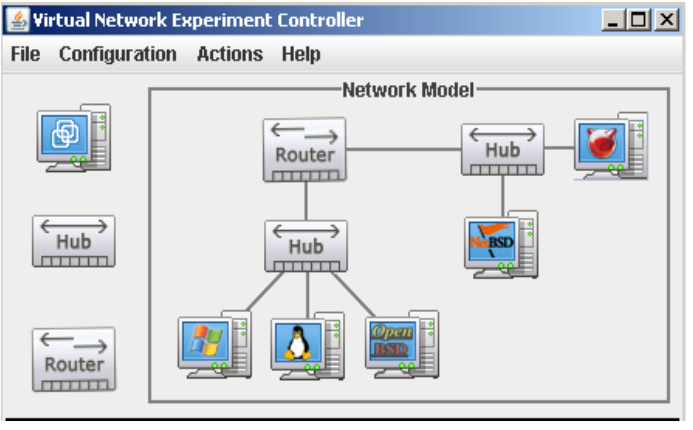


Fig. 1. Snapshot of VNEC - Network Specification

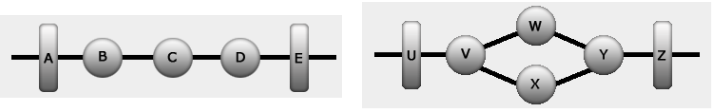
3.2 Task Workflow Specification

The task workflow fulfills two roles: it allows the user to indicate which tasks should be executed by the virtual machines and to specify the order of execution. The task workflow is a directed acyclic graph [3] with a single source and a single sink[3] where each node corresponds to a task (Figure 2). The semantics of such a workflow is that a task is to be executed when all its parent tasks are completed.

A task is either a *command task* or a *control task*. Command tasks are executed by a virtual machine (e.g., *create file*, *delete file*, *kill process*, *open telnet connection*, *open web browser*, etc.), while control tasks are performed by the controller to modify the state of a virtual machine (e.g., *power on*, *shut down*, *take snapshot*, *revert to snapshot*, *clone*, etc.). One must assign a task to each node; this can be done in a custom way by providing the set of command strings that must be executed or by selecting and configuring a predefined command structure. A command structure requires some parameters; for instance, the delete file command structure requires a file name. Each command structure will be automatically converted into command strings at run time by the VM. For instance, the command structure to delete the file "`name`" would translate to "`rm -f name`" on a Linux VM and to "`del name`" on a Windows VM.

A task workflow reads from left to right; a circle represents the execution of a task by a single given VM, while a rectangle stands for the execution of a task by all the virtual machines in the network. For instance, the task workflow shown in Figure 2(a) starts with task A which is executed by all VM (say power on). Once task A is completed, task B, is performed by a single VM (say Linux starts recording traffic). Afterwards, task C (say OpenBSD launches a specific attack against Windows), then task D (say Linux stops recording the traffic) are executed in sequence. Finally, E (power off) is applied on all virtual machines.

[3] A source (resp. sink) is a node with no incoming (resp. outgoing) edges, i.e., a root (resp. leaf).



(a) Linear Task Workflow (b) Non-Linear Task Workflow

Fig. 2. Examples of Task Workflows

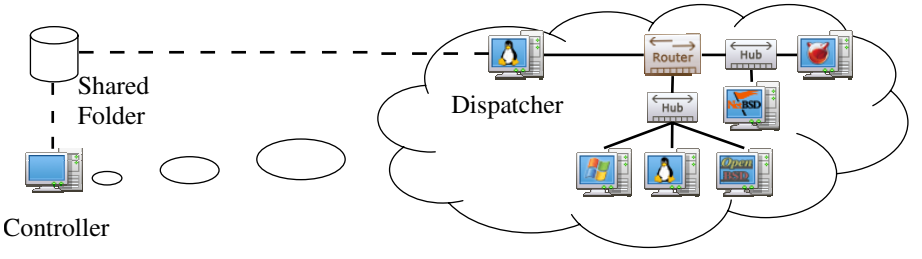


Fig. 3. VNEC Communication Architecture (based on the network of Figure 1)

A task workflow does not have to be linear as displayed in Figure 2(b). Once task V is completed, both tasks W and X begin concurrently. Task Y will start only after both W and X are completed.

3.3 Experiment Execution

Once both the network and the task workflow are specified, the experiment is ready to be launched. To be able to dispatch commands to any virtual machine, we implemented two mechanisms to communicate with the virtual machines: through shared folders (using the VMWare shared folder feature) and through remote method invocation (using Java RMI).

The VMWare shared folder feature allows the physical host and the virtual machines to access a common folder. A VM can simply look for a specific file in the shared folder, parse it and interpret its content. To dispatch a command to a specific VM, the controller creates a file representing the command and place it on the shared folder to be processed by the corresponding VM. This process is both simple and safe. However, the shared folder feature is available only for some virtual machines (recent versions of Windows and Linux). To circumvent this limitation, we developed a second mechanism.

In the second mechanism, each virtual machine is running the server side of a JAVA RMI application; another VM can call the function `execute(Command c)` on the server. Unfortunately, the controller (the physical machine) cannot communicate directly with the VM through the network (to avoid virus propagation out of the virtual environment). To address this problem, we include a Linux VM dedicated to dispatching commands, the *dispatcher* in Figure 3. The controller tells the dispatcher, through a shared folder, which task should be executed by which virtual machine (see dashed lines in Figure 3). Then, the dispatcher for-

wards the task to the corresponding VM using the RMI mechanism (see plain lines in Figure 3).

The two mechanisms are used together to dispatch commands to any VM while still providing a strong containment of the virtual environment.

4 Conclusion and Future Work

VNEC is a generic and powerful tool. It can handle several types of experiments with its flexible specification interface. Moreover, it can use a wide variety of guest operating systems thanks to VMWare virtualization technology, and it can dispatch commands to all virtual machines. Although VNEC is developed with specific applications in mind, i.e., virus propagation and data collection, it will be helpful in several other domains.

We will consider many avenues in the near future. Of particular interest are:

- Develop a module to save and load experiment specifications (probably in XML). This would also serve as a scripting language, an alternative to the graphical interface.
- Provide more powerful constructs in the task workflow specification (conditionals, loops, etc.).
- Analyze the workflow to detect basic anomalies, e.g., if a VM is supposed to execute a command task before it is powered on.
- Monitor the execution of the experiment to detect possible anomalies, i.e., if a virtual machine crashes after an attack, we cannot expect it to execute its subsequent tasks.
- Use multiple virtualization technologies simultaneously, providing a bridge from one virtual environment to another.
- Deploy the experiment across multiple physical machines, while keeping a single controller. This would allow bigger experiments to be run easily without relying on expensive servers.

References

[1] Bauer, C.: Basilisk Homepage, basilisk.cebix.net
[2] Carwell, R., Webb, H.: Guide to Microsoft Virtual PC 2007 and Virtual Server 2005. Thomson Course Technology (2007)
[3] Chartrand, G., Lesniak, L.: Graphs & Digraphs. Chapman & Hall, Boca Raton (2004)
[4] Communication Research Center. CRC Homepage, http://www.crc.ca
[5] Dike, J.: User Mode Linux. Prentice Hall PTR, Englewood Cliffs (2006)
[6] Golden, B.: Virtualization For Dummies. For Dummies (2007)
[7] Lurua, K., Khanvilkar, S.: Virtual Networking with User-Mode Linux. Linux for you (Networking Expert) (January 2005)
[8] Goldsack, P., et al.: SmartFrog: Configuration and Automatic Ignition of Distributed Applications. In: HP OVUA 2003 (2003)
[9] VMWare. Introducing the VIX API. VMWorld 2006 (2006)
[10] Warren, S.: The VMWare Workstation 5 Handbook (Networking and Security). Charles River Media (2005)

Update on System Virtualization Management

Michael Johanssen

IBM Deutschland Research & Development GmbH
D71032 Böblingen, Germany
johanssn@de.ibm.com

Abstract. The DMTF System Virtualization, Partitioning and Clustering (SVPC) Working Group is working on standardizing management interfaces for the management of system virtualization platforms. This paper outlines the modeling approach taken by the SVPC working group, briefly presents the central management profiles already developed.

1 Introduction

The availability of new virtualization specific hardware capabilities for the x86 hardware platform (AMD-V™ [1], Intel® VT [2]) triggered the development of new system virtualization solutions in the recent past [3, 4]. Furthermore a number of well established system virtualization solutions already existed for years on various hardware platforms (e.g. [5, 6, 7, 8]).

The diversity of management interfaces for all these system virtualization solutions generated the demand for a unified set of basic system management functions, like for example for creating or destroying virtual systems, for starting or stopping virtual systems, or for adding / removing virtual resources to / from virtual systems. System virtualization platforms should be enabled to expose basic management functions to a variety of management client applications without requiring client specific management agents for every platform. Management clients should be enabled to perform basic management functions for a variety of system virtualization platforms without being required to know about solution specific management interfaces.

The Distributed Management Task Force (DMTF) has a rich history in developing industry standard system management interfaces for a variety of management domains. Starting in 2005, the DMTF System Virtualization, Partitioning and Clustering (SVPC) working group set out for developing management standards for system virtualization. Following a set of essential design goals and applying well established CIM concepts like the CIM schema and management profiles, the group delivered an initial set of management profiles defining standardized management functionality for central system virtualization elements.

2 Model Considerations

This chapter provides information about structure and evolution of the set of management profiles created by the SVPC working group. The information is based on

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 125–134, 2008.

information presented by the CIM System Virtualization Model White Paper [10] and respective DMTF management profiles referenced in the text.

2.1 Design Goals

A number of central design goals were established as a guideline; these goals are detailed below.

Generality aims at supporting all types of system virtualization. For example, the management interface inherently should enable the management of specialized types of virtual systems like virtual switches or virtual storage subsystems.

Exposure of capabilities aims at enabling a client to adapt to a wide range of potential platform capabilities. For example, clients should be able to detect whether a platform supports the modification of virtual resources.

Definition of reasonable defaults aims at enabling less sophisticated clients to exploit complex functionality. For example, clients should be able to create virtual systems without being required to specify all virtual system components at a high level of detail.

Built-in extensibility aims at enabling incremental growth of the model suite. For example, implementations providing functionality beyond that exposed through the standardized set of interfaces should be enabled to expose management functionality for that additional functionality in a homogeneous and consistent way.

Leverage existing work from other DMTF working groups aims at simplifying and unifying systems management as a whole. For example, clients capable of managing real systems through interfaces specified by the Server Management Architecture for Server Hardware (SMASH) suite of management profiles [11] should be enabled to manage respective aspects of virtual systems using the same set of interfaces.

2.2 CIM Concepts

Generally the Common Information Model (CIM) allows the definition of management interfaces for managed environments. The basic elements provided for the definition of these management interfaces are the CIM Schema and DMTF management profiles.

CIM schema. The CIM schema [12] builds the foundation of CIM based management models. It is a hierarchically organized set of classes that are defined conformant to the CIM meta schema.

The CIM schema is maintained by DMTF using a formal change process that ensures a steady evolution of the schema. During the development of the system virtualization model several new classes were added to the CIM schema, and existing classes were amended.

CIM classes are defined as a collection of instances that all support a common type. A CIM class defines the type through a set of properties and methods.

CIM associations are a special type of class where properties may be defined as references, enabling the definition of relationships between classes. Associations may have

additional non-reference properties, allowing to define different flavors of relationships.

CIM indications are another special type of class that are defined to report the occurrence of events. Lifecycle indications report the creation, deletion or modification of CIM instances, process indications reports events that originate outside of the CIM model.

CIM instances are defined as a unit of data that is a set of property values and can be uniquely identified by a key. Each instance provides values for the properties as defined by its defining class.

CIM instances represent real world objects. While it is conceptually intriguing to think of CIM instances as persistent objects similar to data records in a database, it is important to realize that CIM instances in fact are only temporarily created as they are requested by a client. When a client requires access to a particular CIM instance, the responsible class implementation accesses the managed environment that exposes the real world object represented by the CIM instance and dynamically obtains all the information required to present the CIM instance to the client. In other words, CIM models provide clients with a database like view of reality, but lastly implementations of CIM classes are required to dynamically construct CIM instances whenever clients require access.

Generally CIM classes alone do not provide sufficient context for their use in particular management contexts; that context is provided by management profiles.

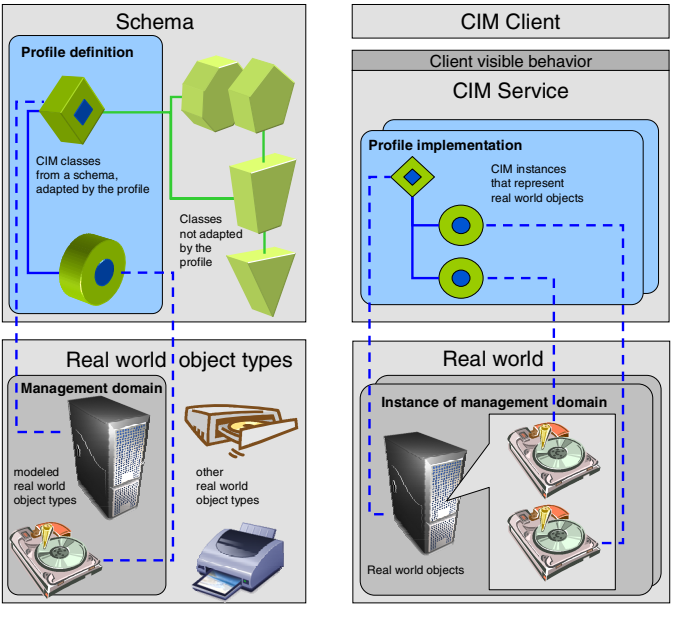


Fig. 1. Profile and management domain

2.3 Management Profiles

Management profiles define client visible behavior for the management of a particular set of components in the real world. These components are typically organized in management domains. This is illustrated in Fig. 1. below.

Client visible behavior is defined by adapting a number of classes from the CIM schema for a particular purpose. For each class adaptation the profile establishes a real world context. The class adaptation describes the use of the class in that particular context. For example, the System Virtualization Profile [16] defines a class adaptation of the CIM_System class for the representation of system virtualization platforms.

Requirement Level. Definitions in management profiles may be qualified with a requirement level that is one of "required", "conditional" or "optional". Conformant implementations must implement all mandatory elements, must implement conditional elements if a specified condition is true, and may implement optional elements. If implemented, optional elements must implemented as specified. Requirement levels enable the definition of optional and conditional features within profiles. Profiles describe discovery mechanisms enabling clients to detect the presence of such features within implementations.

Profile definition. DMTF management profile specifications are created by DMTF working groups that consist of interested participants from various DMTF member companies. Form and structure of profiles is formally described by a meta document named the Profile Usage Guide (PUG) [13]. Profile development follows a rigid process that involves frequent reviews and ballots on the profile content [14]. Once a stable stage is reached during profile development, the profile is reviewed by the appropriate platform subcommittee and finally by the DMTF technical committee. After approval from the DMTF technical committee profiles are published as preliminary standards and available for download from the DMTF website; for example see the System Virtualization profile [16].

Profile implementation. Management profiles are implemented for one or more instances of the management domain addressed by the profile. For example, the System Virtualization profile might be separately implemented for the Microsoft Windows Server® 2008 product exploiting the Hyper-V™ technology [4], and members of the VMware® Server product suite [6]. Frequently such implementations are provided by the vendor of the managed product itself, but product independent implementations are possible as well. For example, an open source implementation using functions of the libvirt systems management library [9] might be provided for the management of installations of the Xen™ hypervisor [5] and other Linux® based virtualizers like the Kernel Virtual Machine (KVM) [3].

Profile discovery. Profile implementations may be discovered using generalized discovery mechanisms like the service location protocol (SLP) [15]. For that reason every profile is required to define a central class. Profile implementations are required to expose an instance of the CIM_RegisteredProfile class in the CIM interop namespace, and provide an association path from that instance to instances of the central class. These requirement ensures that profile conformant central class instances can be discovered.

For example, a management client exploiting the System Virtualization profile [16] may discover implementations of that profile using SLP. As a result the client receives a set of HTTP addresses where each element identifies one implementation of the profile. The client may then continue using normal CIM browsing techniques like traversing associations to inspect the hierarchically organized set of CIM instances exposed by each of these implementations.

3 System Virtualization Management

The management interface defined by the SVPC working group for the management of system virtualization is expressed in form of a comprehensive set of DMTF management profiles.

3.1 Virtual System Profile

The Virtual System profile [17] defines a management interface for the management of single virtual systems. A central paradigm applied when modeling the interface was that general aspects of virtual systems should be manageable by management applications that are not virtualization aware. For that reason the Virtual System profile is derived from the SMASH Computer System profile [18], inheriting the set of management functions defined by the Computer System profile for real systems.

The central concepts introduced by the Virtual System profile are virtual system, virtual resource, virtual system representation and virtual system configuration. As the Virtual System profile is based on the Computer System profile, the set of component profiles defined in context of the Computer System profile for the representation of resources apply to the Virtual System profile as well. Prominent examples are the CPU profile [19] or the System Memory profile [20].

Virtual systems are thought as logically equivalent virtual counterparts of real systems. Each virtual computer system is represented by an instance of the CIM_ComputerSystem class.

Virtual resources. As with real systems, virtual systems are thought to be comprised of a number of (virtual) resources, like for example virtual memory or virtual processors. Virtual resources are represented by instances of the CIM_LogicalDevice class or a class derived from CIM_LogicalDevice.

Virtual system representations are the composition of the instance of the CIM_ComputerSystem class representing a virtual system and the set of instances of the CIM_LogicalDevice class representing its virtual devices, by means of the CIM_SystemDevice association.

Virtual system representations do not generally convey virtualization specific information; instead, the aspect of virtualization is added to a virtual system representation in form of an associated *State* virtual system configuration.

Virtual system configurations. A virtual system configuration is composed of one instance of the CIM_VirtualSystemSettingData class and a set of associated instances of the CIM_ResourceAllocationSettingData class.

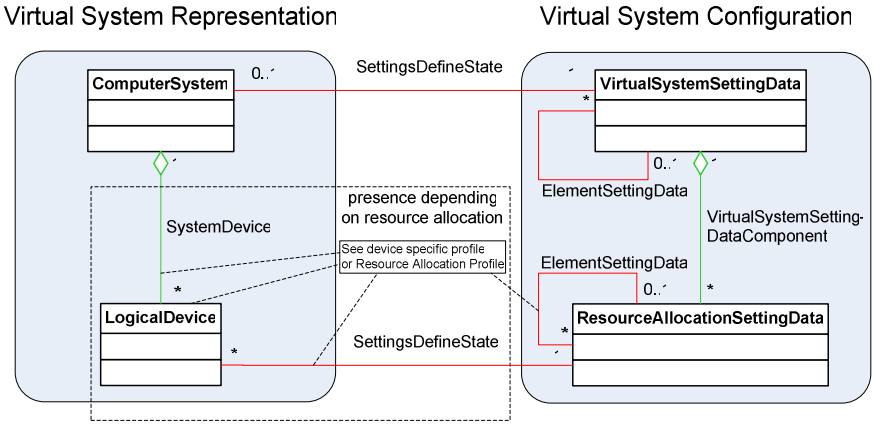


Fig. 2. Virtual System Representation and Configuration

The left side of Fig. 2. shows the class model for a virtual system representation. Note that the format of the virtual system representation is the same as that used for the representation of real systems, effectively making virtual systems inspectable and manageable through interfaces originally defined for real systems.

Virtualization specific aspects are kept in virtual system configurations as shown on the right side of Fig. 2. Two kinds of virtual system configurations are modeled: *Defined* virtual system configurations and *State* virtual system configurations. Elements in these two configurations are connected by the CIM_ElementSettingData association.

Defined virtual system configurations are CIM representations of virtual system definitions. A virtual system definition is a kind of configuration recipe for a virtual system and its resources. Most virtualization platforms use virtual system definitions as the basis for instantiating virtual systems, that is creating the virtual equivalent of a real system. Instantiation typically occurs when a virtual system is activated, and virtual system instances are destroyed when a virtual system is deactivated. It is important to realize that most virtual resources are not allocated while a virtual system is not instantiated. For example this is typically the case for virtual system memory or for processor shares. Furthermore a virtual system instance may deviate from the virtual system definition for various reasons. For example some resources might have been unavailable at virtual system instantiation time, or dynamic changes were applied after the instantiation. On the other hand there are types of virtual resources that are required to be persistently allocated like for example virtual storage extents that contain persistent data.

In many implementations a virtual system definition is an implementation specific formalized description. For example, some implementations define XML schemes and APIs that provide access to respective XML formatted virtual system configuration files, enabling clients to inspect and modify virtual system definitions in an implementation specific way.

Defined virtual system configurations provide CIM based and implementation independent access to virtual system definitions. In a *Defined* virtual system configuration an instance of the CIM_VirtualSystemSettingData class represents general aspects of the virtual system definition, like for example the virtual system type. Instances of the CIM_ResourceAllocationSettingData class represent resource allocation requests for the virtual resources of the virtual system. The resource allocation requests drive the provisioning of virtual resources when the virtual system is activated.

State virtual system configurations are formal representations of the virtual aspects of a virtual system instance. A virtual system instance is the virtual equivalent of a real system. It is composed of virtual resources like for example virtual processors, virtual memory, virtual network ports or virtual disk. Virtual aspects describe how these resources are allocated by the virtualization platform.

State virtual system configurations provide CIM based and implementation independent access to the virtual aspects of virtual system instances. In a *State* virtual system configuration again an instance of the CIM_VirtualSystemSettingData instance represents general aspects of the virtual system, like for example the virtual system type. Instances of the CIM_ResourceAllocationSettingData class represent existing resource allocations that model virtual aspects of allocated virtual resources. Fig. 2 further depicts how elements in the *State* virtual system configuration are associated to their counterparts in the virtual system representation through instances of the CIM_SettingsDefineState association.

State inspection and state management. Like real systems, virtual systems can be arbitrarily activated and deactivated. The activation of a virtual system implies its instantiation, that is, the allocation of virtual resources as defined in the *Defined* virtual system configuration; typically, virtual resources are allocated during virtual system activation. The instantiated virtual system is represented by a virtual system representation representing all allocated resources, with an associated *State* virtual system configuration representing the virtual and allocation aspects.

Virtual system state changes can be inflicted by activities that are not visible through a CIM interface, or may be requested through explicit CIM state management. CIM state management is an optional feature modeled by the RequestStateChange( ) method of the CIM_ComputerSystem class. When invoked on an instance of the CIM_ComputerSystem class representing a virtual system, a state change for that virtual system is requested.

Regardless of the mechanism used for its initiation, a state change becomes visible by monitoring the value of the EnabledState property in that instance.

3.2 System Virtualization Profile

The System Virtualization profile [16] models a management interface for the management of virtualization platforms, addressing the following functional areas:

- The representation of a virtualization platform, the representation of its capabilities and the representation of the relationship between a virtualization platform and its hosted virtual systems.

- The definition of functions for the creation, destruction and modification of virtual systems that are hosted by the virtualization platform. These functions are parameterized such that instances of the CIM_ResourceAllocationSettingData class may be specified as input, effectively enabling a client to provide virtualization specific requirements for the creation or modification of virtual resources.
- The representation and management of virtual system snapshots. Snapshots preserve the state of a virtual system and its resources at the time the snapshot is taken; the virtual system can be restored to the state captured in a snapshot at a later point in time.

Resource type specific details are not specified by the System Virtualization profile itself. Instead the System Virtualization profile provides the scope for a set of resource virtualization profiles. Each resource virtualization profile separately defines virtualization aspects for a particular virtual resource type. Uniformity of resource virtualization profiles is achieved by requiring that each resource virtualization profile is based on the Resource Allocation profile and the Allocation Capabilities profile.

3.3 Resource Allocation Profile

In order to ensure consistent modeling of management interfaces for the management of virtual resource allocation, the SVPC working group initially defined a generic abstract DMTF management profile that serves as the basis for resource type specific resource allocation profiles.

The Resource Allocation Profile [21] defines the resource pool as the conceptual focal point for the allocation of resources, that is, the link between host resources and virtual resources. A resource pool aggregates host resources that support virtual resources through resource allocation.

The aggregated host resources are not required to be of the same resource type as the virtual resource resulting from allocations out of the pool. For example, a resource pool may aggregate host files, but serve for the allocation of virtual disks. Note that the host resources are not required to be shown as elements of the resource pool; instead, the resource pool acts as the conceptual integration point that represents host resources available for or in use by allocations.

The Resource Allocation profile defines a class adaptation of the CIM_ResourceAllocationSettingData class for the representation of resource allocation requests and resource allocations. The CIM_ResourceAllocationSettingData class generically defines various properties that describe allocations, like for example the resource type, the virtual quantity, the amount of reserved host resource and its upper limit or the relative weight of one allocation in relations to others.

3.4 Allocation Capabilities Profile

The Allocation Capabilities profile [22] specifies the foundation for the representation of allocation capabilities of host systems and host resource pools, and for the mutability of virtual resource allocations.

The Allocation Capabilities profile defines a class adaptation of the CIM_ResourceAllocationSettingData class for the representation of settings that define capabilities. The CIM_SettingsDefineCapabilities association qualifies these settings as

minimum, maximum, default or incremental settings. This enables the modeling of ranges and increments of admissible values of properties of the CIM_ResourceAllocationSettingData class. For example, a platforms capability to support the allocation of memory may be specified as ranging from a minimum of 256 MB to a maximum of 4 GB in increments of 64 MB, with a default of 1 GB.

4 Conclusions

The set of DMTF management profiles already released by the SVPC working group provides the foundation for the management of system virtualization platforms. The group continues to extend that set, focusing on still missing resource types like storage and network components. In addition, new functional areas will be addressed in the near future, like for example the definition of management interfaces for the life migration of virtual systems between system virtualization platforms.

Bibliography

[1] AMD® Virtualization Technology, http://www.amd.com/virtualization
[2] Intel® Virtualization Technology, http://www.intel.com/technology/virtualization/index.htm
[3] Kernel Based Virtual Machine, http://kvm.qumranet.com/kvmwiki
[4] Microsoft® Windows Server™ 2008 HyperV™ FAQ, http://www.microsoft.com/windowsserver2008/en/us/hypervfaq.aspx#HyperVGeneral
[5] Xen.org (Description of the Xen® hypervisor), http://www.xen.org/xen/
[6] VMware® (Description of VMware system virtualization solutions), http://www.vmware.com/
[7] IBM® z/VM® (Description of the z/VM hypervisor), http://www.vm.ibm.com/
[8] IBM® PowerVM® (Description of IBM PowerVM based system virtualization), http://www-03.ibm.com/systems/power/software/virtualization/
[9] libvirt - The virtualization API, http://libvirt.org/
[10] DMTF DSP2014 System Virtualization Model White Paper, http://www.dmtf.org/standards/published_documents/DSP2014_1.0.0.pdf
[11] System Management Architecture for Server Hardware Whitepaper, http://www.dmtf.org/standards/published_documents/DSP2001_1.0.1.pdf
[12] DMTF CIM Schema, http://www.dmtf.org/standards/cim/
[13] DMTF DSP1001 Profile Usage Guide, http://www.dmtf.org/standards/published_documents/DSP1001.pdf
[14] DMTF DSP4005 Profile Development Process http://www.dmtf.org/standards/published_documents/DSP4005.pdf
[15] IETF RFC2608 Service Location Protocol Version 2, http://www.ietf.org/rfc/rfc2608.txt?number=2608
[16] DMTF DSP1042 System Virtualization profile, http://www.dmtf.org/standards/published_documents/DSP1042.pdf

[17] DMTF DSP1057 Virtual System profile,
http://www.dmtf.org/standards/published_documents/DSP1057.pdf
[18] DMTF DSP1052 Computer System profile,
http://www.dmtf.org/standards/published_documents/DSP1052.pdf
[19] DMTF DSP1022 CPU profile,
http://www.dmtf.org/standards/published_documents/DSP1022.pdf
[20] DMTF DSP1026 System Memory profile,
http://www.dmtf.org/standards/published_documents/DSP1026_1.0.0.pdf
[21] DMTF DSP1041 Resource Allocation profile,
http://www.dmtf.org/standards/published_documents/DSP1041_1.1.0.pdf
[22] DMTF DSP1043 Allocation Capabilities profile,
http://www.dmtf.org/standards/published_documents/DSP1043.pdf
[23] SNIA Storage Management Initiative Specification (SMI-S) Version 1.3,
http://www.snia.org/tech_activities/standards/curr_standard/smi/SMI-S_Technical_Position_v1.3.0r5.zip

Author Index