

Design of a Domain Authorization-based Hierarchical Distributed Resource Monitoring System in cooperation with Resource Reservation

Atsuko Takefusa Hidemoto Nakada Seiya Yanagita
Fumihiko Okazaki Tomohiro Kudoh Yoshio Tanaka

National Institute of Advanced Industrial Science and Technology (AIST)
{atsuko.takefusa, hide-nakada, seiya.yanagita, f-okazaki, t.kudoh, yoshio.tanaka}@aist.go.jp

Abstract

Grid and Network provisioning technology has enabled the construction of high-quality virtual computing infrastructures spanning several administrative organizations. However, it is still difficult for users to monitor the usage of distributed and various resources managed by multiple domains. We propose an authorization-based hierarchical distributed resource monitoring system called DMS that gathers information based on resource reservation, and filters information with the policies specified by the administrators using XACML, which is a standard authorization model and a policy description language. DMS co-works with the GridARS co-allocation framework to retrieve resource reservation information and adopts web services technologies and an extension of a standard data representation set. To confirm feasibility of the DMS system, we describe monitoring strategies for reserved computing and network resources in Collectors and we have developed a WSRF-based DMS prototype, which enables authorization by XACML. The experiments using the prototype system show: (1) Even when DMS employs a large number of policies, the overhead of the XACML authorization decision process is negligible, since that of WSRF/GSI is more dominant in the total processing time, and (2) the benefits of parallel information aggregation from multiple domains make the retrieval latency acceptable.

1. Introduction

Grid and Network provisioning technology has enabled us to construct high-quality virtual computing infrastructures spanning several administrative domains or organizations. We have developed the GridARS framework[15], which negotiates with multiple resource managers and co-allocates suitable resources, and guarantees requested per-

formance and timeframes with advance reservation. GridARS allows users to allocate various resources distributed over multiple domains via a standard interface based on WSRF (Web Services Resource Framework)[9], easily. In the fall of 2006, Japan's G-lambda research team[18] and the United States' EnLIGHTened Computing research team[17] achieved the world's first inter-domain coordination of resource managers for in-advance reservation of network bandwidth and compute resources between and among both the US and Japan[19]. In our demonstration, users succeeded in performing a parallel MPI application and HD video stream over dynamically composed virtual computing infrastructures.

There are a variety of constituent resources distributed over multi-domain environments, and it is difficult for users to monitor the reserved resource status comprehensively. General monitoring tools, such as Ganglia[2], do not support cooperation with resource reservation. In the G-lambda and EnLIGHTened experiments, we developed a reservation resource monitor (RRM) to gather the status, such as reserved, activated, released, and error, of distributed reserved resources from multiple domains and provide this as their visualized information. However, RRM does not support resource monitoring information, such as CPU utilization, network bandwidth and latency, and the connectivity of the provisioned network, and discloses all of the reserved resource information for demonstration purposes. Moreover, resource management domains are composed hierarchically in actual environments and end users may not be informed of actual resource providers and other resource information, especially in commercial services. Therefore, fine grain authorization at each domain is an important issue for providing monitoring information in multi-domain, multi-resource environments.

We designed a hierarchical distributed resource monitoring system called DMS, which cooperates with resource management systems in multiple domains. DMS consists of

Aggregators and Collectors. An Aggregator gathers monitoring information from related Collectors or Aggregators distributed over multiple domains, and provides the information to the requester. Our Collectors monitor the reserved resources periodically, filter the monitoring information by the domain policy, and provides the requester with the authorized information. Collector adopts XACML (eXtensible Access Control Markup Language)[14], which defines fine grained control of authorized activities and a policy authorization model, and allows domain administrators to define access policies, and then filter and provide the information according to their policies. For cooperation with resource reservation, DMS co-works with resource management systems developed using the GridARS co-allocation framework to retrieve reserved resource information. Also, DMS applies Web services technologies and extensions of standard data representation from GLUE, version 2.0[11], for resource monitoring information in order to ensure interoperability.

To confirm the feasibility of the DMS system, we described monitoring strategies of reserved computing and network resources in Collectors, and we have developed a DMS prototype using Globus Toolkit 4 (GT4), a WSRF reference implementation, and a Java-based XACML reference implementation developed by Sun Microsystems[13]. The experiments using the prototype system show: (1) Even when DMS employs a large number of policies, the overhead of the XACML authorization decision process is negligible, since that of WSRF/GSI is more dominant in the total processing time, and (2) the benefits of parallel information aggregation from multiple domains make the retrieval latency acceptable.

2. Design of a Distributed Monitoring System

We propose a Distributed Monitoring System (DMS), an authorization-based resource monitoring system distributed over multi-domain multi-resource environments. DMS shows the following characteristics: Hierarchical architecture, fine grained authorization by XACML, cooperation with resource management systems developed using GridARS, and interoperability guaranteed by using standard data representation and interfaces.

2.1. System Architecture

Figure 1 shows resource management systems with an advance reservation capability and our proposed resource monitoring system (DMS) over a multi-domain multi-resource environment. In Figure 1, DMS indicates a constituent module of DMS, and RMS indicates a resource management system, such as a Grid resource coordinator or scheduler (GRS), a computing resource manager (CRM), or

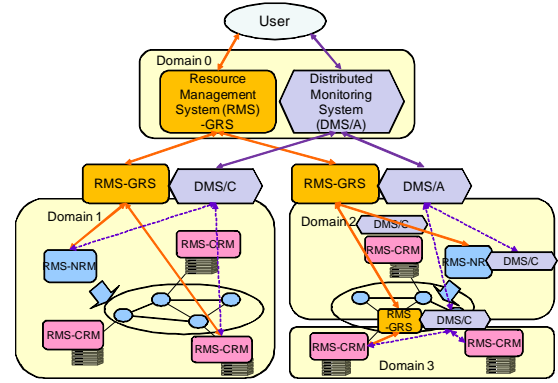


Figure 1. Overview of DMS.

a network resource manager (NRM). DomainX indicates a resource management domain or organization. In Figure 1, a User requests resource co-allocation from RMS and the resource monitoring from DMS in Domain0. However, in an actual resource environment, an RMS, e.g., RMS-GRS in Domain1, provides multiple resources, such as network and computing resources, and another RMS, e.g., RMS-GRS in Domain3, provides resources via another domain's RMS, e.g., RMS-GRS in Domain2.

In order to resolve this situation, we have designed a hierarchical DMS system, which cooperates with RMSs for monitoring of reserved resources and enables authorization of resource monitoring information in each domain. DMS consists of Aggregators, denoted by DMS/A, and Collectors, denoted by DMS/C in Figure 1. An Aggregator gathers requested user's monitoring information collected by related Collectors or Aggregators distributed over multiple domains and provides the user with the information. Collectors located in the lowest layer of a DMS tree monitor the requested user resources periodically, filter the monitoring information based on the domain policy, and provide the requester, the users or a delegated Aggregator, with the authorized information. Both Aggregator and Collector provide a common interface based on WSRF so that domain administrators can select their DMS module function, Aggregator or Collector, and a DMS client, such as a user or Aggregator, can access DMSs in the same manner.

2.2. Cooperation with Resource Reservation and Authentication

In order to monitor reserved resources in multi-domain multi-resource environments, DMS cooperates with the GridARS resource management system. GridARS consists of GridARS-WSRF, which provides a WSRF-based co-allocation framework with advance reservation, and GridARS-Coscheduler, which discovers suitable resources

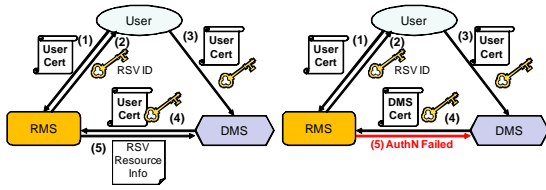


Figure 2. Authentication between RMS and DMS. The delegation version (left), and the non-delegation version (right).

for each user resource request received from GridARS-WSRF, and co-allocates the required resources in coordination with related resource managers such as CRM and NRM. GridARS-WSRF can be used in all RMSs and GridARS-Coscheduler can be used in GRS. For each resource reservation request, the GridARS-based RMS system provides an endpoint reference (EPR) to use to access the reservation service instance for the requested user. The EPR is equivalent to the reservation ID and allows the user to process resource reservation and get the reservation information. Therefore, the user sends DMS a monitoring request with the reservation ID, and DMS gets the reserved resource information using the ID.

GridARS-WSRF has developed using GT4, and supports GSI (Grid Security Infrastructure), which provides an authorization scheme based on a public key infrastructure. Authorization by GSI provides a delegation capability which enables “single-sign-on” to multiple resources. However, RMSs, especially those for commercial services, may not delegate their user to lower layer RMSs because they do not disclose their user information. We have a problem in such non-delegation authentication cases, as shown in Figure 2. Figure 2 shows authentication between an RMS and DMS in delegation (left) and non-delegation (right) situations. In the delegation version, (1) a user sends a reservation request with the user certificate to RMS, (2) the user receives a reservation ID (RID) from the RMS, (3) the user requests monitoring with the user certificate and the RID from DMS, (4) the DMS queries the RMS for the resource information with the user certificate and RID, then (5) the DMS obtains the information from the RMS. On the other hand, in the non-delegation version, (4) the DMS queries the RMS with the DMS certificate, so that authentication fails in the step (5). In order to resolve this problem, the RMS and DMS must be served on the same service container or the RMS and DMS must make some sort of agreement about authentication in advance. We assume the former in our DMS model because an RMS and DMS in a single domain are expected to be operated by the same administrator and the former is thus feasible for the administrator.

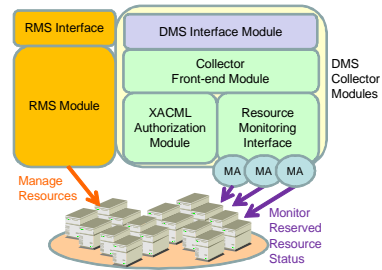


Figure 3. An Overview of a Collector.

2.3. Fine grained Authorization using XACML

In the DMS architecture, Collectors authorize each user access to resource monitoring information by using XACML, so that unauthorized monitoring information is not disclosed to other domains. XACML, standardized by OASIS, defines fine grained control of authorized activities and a policy authorization model. XACML enables general and flexible policy description, even though the description is complicated. This will be resolved by the diffusion of policy description tools, templates, and libraries. Therefore, we adopt XACML as our Collectors authorization model and as a policy description language.

An overview of the Collector architecture and RMS is shown in Figure 3. A Collector consists of the DMS interface module, the Collector front-end module, the XACML authorization module, the resource monitoring interface, and the monitoring agents denoted by MA. The DMS interface module, which is a receiver of the monitoring service, receives user requests and provides resource information to users. The front-end module handles user requests. The authorization module manages service administration policies and authorizes access to the resource monitoring information. MA, which is an actual monitoring entity for each data item, such as CPU load average or network bandwidth, collects dynamic or static resource information in accordance with each user monitoring request. Domain administrators select available MAs in their domains and register access control policies for each piece of monitoring information collected by an MA. The user requests monitoring information from the available information list and can get only the authorized information from among the requested information.

2.4. DMS Interface and the Protocol Sequence

Table 1 shows the DMS service operations provided by an Aggregator and Collector. In Table 1, a monitoring ID,

Table 1. DMS Service operations.

Operation	function	Input / Output
create	Start a monitoring process	RID / MID
configure	Specify collecting information	MID, information list / -
start	Start monitoring at MA	MID (, time) / -
stop	Stop monitoring at MA	MID (, time) / -
getInformation	Get monitoring information	MID (, timeframe) / monitoring information

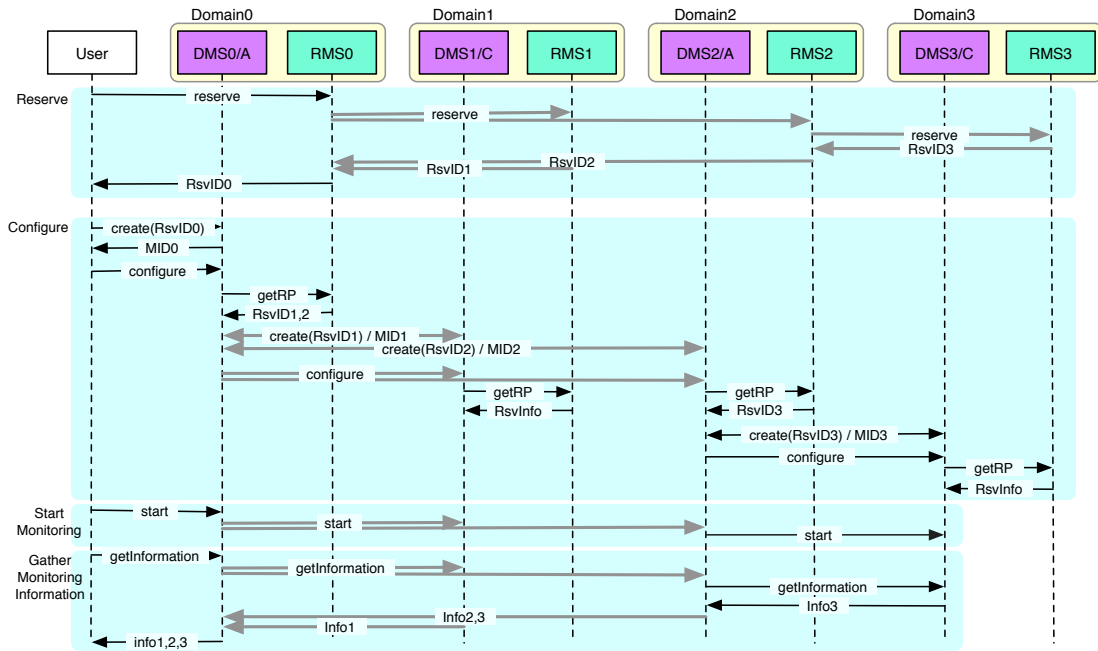


Figure 4. Protocol sequence of monitoring information acquisition

denoted by MID, is issued by DMS for each monitoring request, and a reservation ID, denoted by RID, is issued by RMS for each resource reservation request. Details of each operation are described in Figure 4.

Figure 4 shows the protocol sequence from resource reservation to monitoring information acquisition. We assume four different hierarchical domain architectures in Figure 4. The gray arrows indicate simultaneous processes.

First, a User reserves resources in Domain1 and Domain2 via RMS0 in Domain0 and receives reservation ID RsvID0 from RMS0. At that time, RMS0 negotiates with RMS1 and RMS2 and receives and holds reservation IDs, RsvID1 and RsvID2. RMS2 also negotiates with RMS3 in Domain3 and acquires RsvID3. Details of the reservation protocol sequence are described in [15]. After the reservation process, the User sends DMS0, a `create` request, with RsvID0 to start the monitoring process, and the DMS0 returns monitoring ID MID0 to the User. The User uses MID0

for each monitoring related request in the following process. Next, the User sends DMS0 a `configure` request to specify a list of required monitoring information. Examples of monitoring information are described in Section 3. After DMS0 receives `configure`, DMS0 queries RMS0 with resource reservation information and obtains related reservations in Domain1 and Domain2. DMS0 sends `create` and `configure` to DMS1 and DMS2, and DMS2 sends the requests to DMS3 in the same manner. Then, the User sends DMS0 a `start` request to start resource monitoring at a specified monitoring agent (MA). DMS0 sends `start` to DMS1 and DMS2, and DMS2 sends `start` to DMS3. The MAs start monitoring at the time specified by User in the `start` request or the reservation start time. The `stop` request lets the MA stop monitoring. If the User specifies a time, the MAs stop monitoring at that time. Finally, the User sends DMS0 the `getInformation` request, DMS0 sends `getInformation` to related DMSs, and then the User receives all of the re-

lated resource information, Info1, 2, and 3, as authorized by the proper domains. The User can specify a timeframe for monitoring information in the `getInformation` request. DMS supports the optional operations, `subscribe` and `unsubscribe`, based on WS-Notification[10]. Authorization at each Collector occurs at each notification.

2.5. Data Representation

In order for interoperability with other monitoring tools or user interfaces, DMS applies Web services technologies and extensions of the standard data representation, GLUE version 2.0[11], for resource monitoring information. GLUE defines XML-based data representation for computing resources, but the representation for network resources and time series required for our monitoring is inadequate, so that we extend GLUE for DMS.

3. Monitoring of Reserved Resources

In the following sections, we describe strategies of computing and network resource monitoring by monitoring agents (MA), as shown in Figure 3.

3.1. Monitoring of Computing Resources

Computing resource monitoring information consists of (a) reservation information, (b) reserved resource status, and (c) information related to running applications. (a) includes the reservation ID, the number of reserved CPUs and memory size, start and end time of the reservation, and reservation status. (b) includes statistical data on reserved resource usage, such as amounts of CPU utilized time and memory and disk usage. (b) is collected for each private reservation, and is different from the monitoring information collected by Ganglia or other general cluster monitoring tools. Our monitoring system does not disclose other user usages, even if the resources are shared by multiple users. (c) indicates execution logs of running applications. (c) allows users to acquire distributed application status easily from the DMS framework.

MAs collect (a), (b), and (c) as follows: (a) is provided by CRM via the CRM Web services interface using the reservation ID. (b) is collected via the process information interface provided by each operation system for active processes, and via log files produced by job management middleware or the process accounting package provided by each operating system for terminated processes. In the Linux system, the former is the `proc` file system and the latter is the `psacct` package, respectively. (c) is provided by the specified files for each application.

3.2. Monitoring of Network Resources

In NRM, a network management system (NMS) is generally working for monitoring of all of the network status information in the management domain, and for periodically confirming the integrity of the network. NMS also monitors provisioning of requested paths and confirms that requested bandwidths of the paths are guaranteed. Here, a “path” between two termination points is a reservation unit of network resources in our co-allocation system. NMS monitors both network hardware interfaces of path terminal points and stores the monitoring information in an internal database. Therefore, the MA obtains network monitoring information from the database.

Network resource monitoring information consists of (a) reservation information and (b) reserved path status. (a) includes authenticated user information and reserved peak bandwidth, and other reservation information as described in Section 3.1. (b) includes link status (UP or Down) and packet statistical information, such as the number and the amount of input and output packets. The number of packets includes transmission packets, destroy packets, and error packets.

NMS monitors path status by using SNMP (Simple Network Management Protocol)[3], which is a general monitoring method used to access the network interfaces of path terminal points. Most of the network hardware physical interfaces support SNMP because SNMP provides standard monitoring information, defined by MIB (Management Information Base)[4]. However, virtual interfaces, such as VLAN, might not support SNMP. In this case, NMS uses the CLI (Command Line Interface) provided by the network hardware for monitoring.

3.3. Monitoring of End-to-end Information

Measurements of end-to-end throughput and delay belong to network information, however, NRMs may not be able to collect such information. For example, if a reserved path A to C consists of two sub paths, A to B and B to C, managed by NRM0 and NRM1, NRM0 and NRM1 cannot measure throughput and latency between A to C.

Therefore, end-to-end information is collected by CRMs. We extend GridARS CRM to have the following capability: for each user reservation request the related CRMs exchange allocated computer information at the reservation start time, and each CRM collects end-to-end information between its node and a node managed by the other CRM. An MA in CRM provides this end-to-end information to the requested users. End-to-end information is measured by `ping`.

```

<Policy PolicyId="TimeRangePolicy1"
  RuleCombiningAlgId=
"rule-combining-algorithm:permit-overrides">
  <Description>Allow anyone to access
  between 9am and 11pm. </Description>
  <Target>
    <Subjects><AnySubject /></Subjects>
    <Resources><AnyResource /></Resources>
    <Actions><AnyAction /></Actions>
  </Target>
  <Rule Effect="Permit"
    RuleId="PermitDuringSpecifiedTimeRange">
    <Condition FunctionId="TimeFunction">
      <AttributeValue DataType="xsd:time">
        09:00:00</AttributeValue>
      <AttributeValue DataType="xsd:time">
        23:00:00</AttributeValue>
    </Condition>
  </Rule>
  <Rule Effect="Deny"
    RuleId="DenyAllOthers"/>
</Policy>

```

Figure 5. A policy example: A policy to allow access within a specified time.

```

<PolicySet PolicySetId="PolicySet1"
  PolicyCombiningAlgId=
"policy-combining-algorithm:deny-overrides">
  <Description> Example policy set.
  </Description>
  <Target>
    <Resources><AnyResource /></Resources>
  </Target>
  <PolicyIdReference>TimeRangePolicy1
  </PolicyIdReference>
  <PolicyIdReference>DNMatchPolicy1
  </PolicyIdReference>
</PolicySet>

```

Figure 6. A policy set example.

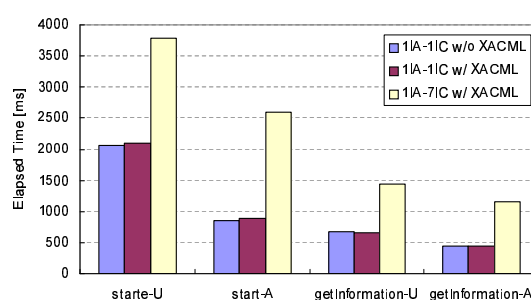


Figure 7. Elapsed time of start and getInformation.

4. DMS Prototype and the Experiments

4.1. WSRF-based DMS Prototype

We have developed a DMS prototype using Globus Toolkit 4 (GT4), a reference implementation of WSRF, and a Java-based XACML reference implementation developed by Sun Microsystems[13]. GT4 supports GSI-based user authentication and delegation, grid-mapfile, that describes mapping information between distinguished name and local user name, and secure transfer via SSL (Secure Socket Layer).

4.2. Examples of XACML-based Policy Description

The XACML reference implementation makes a policy decision from the specified policy files and evaluation functions. Evaluation functions consist of functions defined by the XACML specification and original functions developed by each administrator. Figure 5 is an example of a policy description, which specifies a data access timeframe. In Figure 5, the PolicyID attribute in the Policy tag specifies the ID of this policy, and the RuleCombiningAlgId attribute specifies the decision policy among multiple rules. Here, permit-overrides indicates that the policy de-

cision becomes “Permit” if any rule in this policy decides on Permit. The <Target> tag specifies the application of this policy to all items specified in Subject, Resource, Action. The <Rule> tag lists rules to specify an available timeframe with an original evaluation function whose ID is TimeFunction.

Figure 6 shows an example of a policy set of two policies. The PolicySetId attribute in <PolicySet> specifies the policy set ID, and the RuleCombiningAlgId attribute specifies the decision policy among multiple policies. deny-override indicates the policy set decision becomes “Deny” if any policy in this policy set decides on Deny.

4.3. Experiment 1: Overhead in Getting Information

Using this prototype, we perform two kinds of experiments. The first experiment shows the overhead involved in getting information from the WSRF-based DMS prototype. We compare the cases in which Aggregator and Collector

Table 2. Elapsed time of `start` and `getInformation` measured by InformationCollectors. 7Collector indicates the average of seven results.

	start	getInformation
1A-1C w/o XACML	2.7 ms	0.8 ms
1A-1C w/ XACML	2.2 ms	1.9 ms
1A-7C w/ XACML	3.0 ms	1.4 ms

are one to one and one to seven. In this experiment, an Aggregator and Collectors are located in the same cluster and latencies between the hosts are about $40 \mu s$. All of the hosts are connected to a 1 Gbps network switch. The Java version is JDK 1.5.0, provided by Sun Microsystems. The User receives the following UNIX command results:

```
$ ps ux --cols 80 -u <user name>
```

In this experiment, Collectors make decisions from a policy set including the following two policies:

- Whether the User request is issued in the specified timeframe or not.
- Whether there is mapping between the distinguished name in the user certificate and the local user name in the `grid-mapfile` or not.

Figure 7 shows the overhead of `start` and `getInformation` in ms. The X axis indicates experimental results measured from a User denoted by -U and an Aggregator denoted by -A. The difference between -U and -A is the WSRF over GSI-based message passing overhead. Three different results, 1A-1C w/o XACML, 1A-1C w/ XACML, and 1A-7C w/XACML, are a combination of one Aggregator and one or seven Collector(s) with or without authorization by XACML. The results show the shortest elapsed times from 10 measurements.

Comparison of 1A-1C w/ and w/o XACML shows that there is less authorization overhead in the experiments. We also measured the authorization overhead in the Collector at that time and they were 0.8 to 3 ms. Therefore, the overhead of WSRF/GSI processes is dominant and authorization process overhead is negligible.

In a comparison of 1A-1C and 1A-7C, response times of 1A-7C become longer due to negotiations with seven Collectors via WSRF/GSI. However, the ratios of 1A-7C and 1A-1C remain 2.9 times in `start-A` and 2.6 times in `getInformation-A` because this negotiation occurred concurrently. So the response times are not proportional to the number of Collectors and are acceptable in the case where the Aggregator collects information from seven different Collectors.

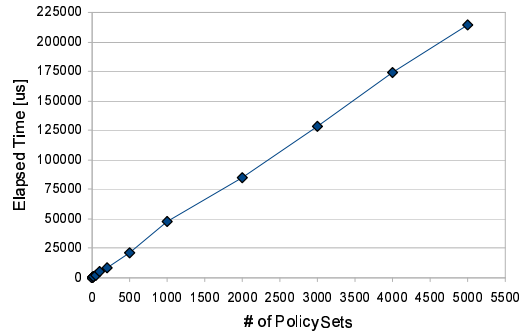


Figure 8. Elapsed time of XACML authorization decisions.

4.4. Experiment 2: XACML Authorization Overhead

Next, we investigate the elapsed time of an XACML authorization decision for the altered number of policies. In our prototype system, policies are read at the start time of the service container. So observed elapsed time does not include policy reading time. The effect of GC (Garbage Collection) is also excluded from the elapsed time.

The experimental results are shown in Figure 8. The vertical axis indicates elapsed time in μs and the horizontal axis indicates the number of policy sets. We employ the policy set, which includes two policies, used in Section 4.3 and the number of policy sets is altered from 1 to 5000. The results show the shortest elapsed times from 10 measurements.

Figure 8 shows that authorization overhead is approximately proportional to the number of policy sets. In addition, the overhead is 50 ms for the 1000 policy set and 230 ms for the 5000 policy set. Therefore, even for a large number of policies, the XACML overhead is still negligible over the total processing time described in Section 4.3.

5. Related Work

Globus Toolkit, version 4.2 (GT4) provides an XACML-based authorization framework which enables authorization to access a WSRF-based service instance[5]. Authorization granularity in GT4 is “service” while the granularity in DMS is each monitoring information item collected by MAs. DMS enables fine grained authorization control.

MonALISA (MONitoring Agents using a Large Integrated Services Architecture)[6] stores a registry with monitoring information collected by general monitoring tools, such as Ganglia and MRTG (Multi Router Traffic Grapher)[7], and provides the information to users via a client interface based on Web services. MonALISA does

authenticate users, but does not authorize access to the monitoring information. Inca[12] is a user-level Grid monitoring system for TeraGrid[16] resource administrators, and collects monitoring information by user authority. MonALISA and Inca gather and manage monitoring information from multiple domains in a central database, thus domain administrators cannot control access policies to the information according to their management policies.

WMSMonitor is a monitoring tool for workload and job lifecycle for EGEE gLite[1]. It supports requirements of various user categories, such as administrators, developers, and VO (virtual organization) users. AMon is a monitoring system used to collect status data, resource usage data, and user output for applications used by the High Energy Physics Community Grid (HEPCG) in the D-Grid initiative[8]. WMSMonitor and AMon do not filter monitoring information in each resource management domain, while DMS enables authorization in each domain. In addition, none of the above monitoring tools deal with virtual computing and network infrastructure provisioned dynamically.

6. Conclusions

We proposed an authorization-based hierarchical distributed resource monitoring system called DMS that gathers information based on resource reservation and filters information with the domain administrators' policies using XACML. DMS consists of Aggregators, which enable data aggregation from distributed DMS modules, and Collectors with monitoring and XACML-based authorization capabilities in each management domain. DMS co-works with resource management systems developed using GridARS to retrieve reserved resource information. Also DMS adopts Web services technologies and the extension of GLUE, version 2.0. To confirm the feasibility of DMS, we described resource monitoring strategies and we have developed a WSRF-based prototype. The experiments using the prototype show: (1) Even when DMS employs a large number of policies, the overhead of the XACML authorization decision process is negligible, since that of WSRF/GSI is more dominant in the total processing time, and (2) the benefits of parallel information aggregation from multiple domains make the retrieval latency acceptable.

For future work, we will continue to develop the DMS system based on the proposed design, and the graphical user interface.

Acknowledgments

This work was partly funded by the National Institute of Information and Communications Technology.

References

- [1] D. Cesini, D. Dongiovanni, E. Fattibene, and T. Ferrari. WMSMonitor: a Monitoring Tool for Workload and Job Lifecycle in Grids. In *"Proc. the 9th IEEE/ACM International Conference on Grid Computing (Grid2008)"*, pages 209–216, Oct. 2008.
- [2] Ganglia Monitoring System. <http://ganglia.info/>.
- [3] J.D. Case and M. Fedor and M.L. Schoffstall and C. Davin. Simple Network Management Protocol (SNMP), 5 1990.
- [4] K. McCloghrie and M.T. Rose. Management Information Base for Network Management TCP/IP-based Internets: MIB-II, 3 1991.
- [5] B. Lang, I. Foster, F. Siebenlist, R. Ananthkrishnan, and T. Freeman. A Multipolicy Authorization Framework for Grid Security. In *Proc. NCA06*, 2006.
- [6] C. Legrand, H. B. Newman, R. Voicu, C. Cirstoiu, C. Grigoras, M. Toarta, and C. Dobre. MonALISA: An Agent based, Dynamic Service System to Monitor, Control and Optimize Grid based Applications. In *Proc. CHEP2004*, September 2004.
- [7] MRTG Monitoring Tool. <http://www.mrtg.org/>.
- [8] R. Mueller-Pfefferkorn, H. Eichenhardt, R. Neumann, and T. William. User- and Job-Centric Monitoring: Analysing and Presenting Large Amounts of Monitoring Data. In *"Proc. the 9th IEEE/ACM International Conference on Grid Computing (Grid2008)"*, pages 225–232, Oct. 2008.
- [9] S. Graham and A. Karmarkar and J. Mischinsky and I. Robinson and I. Sedukhin, editor. *Web Services Resource 1.2 (WS-Resource)*. OASIS, Apr 2006.
- [10] S. Graham and D. Hull and B. Murray, editor. *Web Services Base Notification 1.3 (WS-BaseNotification)*. OASIS, Oct 2006.
- [11] Sergio Andreatto, editor. *GLUE Specification v2.0 (OGF Working Draft)*. OGF, 2008. <http://forge.gridforum.org/sf/go/doc15226?nav=1>.
- [12] S. Smallen, C. Olschanowsky, K. Ericson, P. Beckman, and J. M. Schopf. The Inca Test Harness and Reporting Framework. In *Proc. the IEEE/ACM SC2004 Conference*, November 2004.
- [13] Sun's XACML Implementation. <http://sunxacml.sourceforge.net/>.
- [14] T. Moses, editor. *eXtensible Access Control Markup Language (XACML) Version 2.0*. OASIS, 2 2005.
- [15] A. Takefusa, H. Nakada, T. Kudoh, Y. Tanaka, and S. Sekiguchi. GridARS: An Advance Reservation-based Grid Co-allocation Framework for Distributed Computing and Network Resources. *Job Scheduling Strategies for Parallel Processing*, 4942/2008:152–168, 4 2008.
- [16] TeraGrid. <http://www.teragrid.org/>.
- [17] The EnLIGHTened Computing project. <http://enlightenedcomputing.org/>.
- [18] The G-lambda project. <http://www.g-lambda.net/>.
- [19] S. R. Thorpe, et al. G-lambda and EnLIGHTened: Wrapped In Middleware Co-allocating Compute and Network Resources Across Japan and the US. In *Proc. GridNets2007*, 2007.