Empowered by Innovation NEC

ASET

Automatic Cache Update Control for Scalable Resource Information Service with WS-Management

September 23, 2009

Kumiko Tadano, Fumio Machida, Masahiro Kawato, and Yoshiharu Maeno

Service Platforms Research Laboratories,

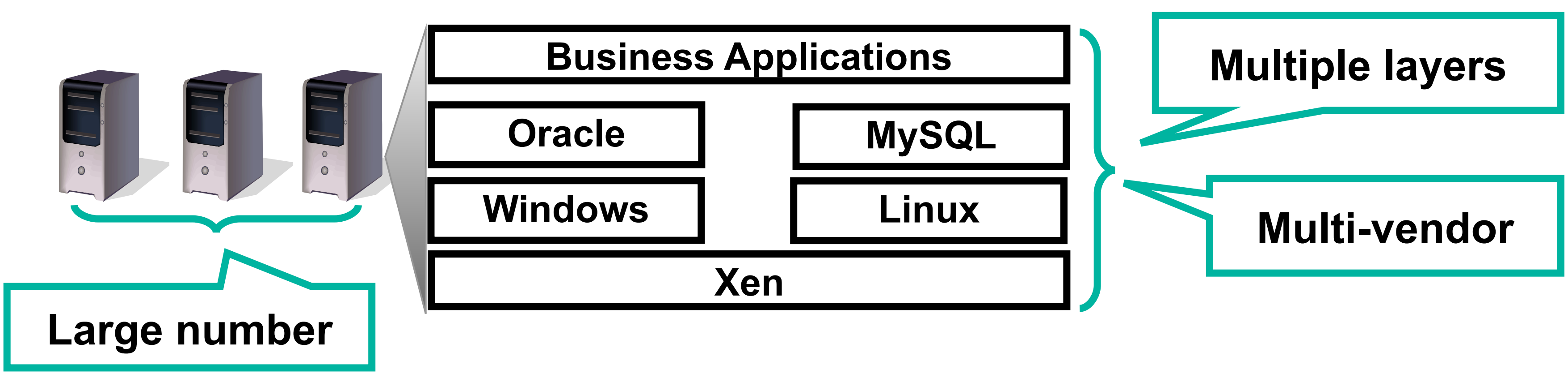
NEC Corporation

Outline

- Introduction
  - Secure Platform Project
- Approach
  - Requirements of Resource Information Service
  - Existing Approaches for Cache Update Control
  - Proposed Solution
- Design
  - Resource Information Manager
  - Resource Information Cache Update Control
- Performance Evaluation
  - Experiment
  - Simulation Studies
- Conclusion

Introduction

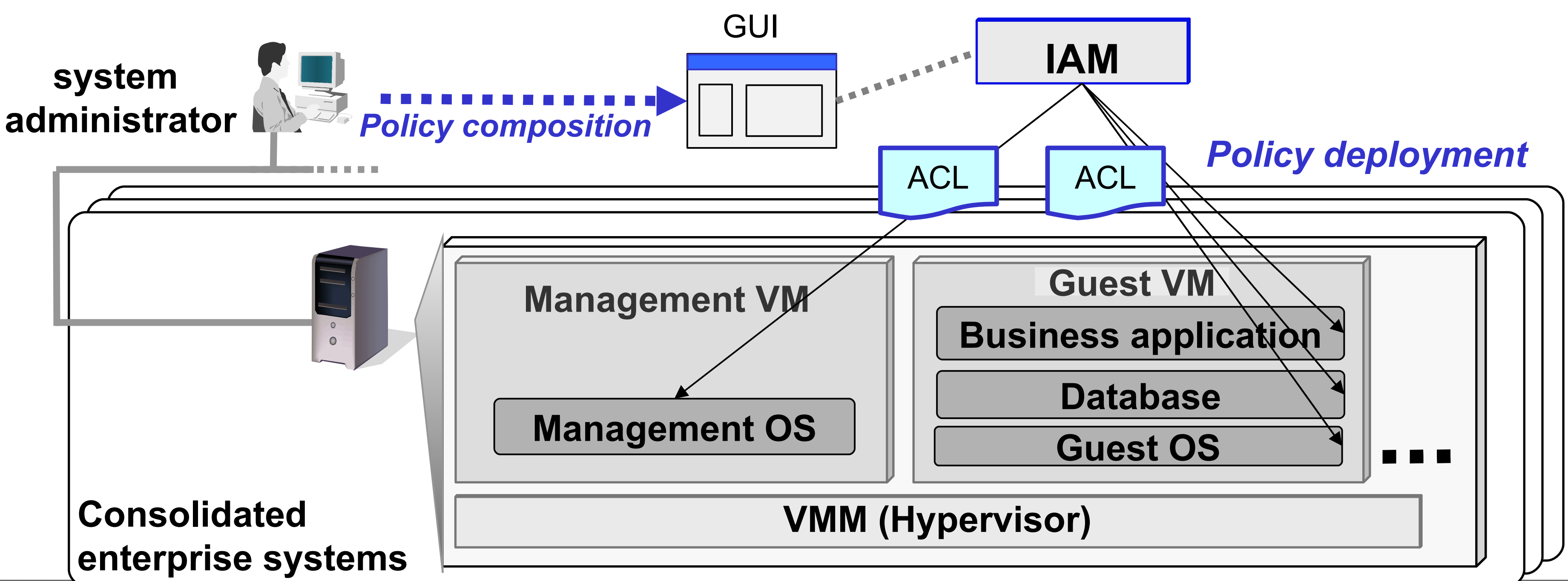
- Advanced datacenters host many enterprise applications using virtualization for server consolidation
- Consolidated security management is a complex and risky task for system administrators
  - There are a large number of managed resources
  - All the security modules should be configured properly and consistently
  - The complexity may cause misconfiguration



Secure Platform Project

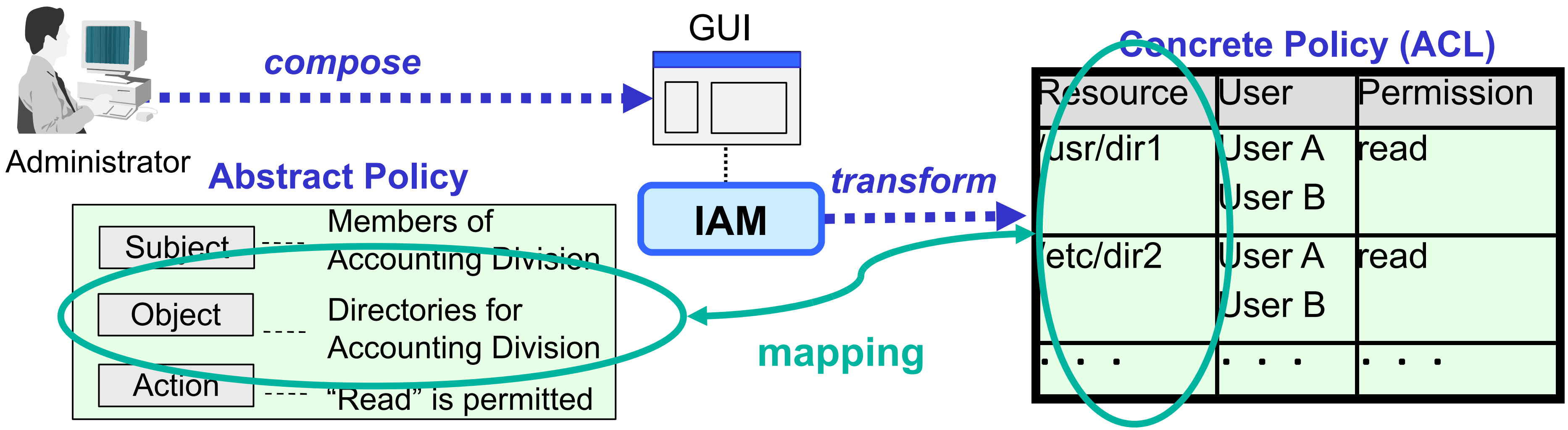
Integrated Access Control Manager (IAM)

- Integrates access control policies for various resources (OS, VM, DB..) based on the standards (CIM, WS-Management)
- Provides a common interface to compose abstract access control policies
- Automatically transforms the abstract policies into concrete policies (ACLs) for individual resources
  - The transformation requires information of various resources



Secure Platform Project -Policy Transformation-

- Administrator composes RBAC policy using GUI



- For policy transformation, mapping between abstract and concrete resources is required
- For the mapping, the administrator requires information of various resources
  - The mapping may vary with time
- Fast retrieval of resource information is the key point of this work

Outline

- Introduction
  - Secure Platform Project
- Approach
  - Requirements of Resource Information Service
  - Existing Approaches for Cache Update Control
  - Proposed Solution
- Design
  - Resource Information Manager
  - Resource Information Cache Update Control
- Performance Evaluation
  - Experiment
  - Simulation Studies
- Conclusion

Requirements of the Resource Information Service

Requirements

- Short response time for information retrieval
- Up-to-date resource information
- Scalability for a large amount of information on various resources (e.g. 1000 managed physical hosts)

A solution

- Caching of resource information for quick response
- Cache update control method to keep resource information up-to-date

What kind of cache update control method meets the requirements?

Existing Approaches for Cache Update Control

Notification-based cache update control

- An agent in a managed host notifies resource state changes to the manager
- The manager may get too many notifications from many agents at a time
- ⇒Overload of management host

Reactive cache update control

- The manager updates cached information reactively when cache miss occurs
- It always takes long time to retrieve resource information for the first time after expiration
- ⇒Long response time

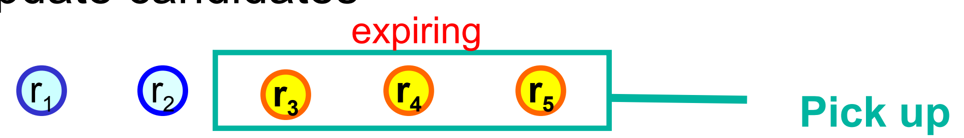
Proactive cache update control: Web Service Polling Engine

- The manager updates cached information proactively and periodically
- This algorithm updates whole resource information and needs to run too many update processes
- ⇒Does not scale

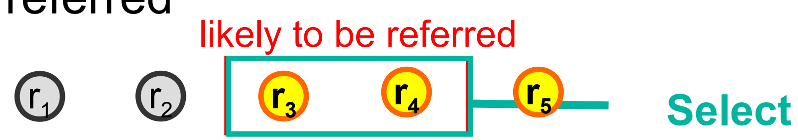
Proposed Solution

A selective cache update control method for resource information cache

- The method periodically invokes the cache update control process:
  1. picks up the cached resource instances which will expire in the near future as update candidates



  2. selects the resource instances from the update candidates which are likely to be referred



  3. updates the selected resource instances

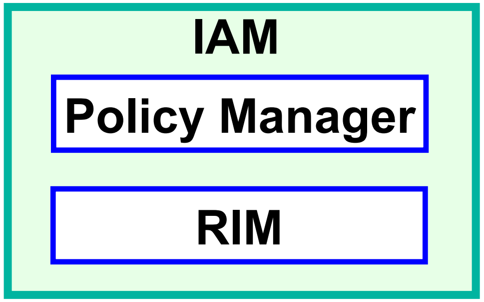
Advantages

- Short query response time by automatically updating cached information likely to be referred
- High scalability by selecting part of cached instances as update targets

Outline

- Introduction
  - Secure Platform Project
- Approach
  - Requirements of Resource Information Service
  - Existing Approaches for Cache Update Control
  - Proposed Solution
- Design
  - Resource Information Manager
  - Resource Information Cache Update Control
- Performance Evaluation
  - Experiment
  - Simulation Studies
- Conclusion

Resource Information Manager (RIM)



Structure

- IAM consists of Policy Manager and RIM

Functionalities

- RIM receives CQL queries from system administrator via Policy Manager and returns the requested resource information
  - CQL: CIM Query Language
- RIM adopts the selective cache update control method for quick response

Compliance with standards

- RIM collects information of various resources with WS-Management
- Resource information is represented with CIM based information schema

Control Sequences of Resource Information Manager

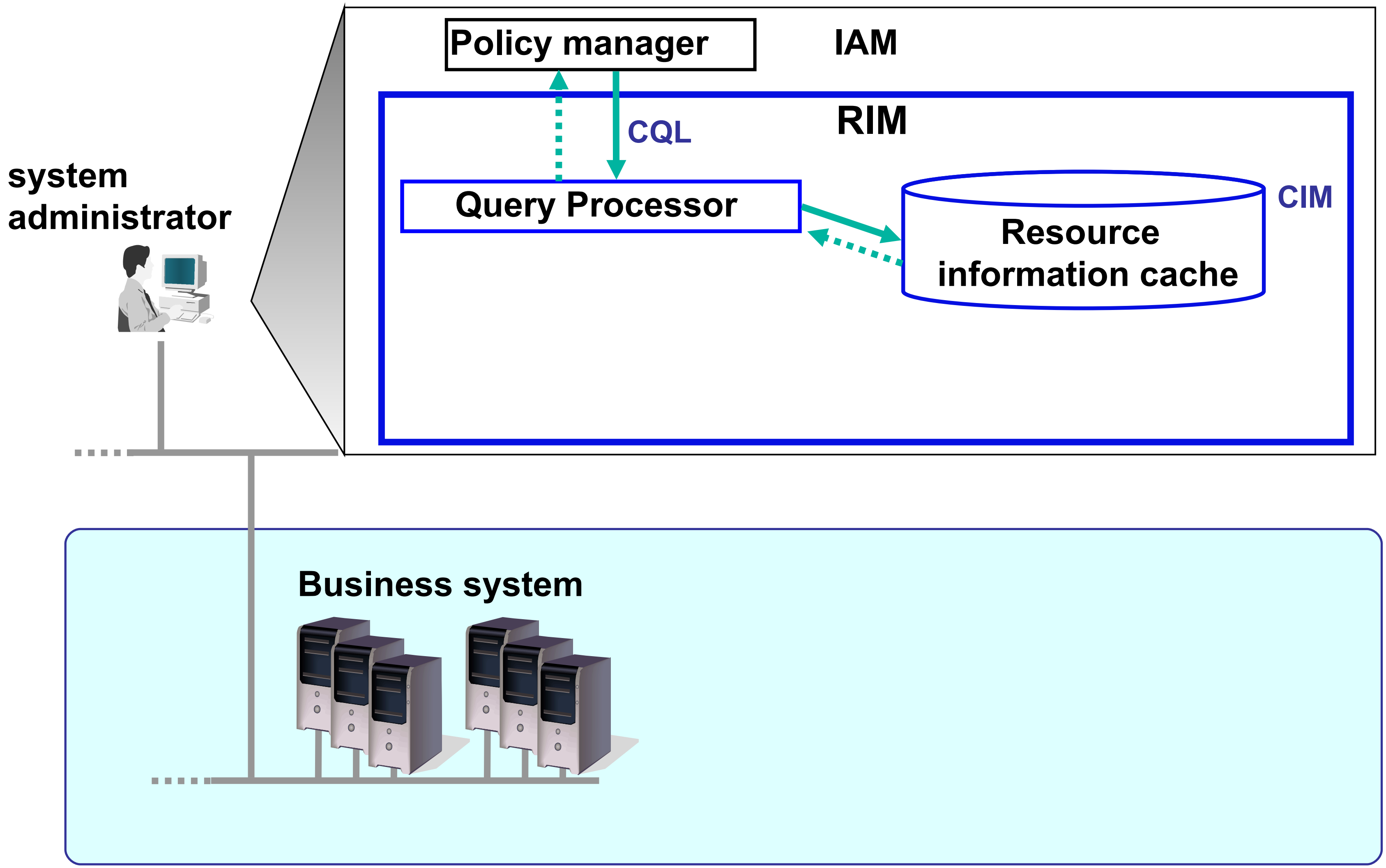
Query Processing

- Cache Hit : When the requested information exists in the cache
- Cache Miss: When the requested information does not exist in the cache or the cached information is expired

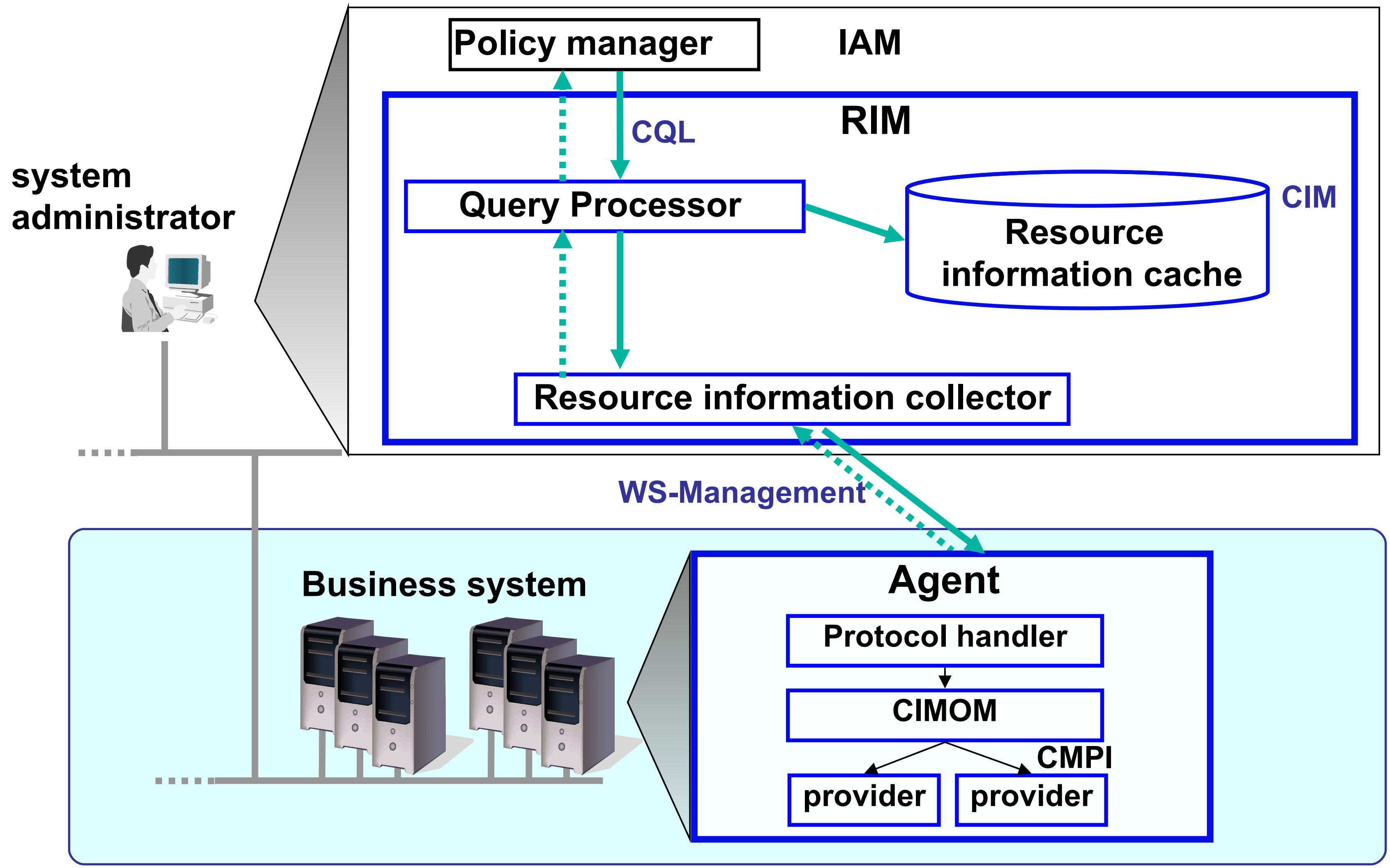
Automatic Cache Update Control

- Cache update controller (CUC) periodically invokes the selective cache update control process
  - CUC is a sub-component of RIM

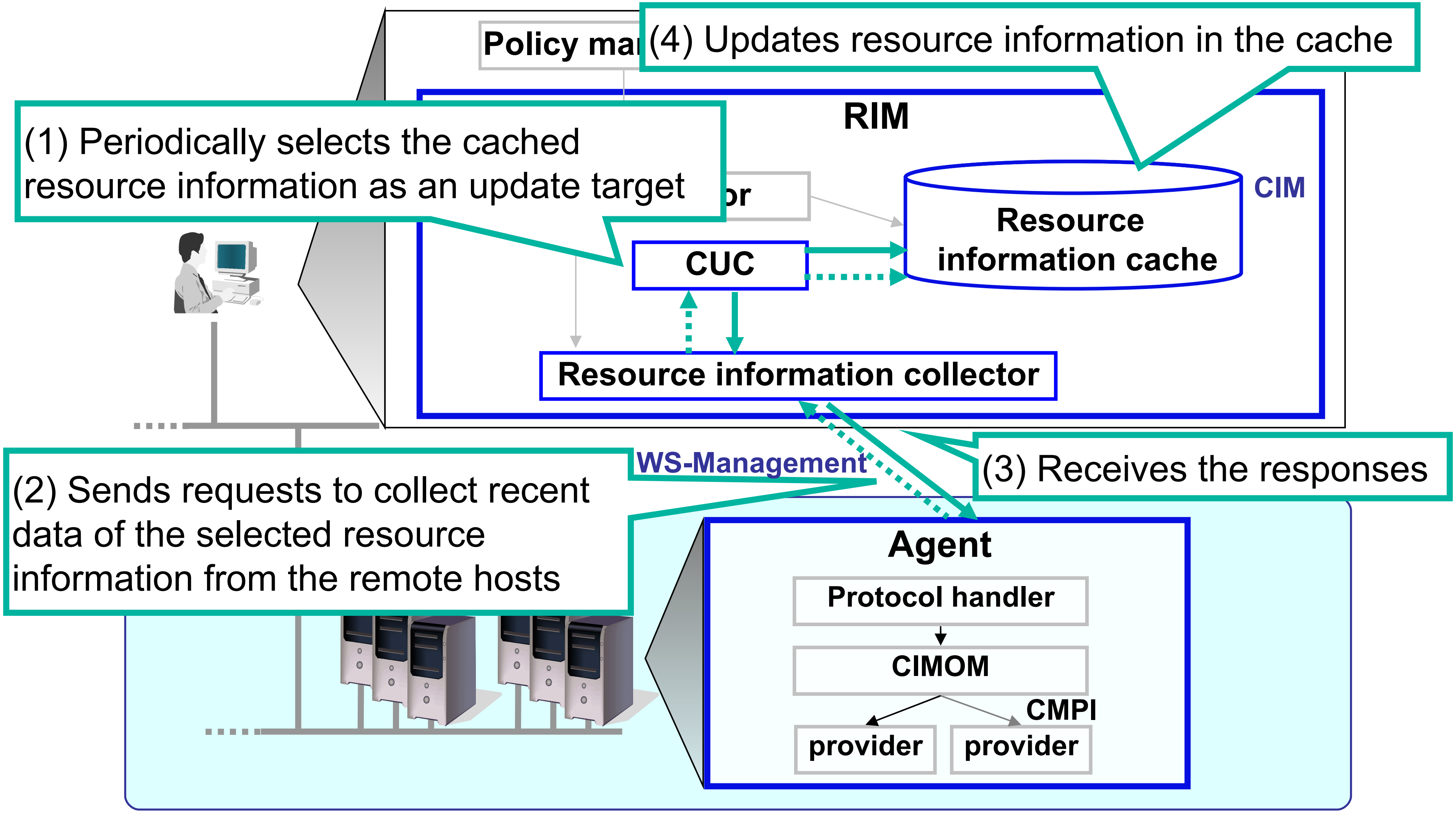
Control Sequence －Cache Hit-



Control Sequence -Cache Miss-



Control Sequence -Automatic Cache Update Control-



  Empowered by Innovation NEC

Update Target Selection Process of the CUC

Conditions

- The number of resources whose information is updated during a certain time interval $t_{interval}$ is limited by m
- Cached Information of each resource has expiration time that is assigned using *time to live* (*TTL*).

CUC works as follows:

Repeat *step 1* through *step 5* once in $t_{interval}$

1. Select resources whose cached information will expire within *threshold of time to expiration* (T_{ex}) as candidates for update
2. For each candidate, predict the query request rate based on the system administrator's access patterns
3. Select m candidates with highest query request rates as update targets
4. Collect resource information of the update targets through the resource information collector
5. Update information in the cache with collected resource information

Outline

- Introduction
  - Secure Platform Project
- Approach
  - Requirements of Resource Information Service
  - Existing Approaches for Cache Update Control
  - Proposed Solution
- Design
  - Resource Information Manager
  - Resource Information Cache Update Control
- Performance Evaluation
  - Experiment
  - Simulation Studies
- Conclusion

Performance Evaluation

Purpose

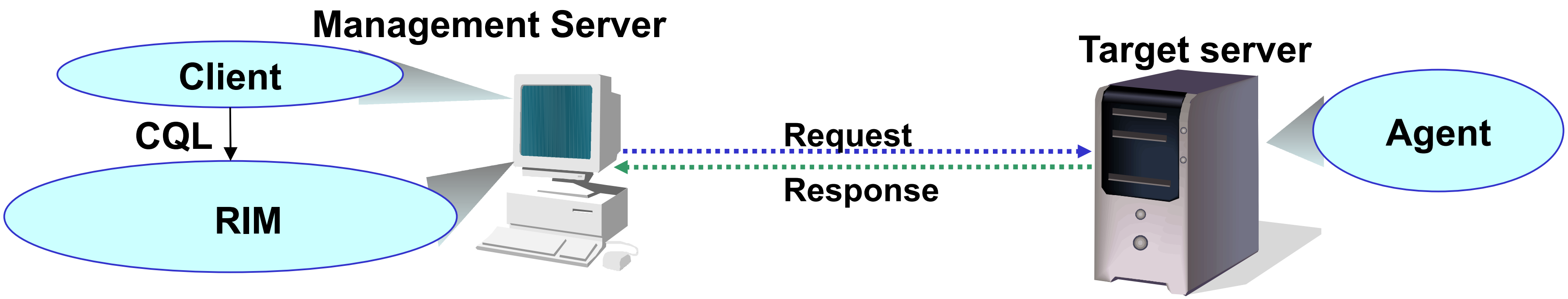
- Query performance measurement for cache hit and miss cases
- Simulation of the average query response time with the proposed method in maximally and minimally effective cases
  - The simulation uses the results obtained from the measurements

Experimental Setup for Query Performance Measurement

We measured the response time of the CQL

- Each CQL requests to retrieve an instance of a CIM class

Case 1 : Cache Hit
Case 2 : Cache Miss



Management server	
CPU	Intel Xeon E5205 1.86GHz
Memory	10GB
OS	CentOS 5.1

Target server	
CPU	Intel Xeon 5160 3.00GHz
Memory	10GB
OS	CentOS 5.1

Experiment -Result-

Average response time of each query (measured 10 times)

Query target	CQL	Response time [sec]		Size of response message [bytes]
		Cache miss	Cache hit	
CIM_ComputerSystem	SELECT * FROM CIM_ComputerSystem WHERE Name='hostA'	3.0691	0.0043	1734
CIM_FileSystem	SELECT * FROM CIM_FileSystem WHERE Name='/'	1.0686	0.0042	1886
SPF_Directory	SELECT * FROM SPF_Directory WHERE Name='/etc/'	0.0705	0.0046	8979
CIM_LogicalFile	SELECT * FROM CIM_LogicalFile WHERE Name='/etc/yum.conf'	0.0411	0.0053	1496
CIM_EnabledLogicalElementCapablities	SELECT * FROM CIM_EnabledLogicalElementCapabilities	1.0939	0.0044	1556
Average		1.0686	0.0046	3130

Simulation Studies

To evaluate the CUC in a consistent and systematic manner, we simulate the CUC in the maximally and minimally effective cases and the reactive cache update control:

- Maximally effective case :optimal proactive update
  - The query request rate of each resource is completely estimated
- Minimally effective case :random proactive update
  - Resource information is updated just randomly
- Reactive update
  - Automatic cache update is not performed

Simulation of the CUC

Simulated value

- We simulated average query response time T_{res}

$$T_{res} = p \cdot T_{hit} + (1 - p) \cdot T_{miss}$$

T_{hit} :Query response time (cache hit)
T_{miss} :Query response time (cache miss)
p :Cache hit rate

- T_{hit} and T_{miss} are obtained from measurements

Conditions

- Constant query request rate s_i for each target resource r_i is given

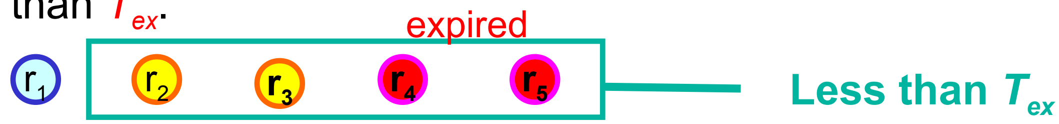
Update Control Methods to be Compared

- *Op-proactive*: Optimal proactive update
  - Selects resources whose cached information will expire within T_{ex} in descending order of query request rate, and updates them proactively
- *Rd-proactive*: Random proactive update
  - Selects resources whose cached information will expire within T_{ex} randomly and updates them proactively
- Reactive: Reactive update
  - Updates resource information only when cache miss occurs.

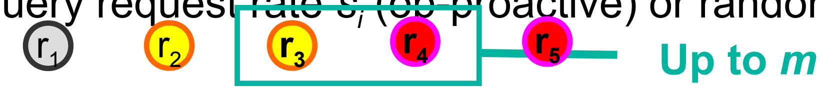
Simulation Procedure

The simulation is performed according to the following steps:

- *Step 1.* Extract cached resource information whose remaining time before expiration is less than T_{ex}.

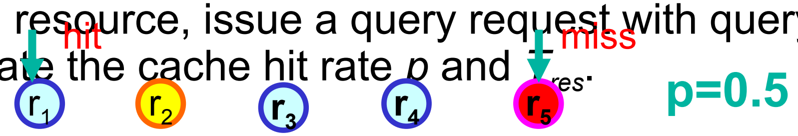


- *Step 2.* Select update targets from the extracted resource information in step 1 up to m based on query request rate s_i (op-proactive) or randomly (rd-proactive).



- *Step 3.* Update expiration time of the update targets with the summation of the current time and TTL_i.

- *Step 4.* For each resource, issue a query request with query request rate s_i. From the result, calculate the cache hit rate p and T_{res}.



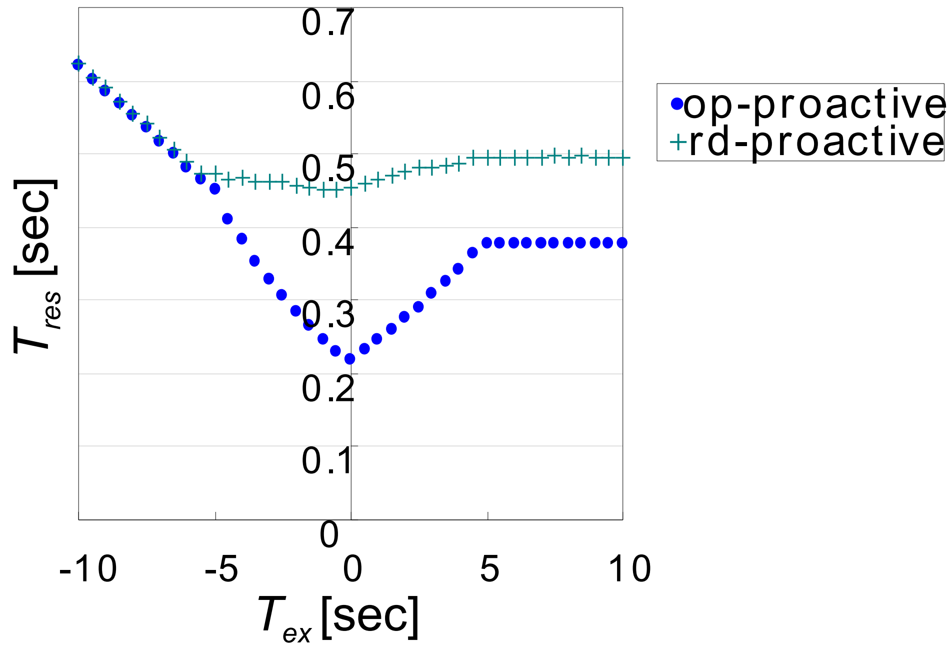
- *Step 5.* Repeat step 1 to step 4 from t_{start} to t_{end} at a certain update interval $t_{interval}$.

Reactive method skips steps 1-3

In the simulation, we evaluate T by varying the T and m

Simulation Results –Effect of varying T_{ex} on T_{res}-

- We evaluated the average query response time T_{res} by varying the threshold of time to expiration T_{ex} in the range of -10 to 10 [sec]
  - Obtain the optimal value of the T_{ex} to minimize T_{res}



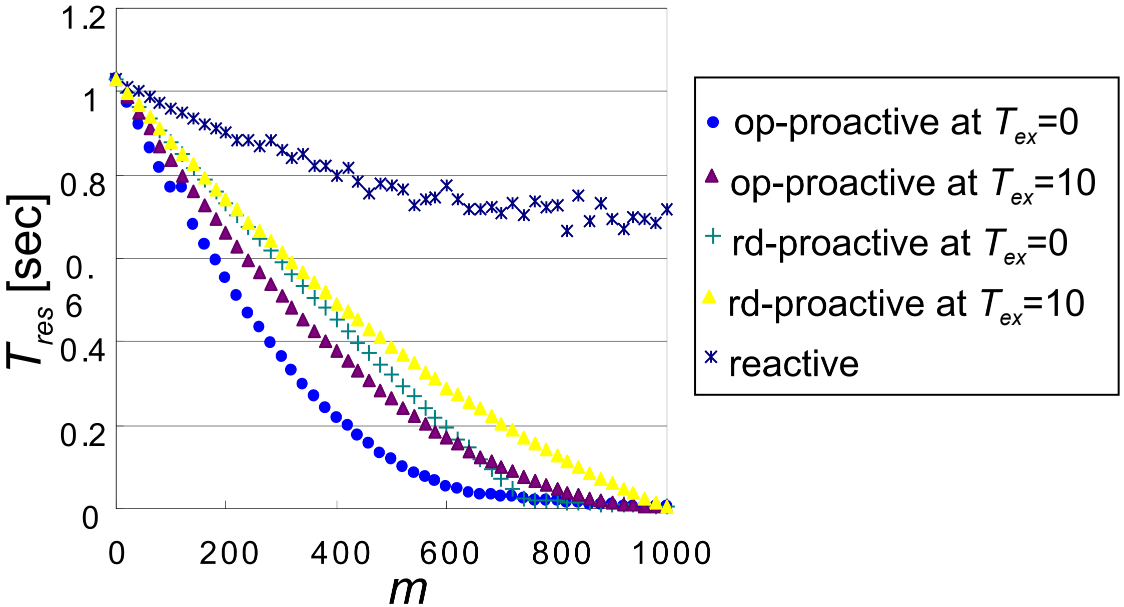
Parameter settings	
R	1000
T_{hit}	0.0046 [sec]
T_{miss}	1.0686 [sec]
s_i	random number of [0-1]
TTL_i	random number of [0-10]
t_{start}	0
t_{end}	200 [sec]
$t_{interval}$	5 [sec]

- The minimum value of T_{res}
  - Op-proactive 0.219 [sec] at $T_{ex} = 0$ ← optimal value
  - Rd-proactive 0.451 [sec] at $T_{ex} = -1$

Empowered by Innovation NEC

Simulation Results –Effect of varying *m* on T_{res}-

- We evaluated the average query response time T_{res} by varying the number of resources whose information is updated per update interval *m* in the range of 0 to 1000



Parameter settings:

R	1000
T_{hit}	0.0046 [sec]
T_{miss}	1.0686 [sec]
s_i	random number of [0-1]
TTL_i	random number of [0-10]
t_{start}	0
t_{end}	200 [sec]
$t_{interval}$	5 [sec]

Ratio of the best case of op-proactive and reactive:

$$\frac{Average(op-proatcive \quad at \quad T_{ex}=0)}{Average(reactive)} = 33.34\%$$

Reduced by 66.66%

Outline

- Introduction
  - Secure Platform Project
- Approach
  - Requirements of Resource Information Service
  - Existing Approaches for Cache Update Control
  - Proposed Solution
- Design
  - Resource Information Manager
  - Resource Information Cache Update Control
- Performance Evaluation
  - Experiment
  - Simulation Studies
- Conclusion

Conclusion

- We proposed an selective cache update control method for the resource information service to reduce average query response time in large-scale systems.
- We implemented the method based on the WS-Management and the CIM standards
- We measured basic query performance of the implementation
- We evaluated average query response time by simulating update control methods
  - With given query request rate and the experimental results
- Average query response time of the proposed method is 66.66% shorter than that of the reactive method

Thank you!

Simulation Studies –Effect of varying *m* on T_{res}-

- We investigate the variation of T_{res} due to:
  1. Adjustment of the T_{ex}
  2. Update target selection methods
  3. Introduction of the proposed method (compared to the existing method)

	Target methods for comparison	Difference of T_{res}				m
		Average [sec]	Average [ratio]	Maximum [sec]	Average [ratio]	
1	op-proactive at T_{ex}=0 and 10	0.083	23.55%	0.159	39.52%	380
2	op-proactive and rd-proactive at T_{ex}=0	0.159	39.52%	0.237	49.39%	380
3	op-proactive at T_{ex}=0 and reactive	0.536	66.66%	0.738	98.31%	840