the globus alliance
www.globus.org

National e-Science Centre

Distributed Monitoring and Information Services for the Grid

Jennifer M. Schopf

UK National eScience Centre

Argonne National Lab

April 27, 2005

My Definitions

- Grid:
  - Shared resources
  - Coordinated problem solving
  - Multiple sites (multiple institutions)
- Monitoring:
  - Discovery
    - > Registry service
    - > Contains descriptions of data that is available
  - Expression of data
    - > Access to sensors, archives, etc.

What do *I* mean by Grid monitoring?

- Grid level monitoring concerns data
  - Shared between administrative domains
  - For use by multiple people
  - Think scalability
- Different levels of monitoring needed:
  - Application specific
  - Node level
  - Cluster/site Level
  - Grid level

Grid Monitoring Does Not Include...

- All the data about every node of every site
- Years of utilization logs to use for planning next hardware purchase
- Low-level application progress details for a single user
- Application debugging data (except perhaps notification of a failure of a heartbeat)
- Point-to-point sharing of all data over all sites

Overview of This Talk

- Evaluation of information infrastructures
  - Globus Toolkit MDS2, R-GMA, Hawkeye
  - Insights into performance issues
- What monitoring and discovery could be
- Next-generation information architecture
  - Web Service Resource Framework (WS-RF) mechanisms
  - Integrated monitoring & discovery architecture for GT4
  - Performance Numbers

Performance and the Grid

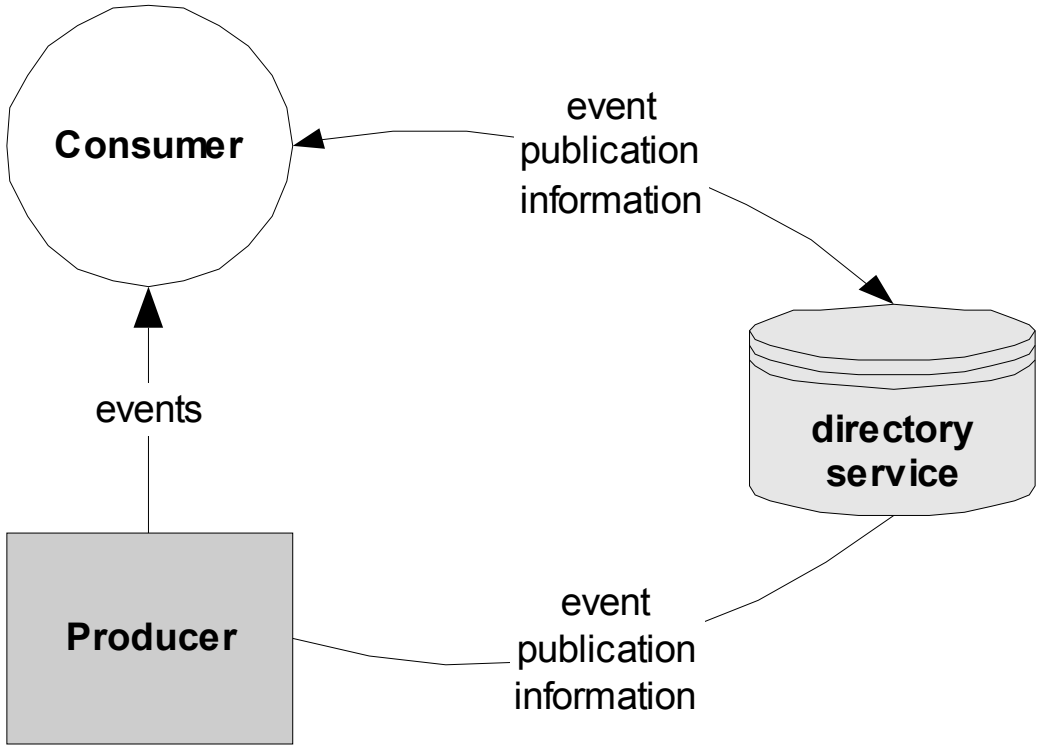
- It's not enough to use the Grid, it has to perform – otherwise, why bother?
- First prototypes rarely consider performance (tradeoff with dev't time)
  - MDS1–centralized LDAP
  - MDS2–decentralized LDAP
  - MDS3–decentralized OGSA Grid service
    - > Prototype of a WS-based monitoring approach
  - MDS4–decentralized WS-RF Web service
- Often performance is simply not known

So We Did Some Performance Analysis

- Common model for our systems
- 3 Monitoring systems
  - Globus Toolkit MDS2
  - EDG's R-GMA
  - Condor's Hawkeye
- Tried to compare apples to apples
- Got some numbers as a starting point

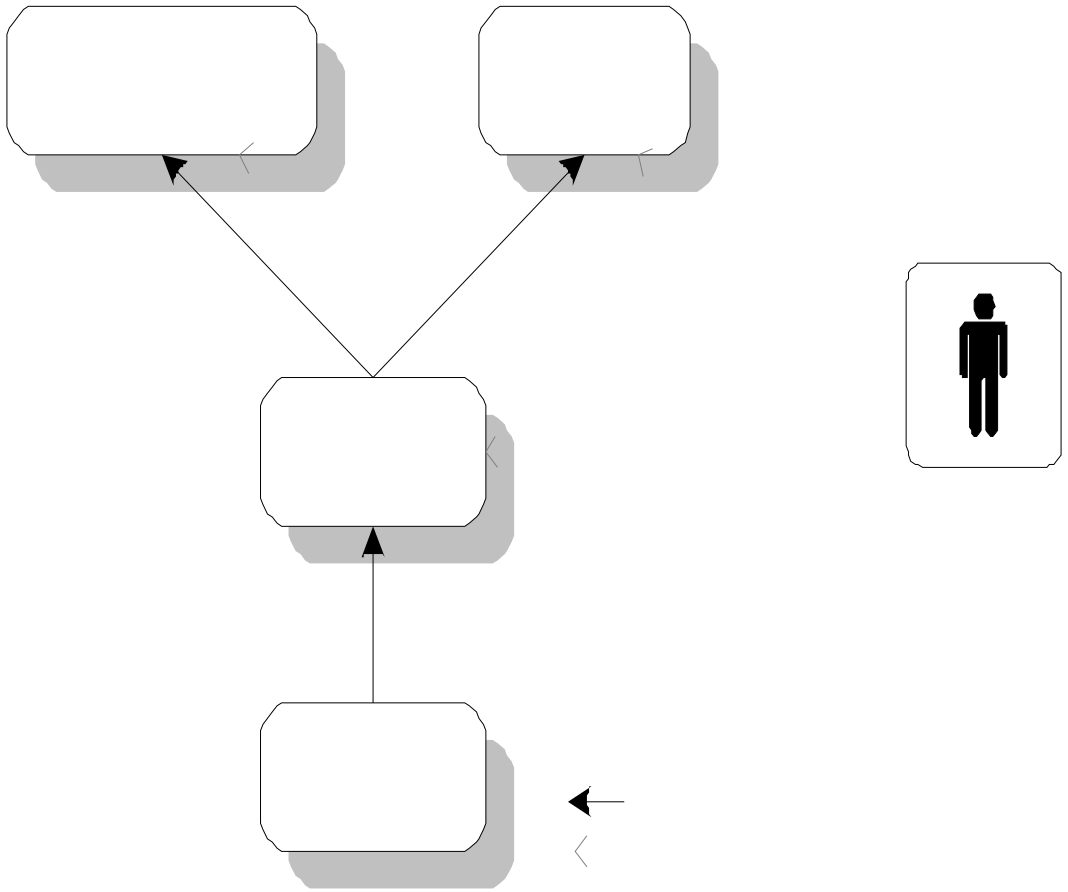
GGF Grid Monitoring Architecture

- Defines only the basic concepts
- No API's, protocols, schema defined
- Every monitoring system I've seen fits this model



Our Generic Model

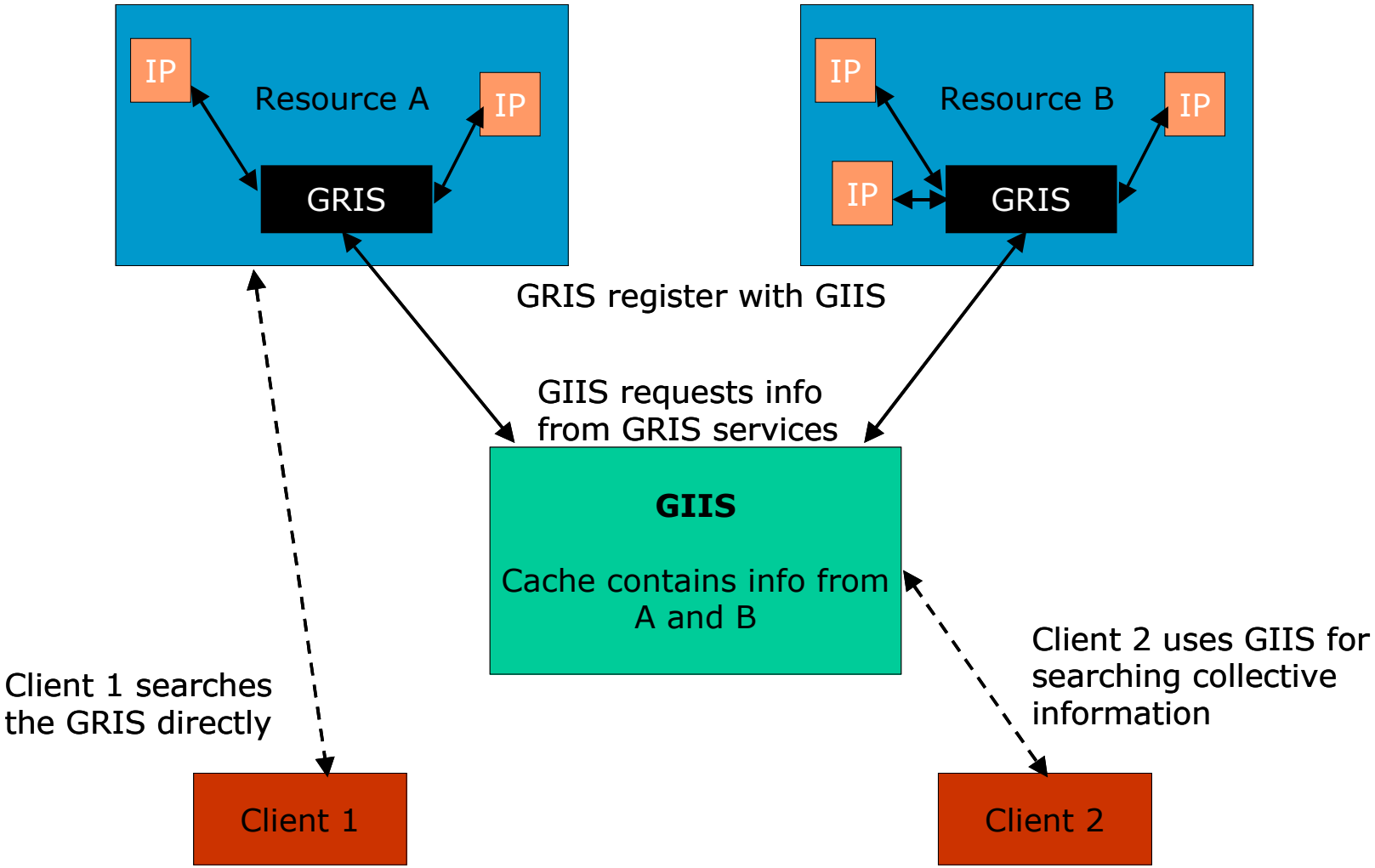
- Defines functionality for our experiments
- No API's, protocols, schema defined
- Can be mapped to GMA



Globus Monitoring and Discovery Service (MDS2)

- Part of Globus Toolkit, compatible with other elements
- Used most often for resource selection
  - aid user/agent to identify host(s) on which to run an application
- Standard mechanism for publishing and discovery
- Decentralized, hierarchical structure
- Soft-state protocols
- Caching (lazy)
- Grid Security Infrastructure credentials

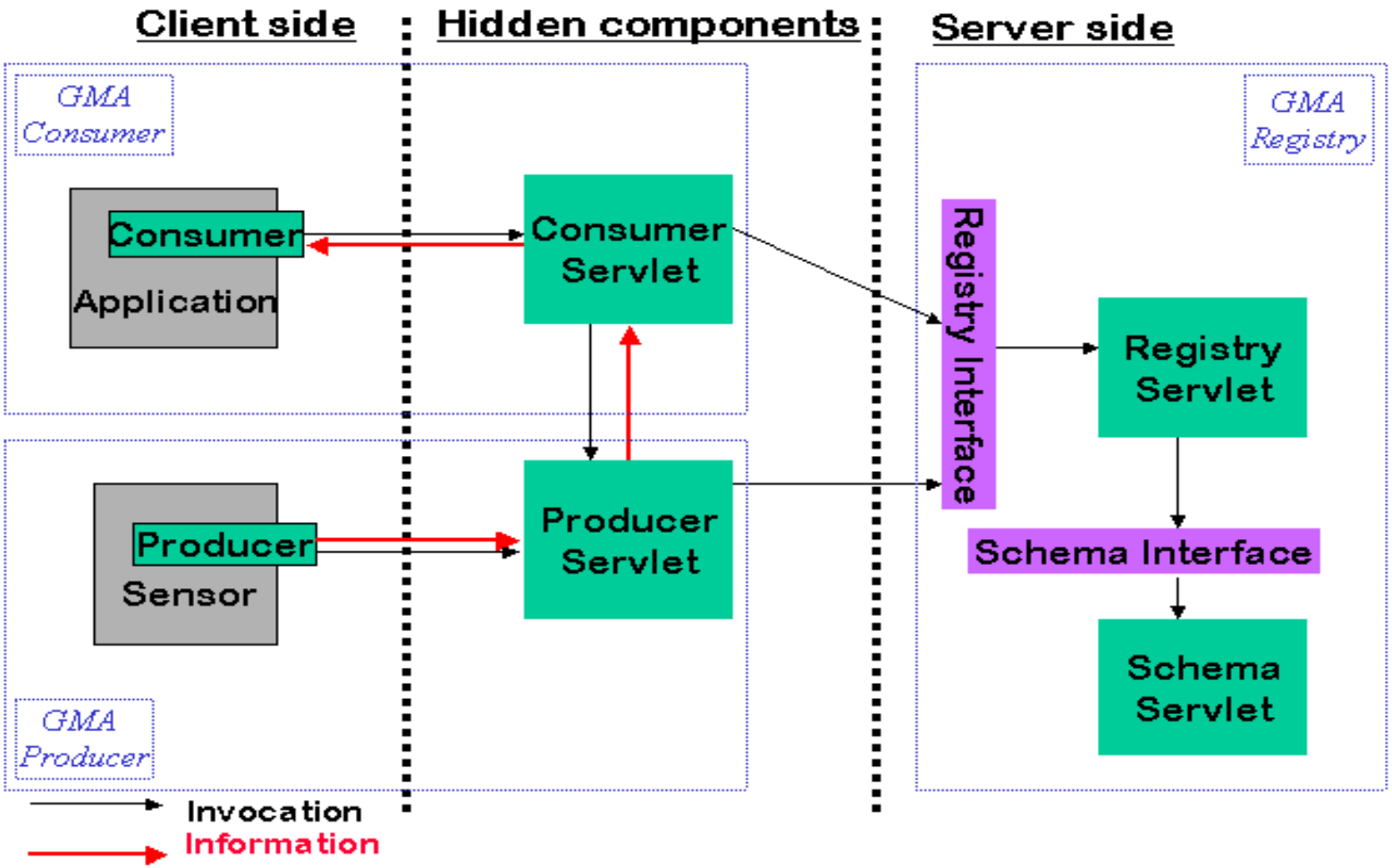
MDS2 Architecture



Relational Grid Monitoring Architecture (R-GMA)

- Monitoring used in the EU Datagrid Project
  - Steve Fisher, RAL, and James Magowan, IBM-UK
- Implementation of the Grid Monitoring Architecture (GMA) defined within the Global Grid Forum (GGF)
- Based on the relational data model
- Used Java Servlet technologies
- Focus on notification of events
- User can subscribe to a flow of data with specific properties directly from a data source

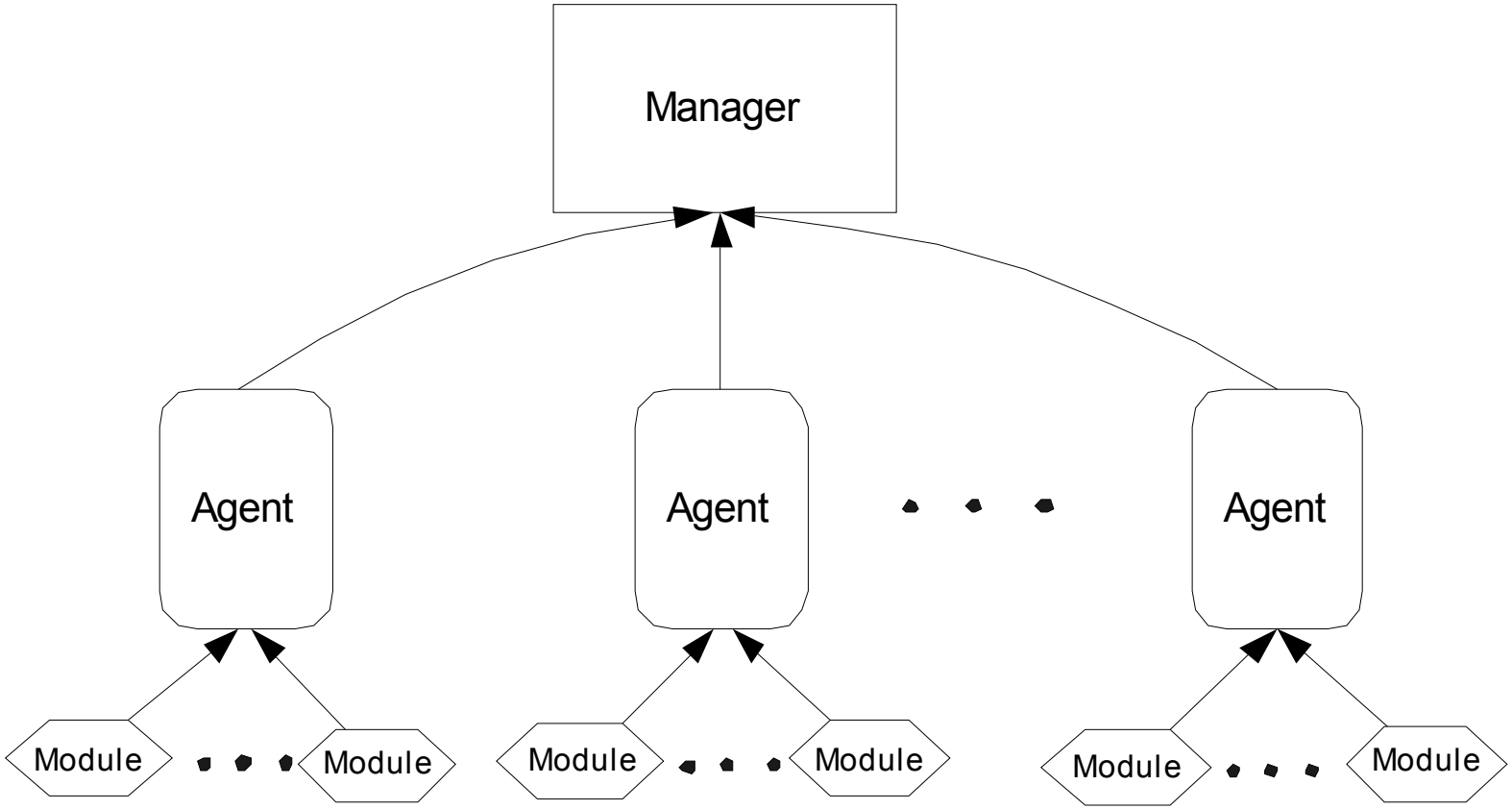
R-GMA Architecture



Hawkeye

- Developed by Condor Group
- Focus – automatic problem detection
- Underlying infrastructure builds on the Condor and ClassAd technologies
  - Condor *ClassAd* Language to identify resources in a pool
  - *ClassAd Matchmaking* to execute jobs based on attribute values of resources to identify problems in a pool
- Passive Caching – updates to Agents done periodically by default

Hawkeye Architecture



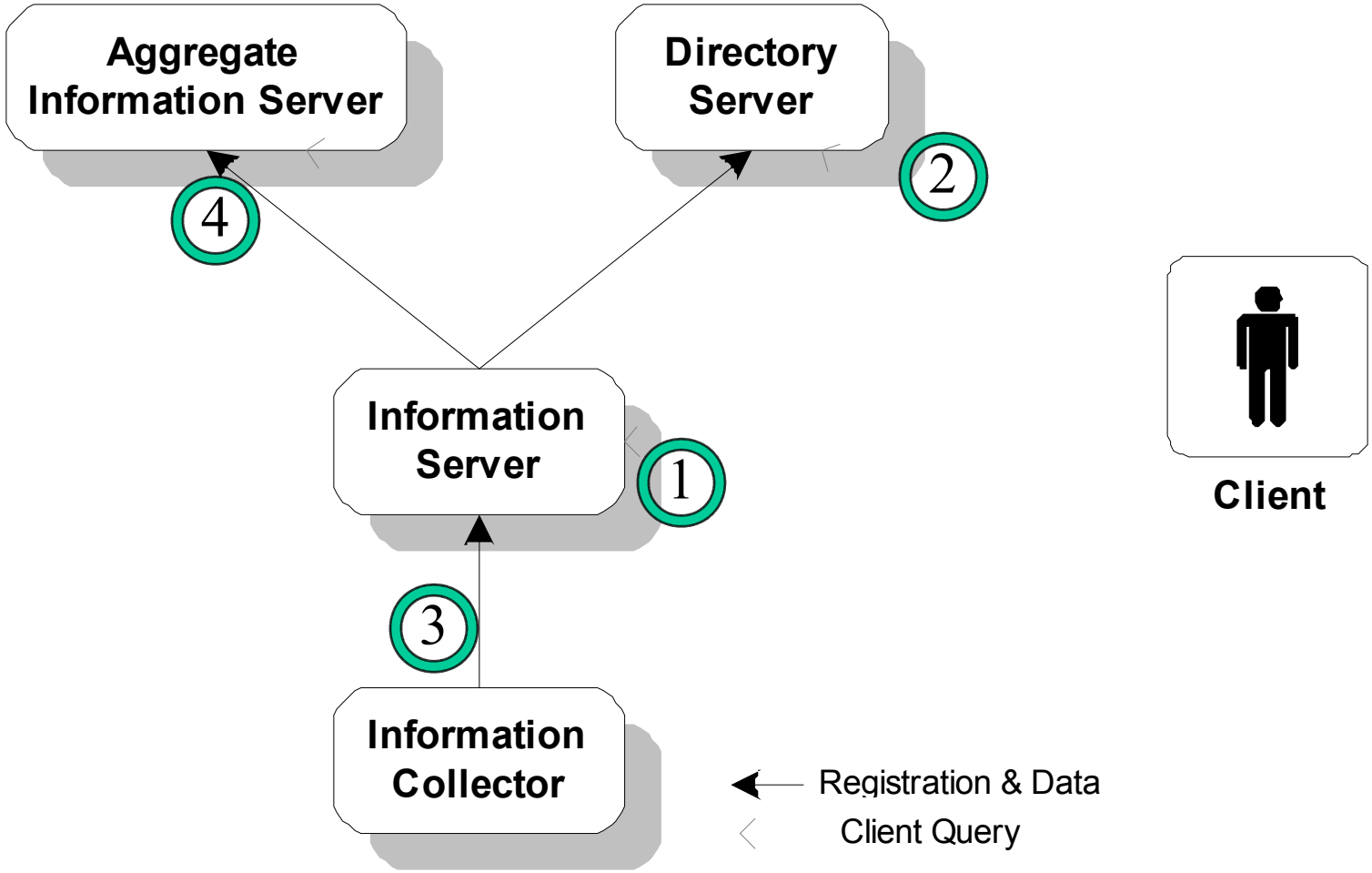
Comparing Information Systems

	MDS2	R-GMA	Hawkeye
Info Collector	Information Provider	Producer	Module
Info Server	GRIS	Producer Servlet	Agent
Aggregate Info Server	GIIS	Combo Producer-Consume	Manager
Directory Server	GIIS	Registry	Manager

Experiments

- How many users can query an information server at a time?
- How many users can query a directory server?
- How does an information server scale with the amount of data in it?
- How does an aggregator scale with the number of information servers registered to it?

Our Experiments



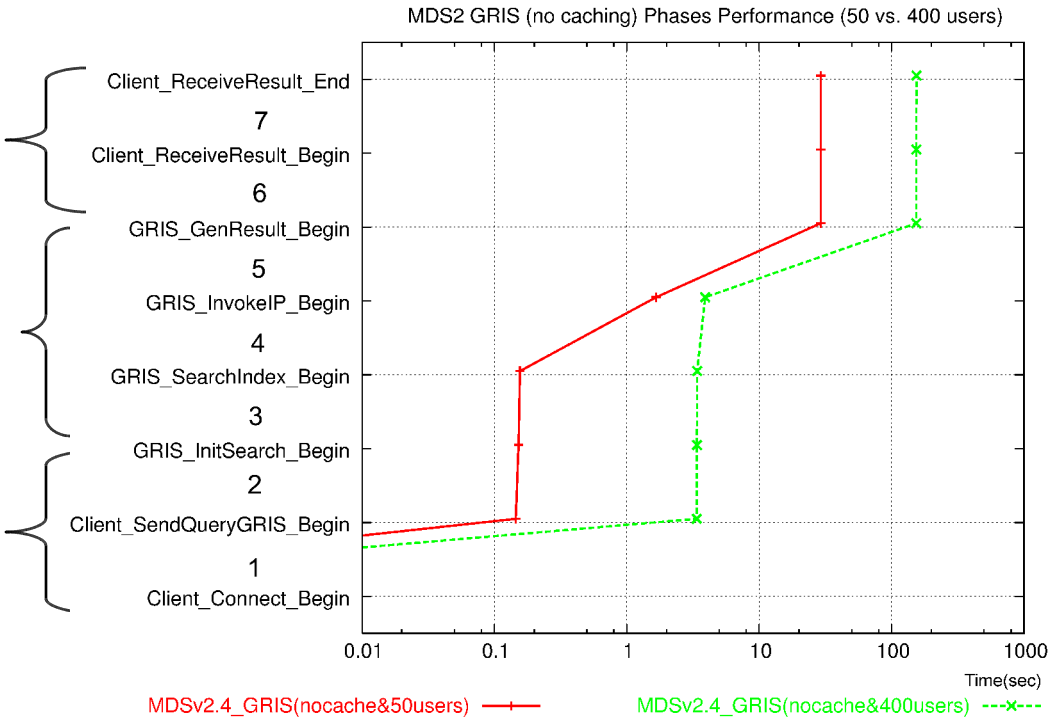
Comparing Information Systems

- We also looked at the queries in depth - NetLogger
- 3 phases
  - Connect, Process, Response

Response

Process

Connect



Some Architecture Considerations

- Similar functional components
  - Grid-wide for MDS2, R-GMA; Pool for Hawkeye
  - Global schema
- Different use cases will lead to different strengths
  - GIIS for decentralized registry; no standard protocol to distribute multiple R-GMA registries
  - R-GMA meant for streaming data – currently used for NW data; Hawkeye and MDS2 for single queries
- Push vs Pull
  - MDS2 is PULL only
  - R-GMA allows push and pull
  - Hawkeye allows triggers – push model

Testbed

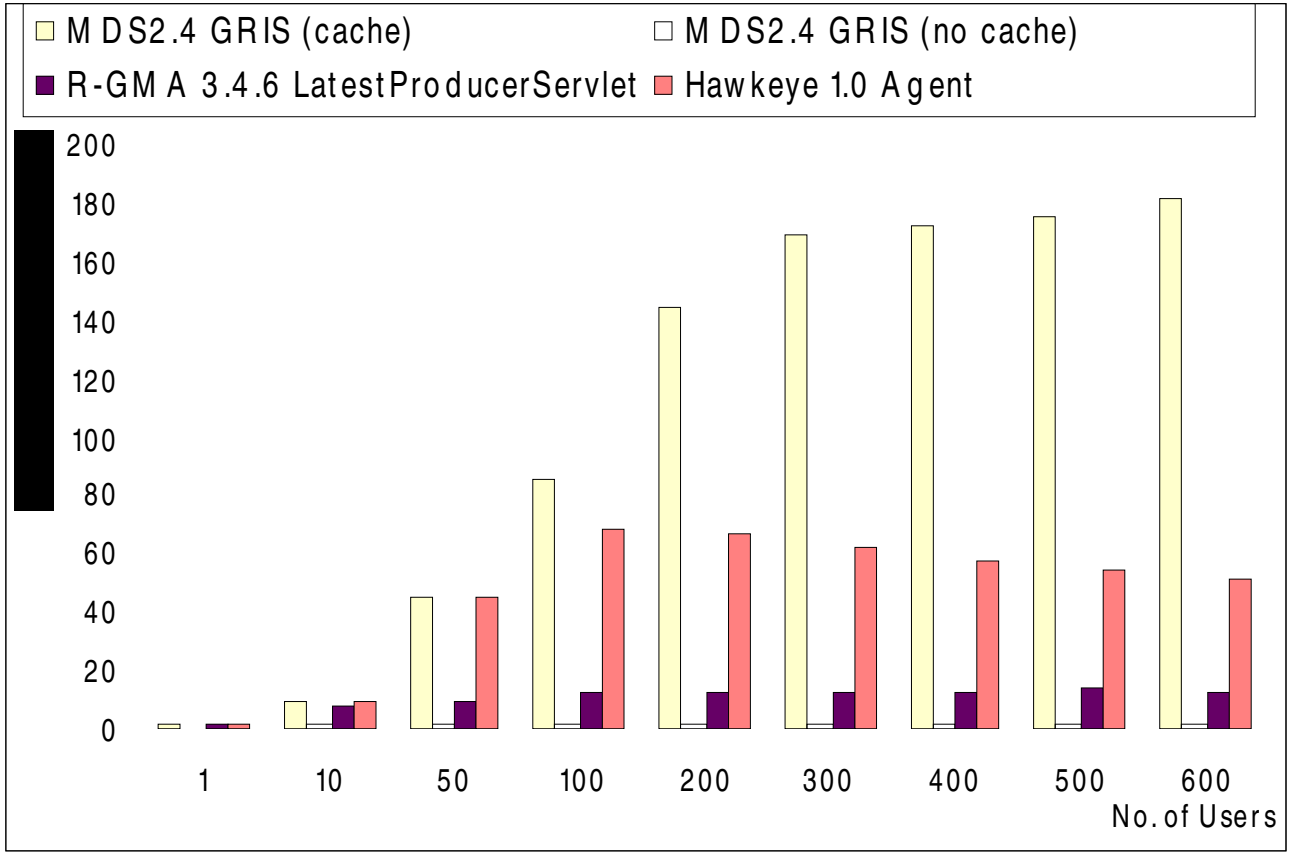
- Lucky cluster at Argonne
  - 7 nodes, each has two 1133 MHz Intel PIII CPUs (with a 512 KB cache) and 512 MB main memory
- Users simulated at the UC nodes
  - 20 P3 Linux nodes, mostly 1.1 GHz
  - R-GMA has an issue with the shared file system, so we also simulated users on Lucky nodes
- All figures are 10 minute averages
- Queries happening with a one second wait between each query (think synchronous send with a 1 second wait)

Metrics

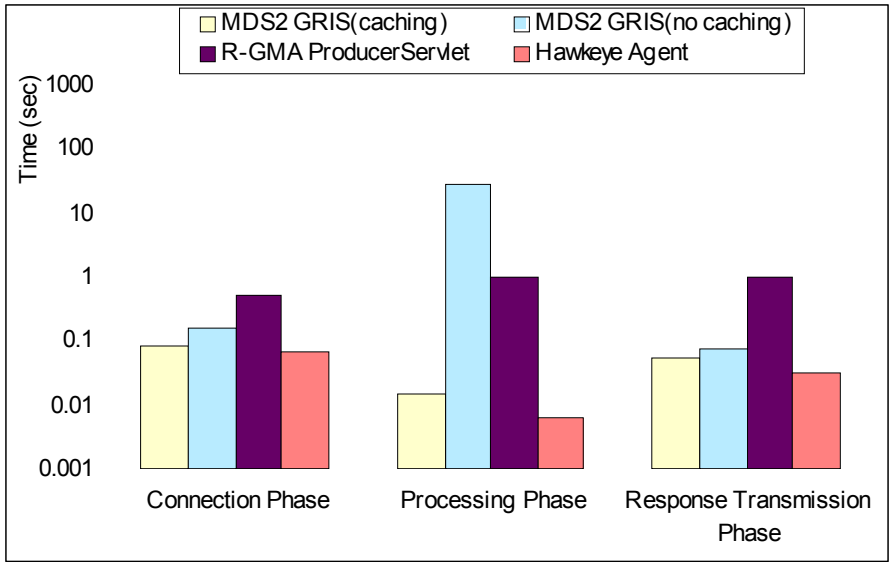
- Throughput
  - Number of requests processed per second
- Response time
  - Average amount of time (in sec) to handle a request
- Load
  - percentage of CPU cycles spent in user mode and system mode, recorded by Ganglia
  - High when running small number compute intensive aps
- Load1
  - average number of processes in the ready queue waiting to run, 1 minute average, from Ganglia
  - High when large number of aps blocking on I/O

Information Server Throughput vs. Number of Users

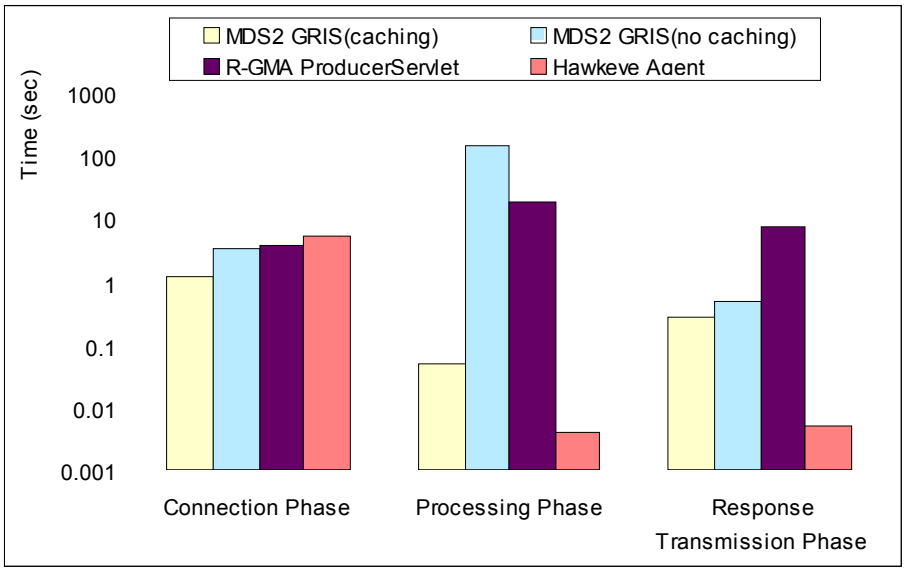
(Larger number is better)



Query Times



50 users



400 users

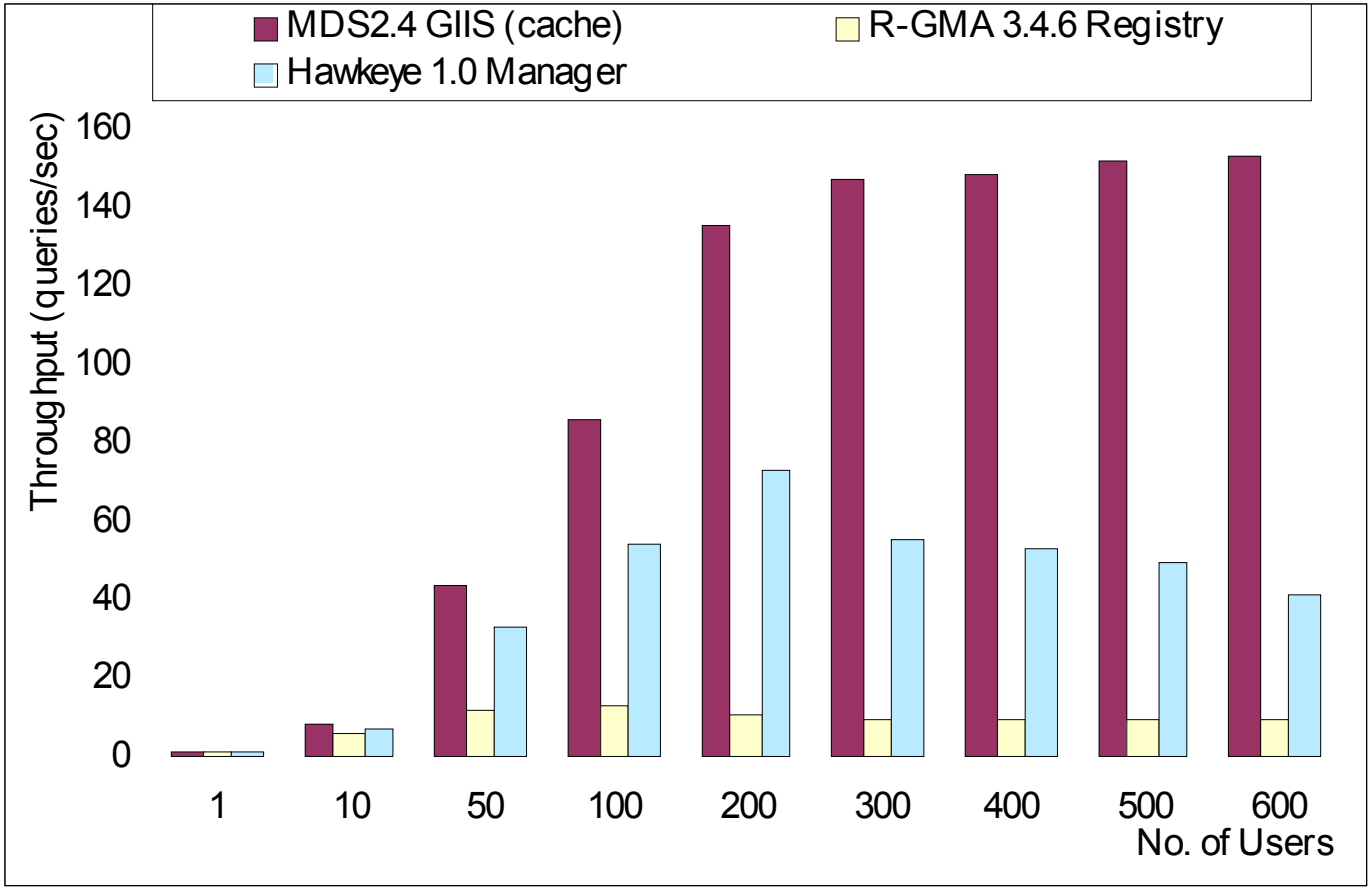
(Smaller number is better)

Experiment 1 Summary

- Caching can significantly improve performance of the information server
  - Particularly desirable if one wishes the server to scale well with an increasing number of users
- When setting up an information server, care should be taken to make sure the server is on a well-connected machine
  - Network behavior plays a larger role than expected
  - If this is not an option, thought should be given to duplicating the server if more than 200 users are expected to query it

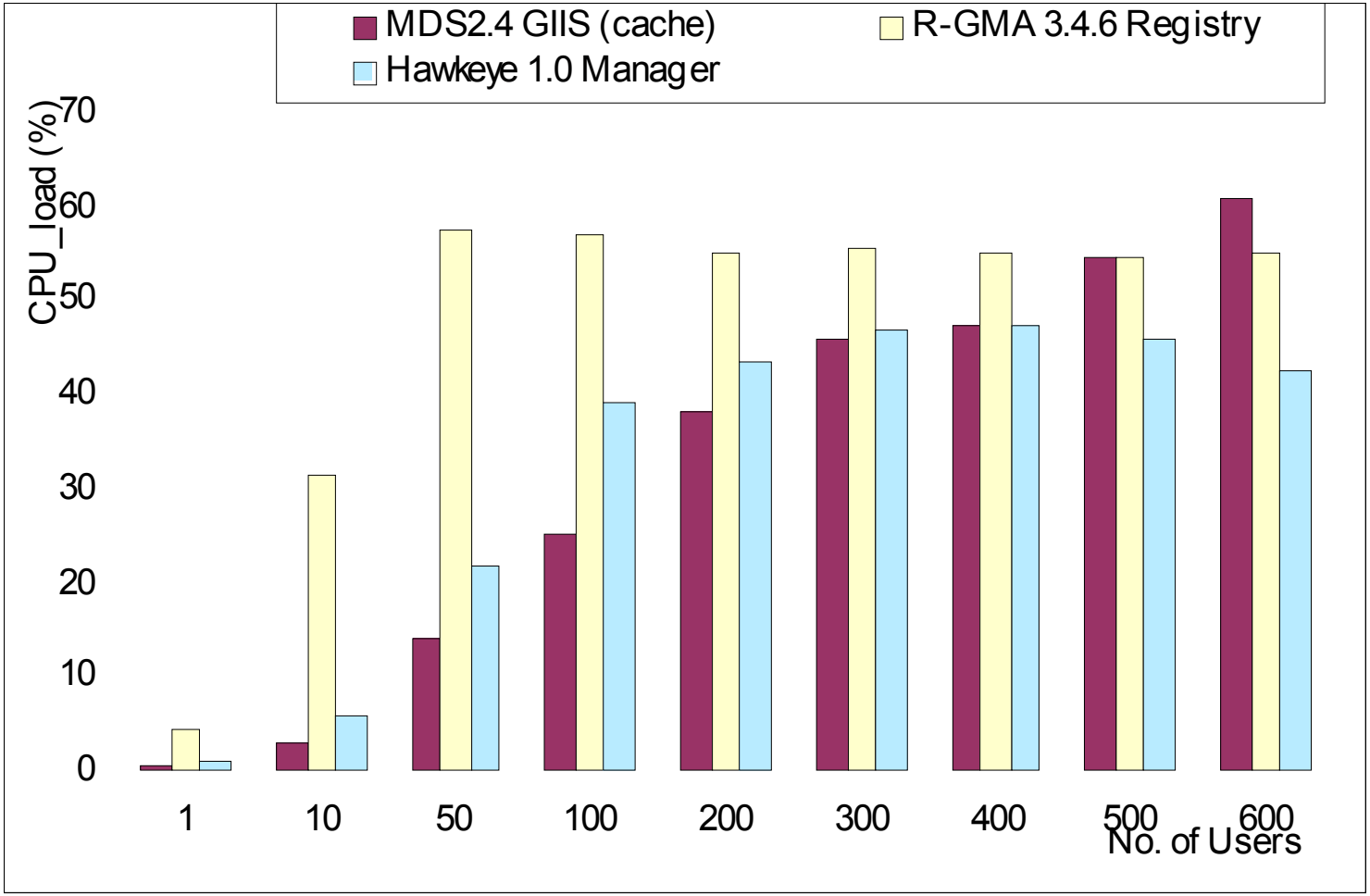
Directory Server Throughput

(Larger number is better)

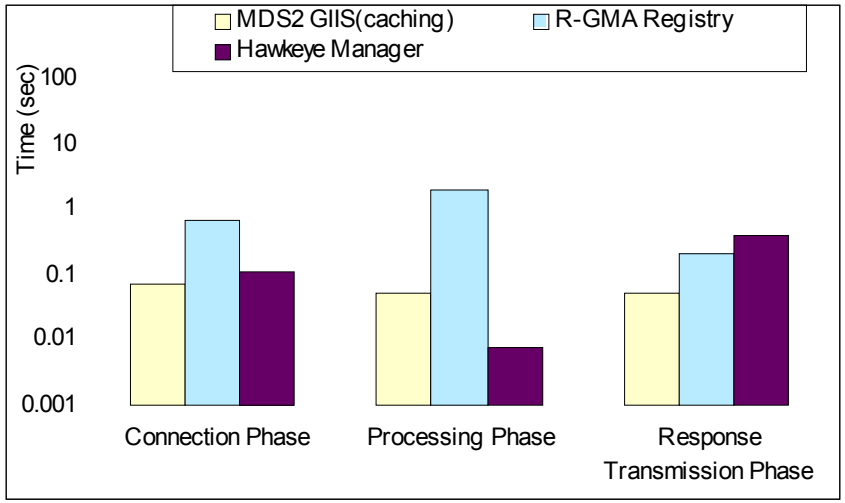


Directory Server CPU Load

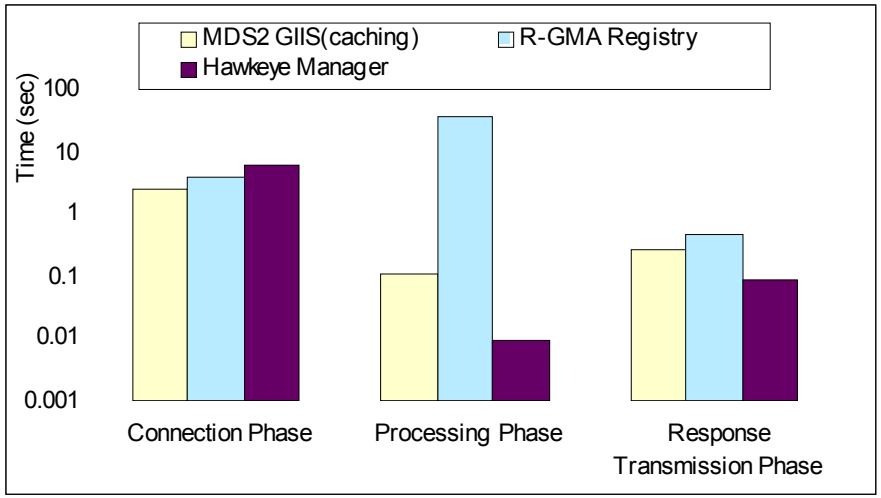
(Smaller number is better)



Query Times



50 users



400 users

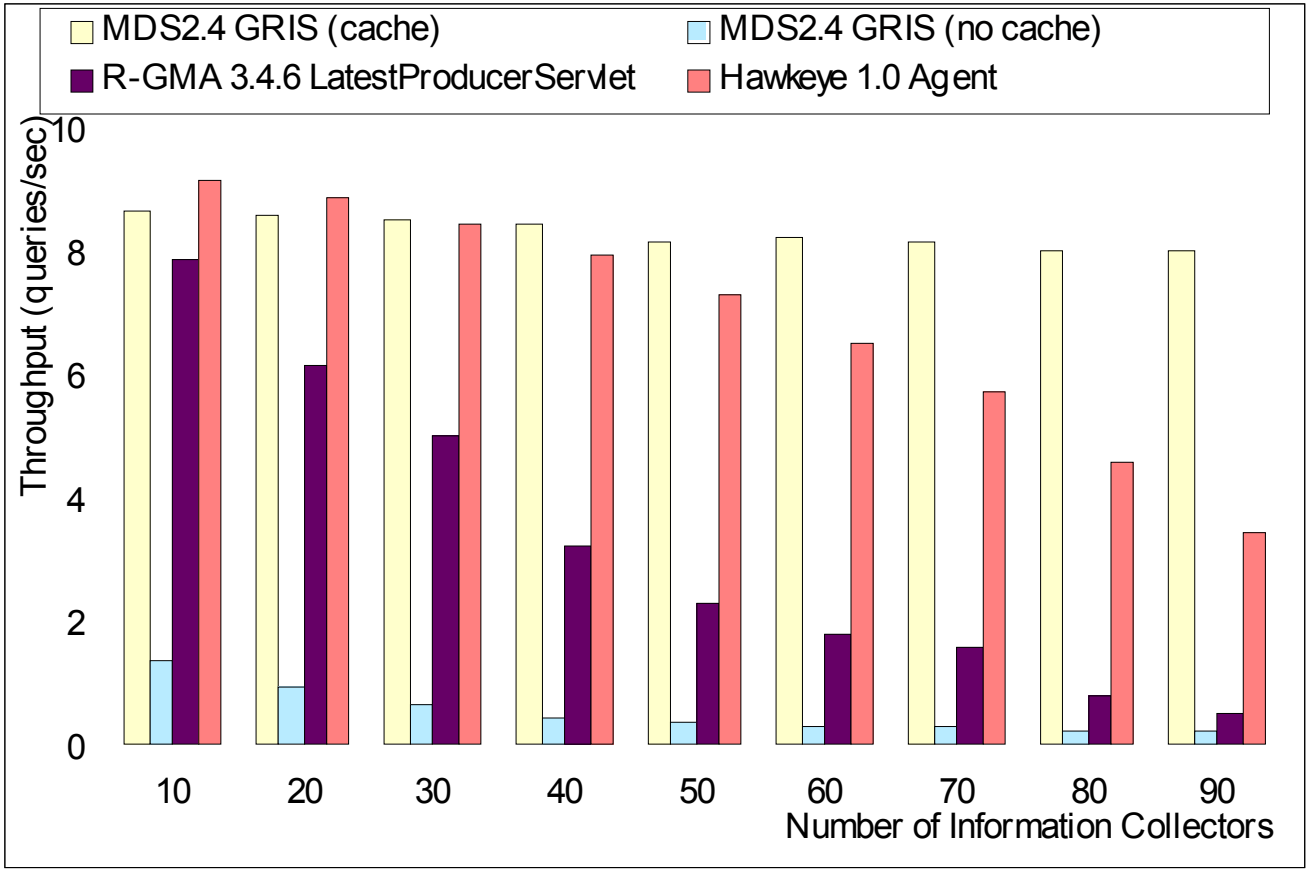
(Smaller number is better)

Experiment 2 Summary

- Because of the network contention issues, the placement of a directory server on a highly connected machine will play a large role in the scalability as the number of users grows
- Significant loads are seen even with only a few users, it will be important that this service be run on a dedicated machine, or that it be duplicated as the number of users grows.

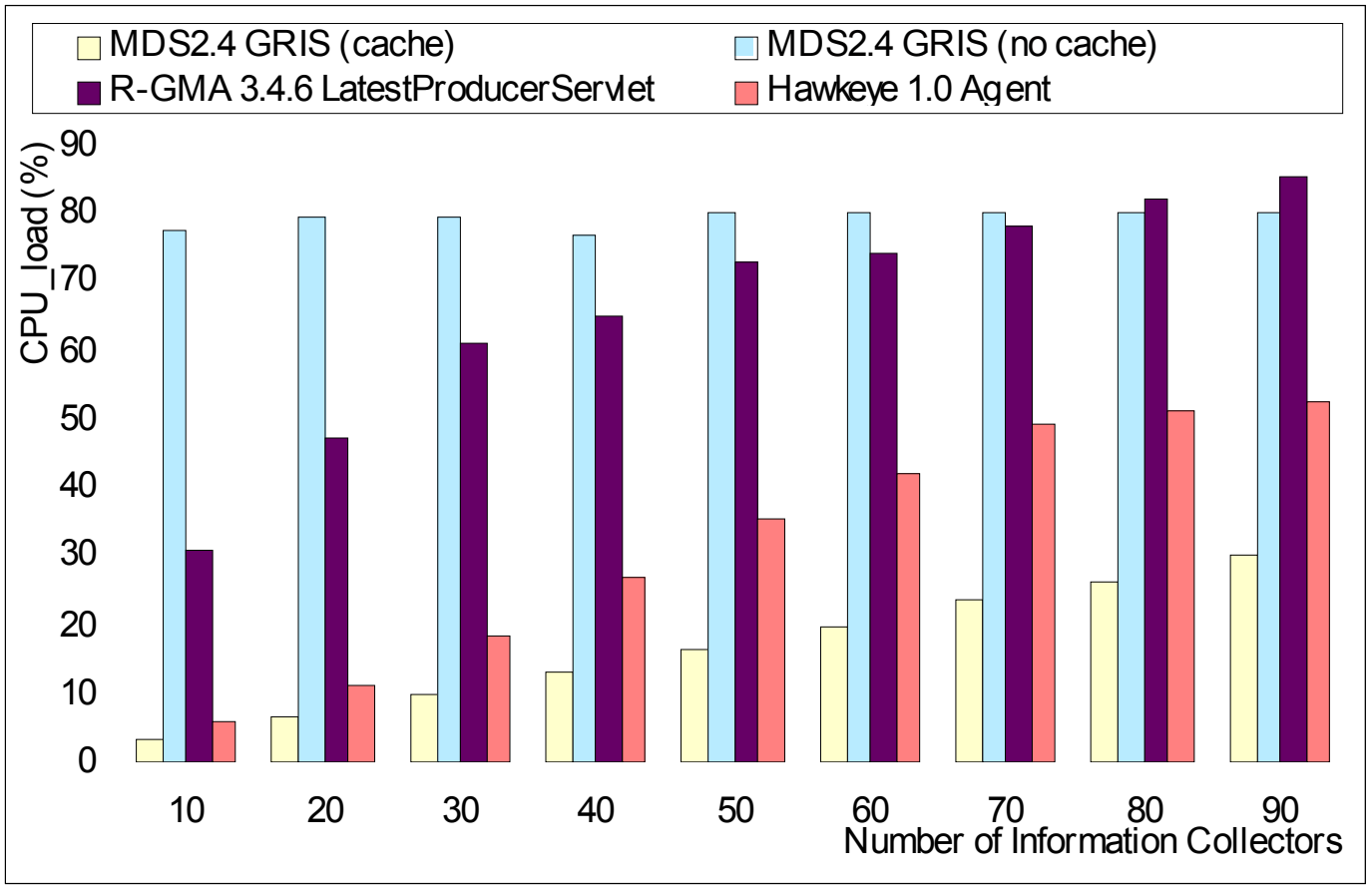
Information Server Scalability with Information Collectors

(Larger number is better)

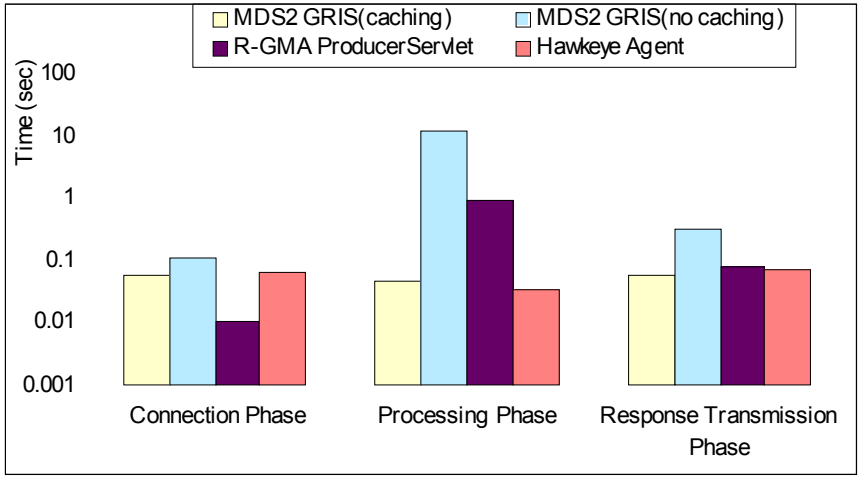


Experiment 3 Load Measurements

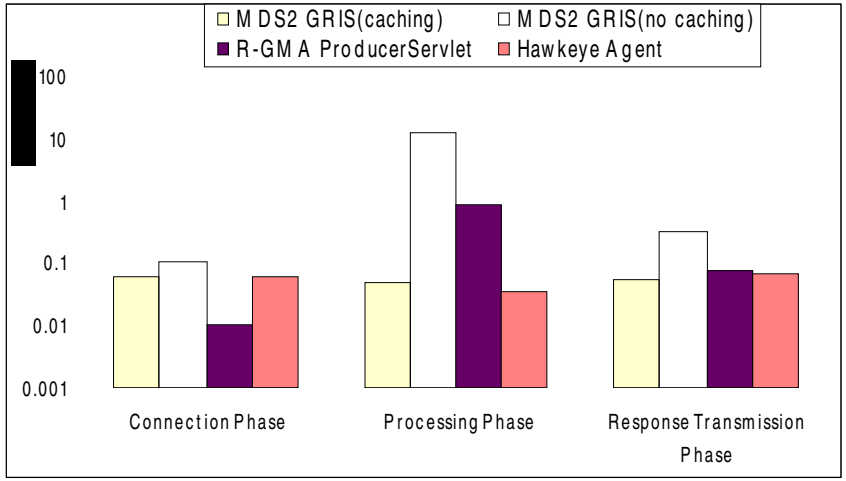
(Smaller number is better)



Experiment 3 Query Times



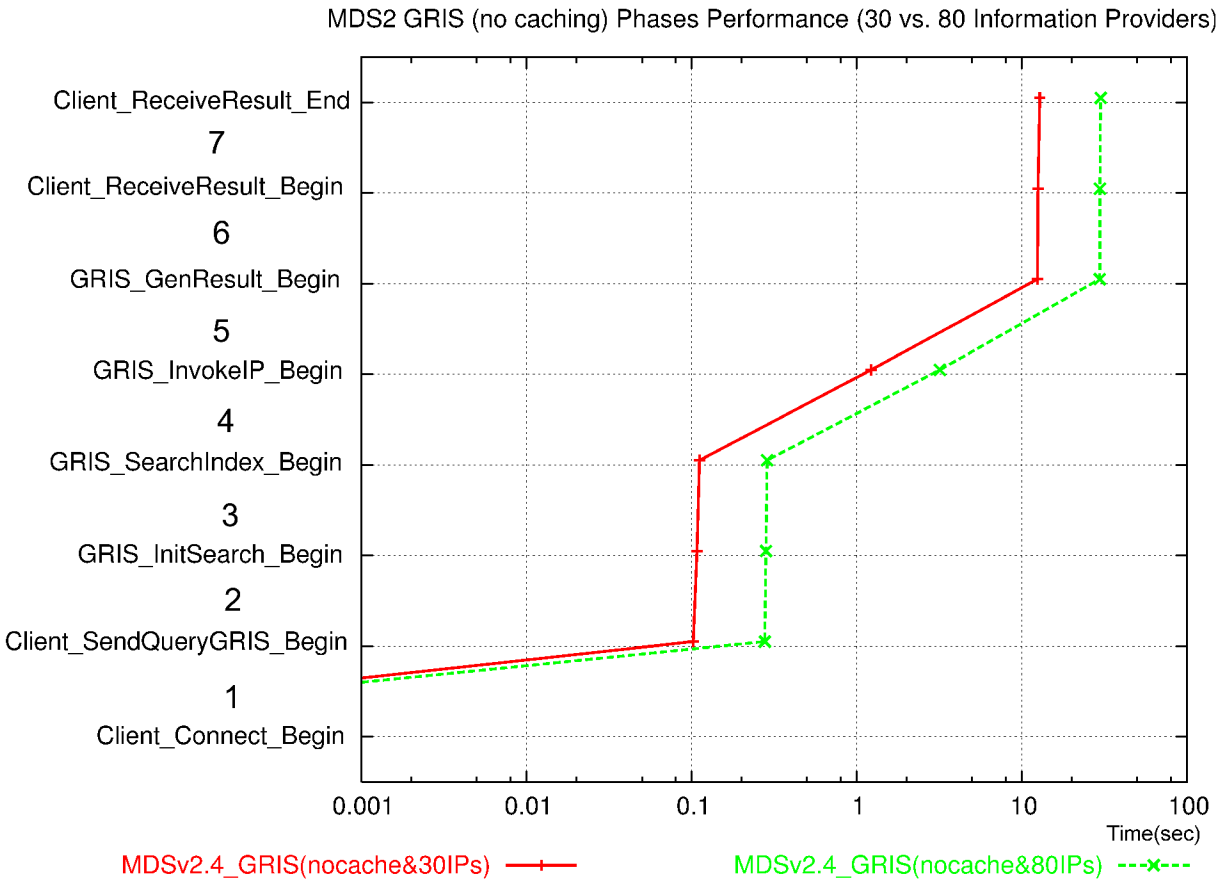
30 Info Collectors



80 Info Collectors

(Smaller number is better)

Sample Query



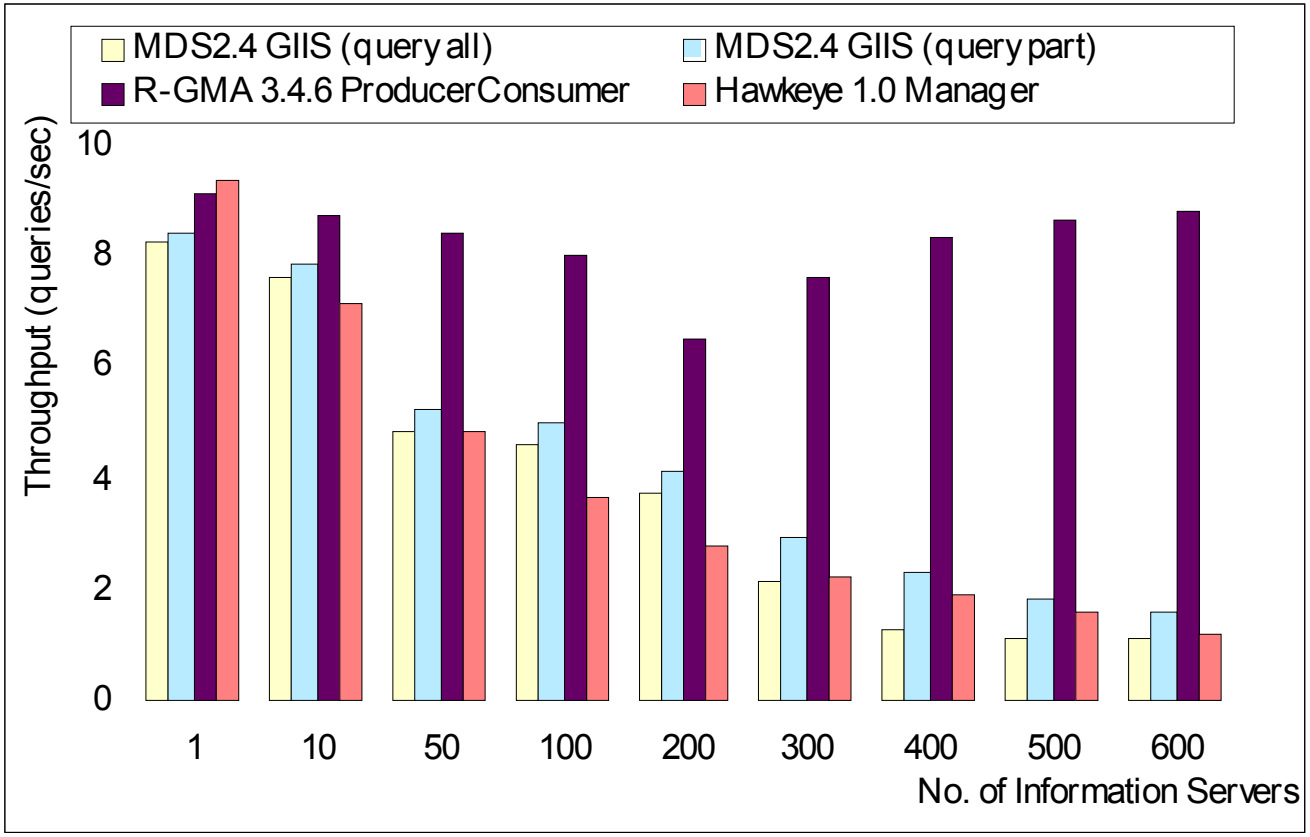
Note: log scale

Experiment 3 Summary

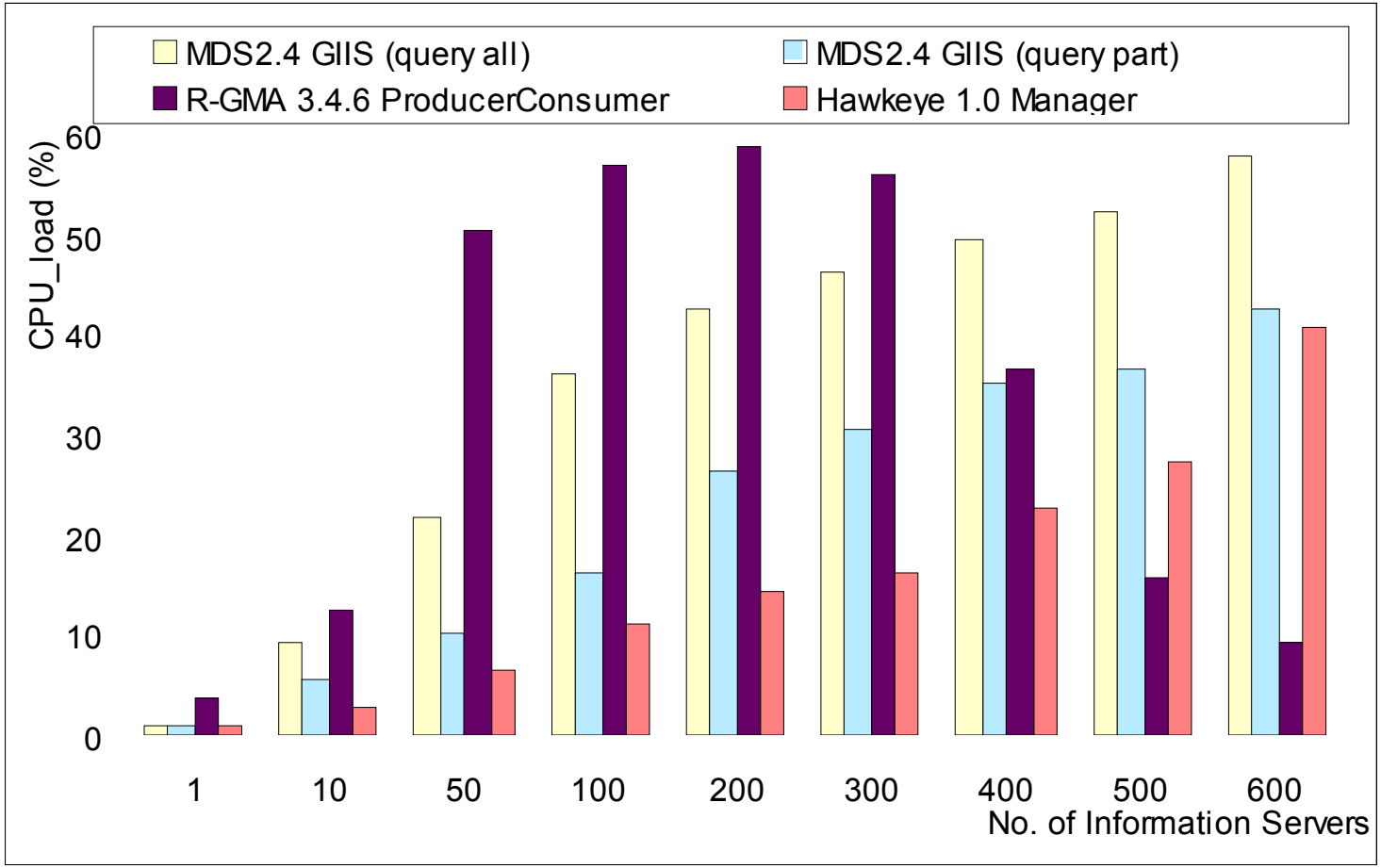
- The more the data is cached, the less often it has to be fetched, thereby increasing throughput
- Search time isn't significant at these sizes

Aggregate Information Server Scalability

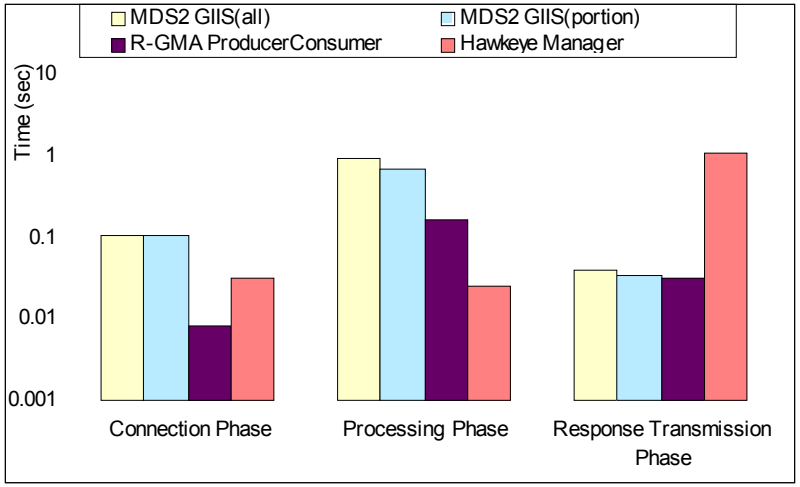
(Larger number is better)



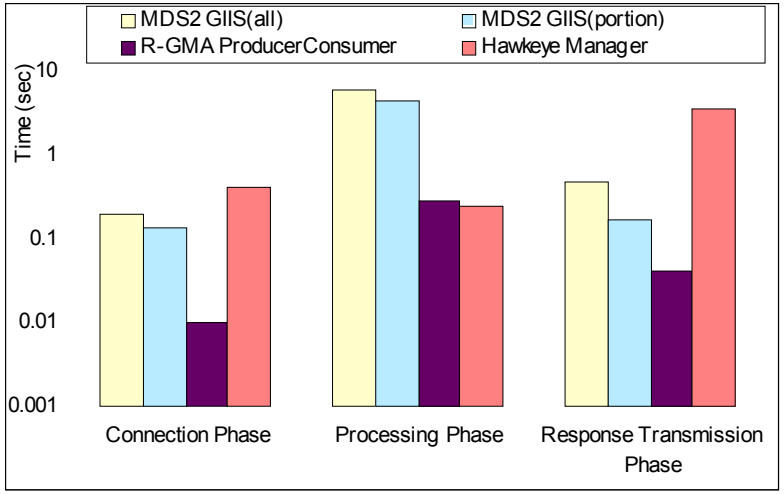
Load


MDS2.4 GIIS (query all)
MDS2.4 GIIS (query part)
R-GMA 3.4.6 ProducerConsumer
Hawkeye 1.0 Manager
CPU_load (%)
0
10
20
30
40
50
60
1
10
50
100
200
300
400
500
600
No. of Information Servers

Query Response Times



50 Info Servers



400 Info Servers

(Smaller number is better)

Experiment 4 Summary

- None of the Aggregate Information Servers scaled well with the number of Information Servers registered to them
- When building hierarchies of aggregation, they will need to be rather narrow and deep having very few Information Servers registered to any one Aggregate Information Server.

Overall Results

- Performance can be a matter of deployment
  - Effect of background load
  - Effect of network bandwidth
- Performance can be affected by underlying infrastructure
  - LDAP/Java strengths and weaknesses
- Performance can be improved using standard techniques
  - Caching; multi-threading; etc.

So what could monitoring be?

- Basic functionality
  - Push and pull (subscription and notification)
  - Aggregation and Caching
- More information available
  - All services should be monitor-able automatically
- More higher-level services
  - Triggers like Hawkeye
  - Viz of archive data like Ganglia
- Plug and Play
  - Well defined protocols, interfaces and schemas
- Performance considerations
  - Easy searching
  - Keep load off of clients

Topics

- Evaluation of information infrastructures
  - Globus Toolkit MDS2, RGMA, Hawkeye
  - Throughput, response time, load
  - Insights into performance issues
- What monitoring and discovery could be
- **Next-generation information architecture**
  - Web Service Resource Framework (WS-RF) mechanisms
  - Integrated monitoring & discovery architecture for GT4
  - Performance

Web Service Resource Framework (WS-RF)

- Defines standard interfaces and behaviors for distributed system integration, especially (for us):
  - Standard XML-based service information model
  - Standard interfaces for push and pull mode access to service data
    - > Notification and subscription

MDS4 Uses Web Service Standards

- WS-ResourceProperties
  - Defines a mechanism by which Web Services can describe and publish resource properties, or sets of information about a resource
  - Resource property types defined in service's WSDL
  - Resource properties can be retrieved using WS-ResourceProperties query operations
- WS-BaseNotification
  - Defines a subscription/notification interface for accessing resource property information
- WS-ServiceGroup
  - Defines a mechanism for grouping related resources and/or services together as service groups

MDS4
Monitoring and Discovery System

- Higher level services
  - Index Service
  - Trigger Service
  - Common aggregator framework
- Information providers
  - Monitoring is a part of every WSRF service
  - Non-WS services can also be used
- Clients
  - WebMDS

- All of the tool are schema-agnostic, but interoperability needs a well-understood common language

MDS4 Index Service

- Index Service is both registry and cache
- Subscribes to information providers
  - Data, datatype, data provider information
- Caches last value of all data
- In memory default approach

Index Service Facts 1

- No single global Index provides information about every resource on the Grid
  - Hierarchies or special purpose index's are common
  - Each virtual organization will have different policies on who can access its resources
  - No person in the world is part of every VO!
- The presence of a resource in an Index makes no guarantee about the availability of the resource for users of that Index
  - Ultimate decision about whether the resources is left to direct negotiation between user and resource
  - MDS does not need to keep track of policy information (something that is hard to do concisely)
  - Rscs do not need to reveal their policies publicly

Index Service Facts 2

- MDS has a soft consistency model
  - Published information is recent, but not guaranteed to be the absolute latest
  - Load caused by information updates is reduced at the expense of having slightly older information
  - Free disk space on a system 5 minutes ago rather than 2 seconds ago.
- Each registration into an Index Service is subject to soft-state lifetime management
  - Reg's have expiry times and must be periodically renewed
  - Index is self-cleaning, since outdated entries disappearing automatically

MDS4 Trigger Service

- Subscribe to a set of resource properties
- Evaluate that data against a set of pre-configured conditions (triggers)
- When a condition matches, email is sent to pre-defined address

- Similar functionality in Hawkeye
- GT3 tech-preview version in use by ESG

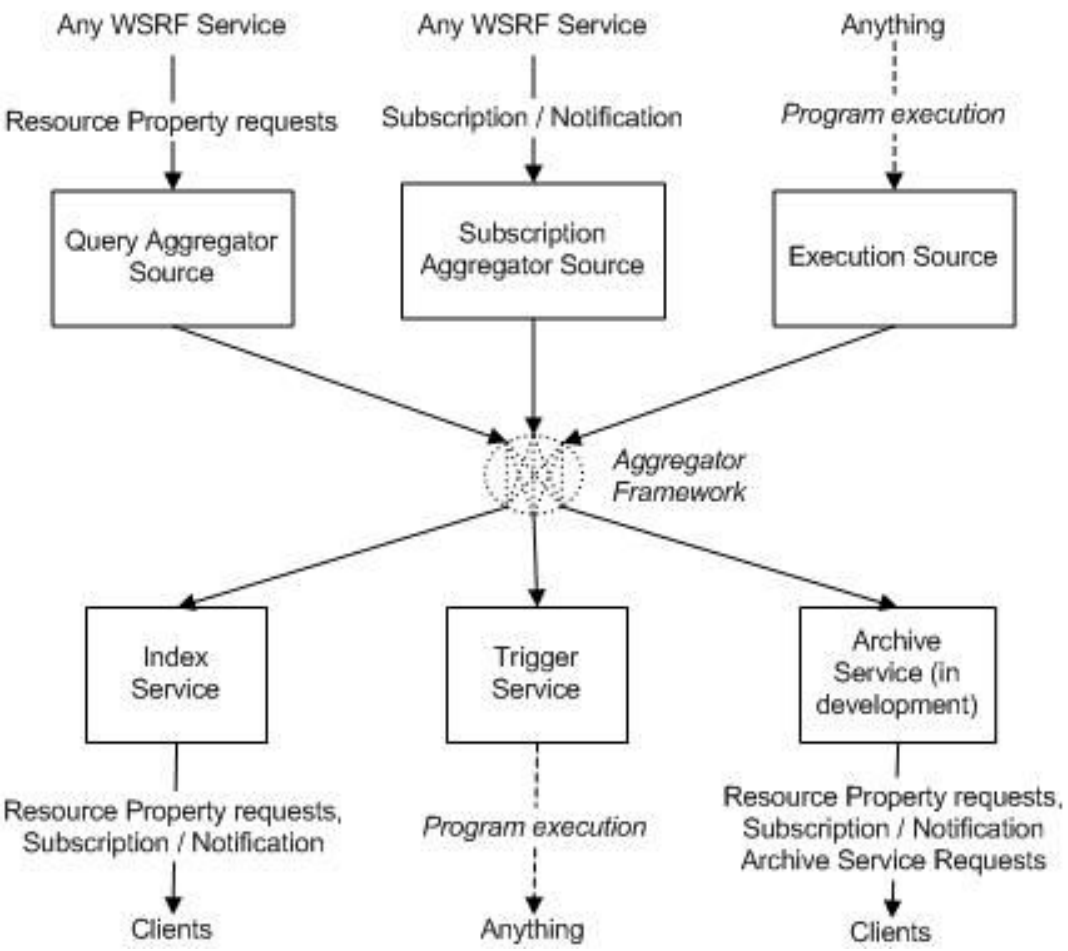
Aggregator Framework

- General framework for building services that collect and aggregate data
  - Index and Trigger service both use this
- 1) Collect information via aggregator sources
  - Java class that implements an interface to collect XML-formatted data
  - *Query source* uses WS-ResourceProperty mechanisms to poll a WSRF service
  - *Subscription source* collects data from a service via WS-Notification subscription/notification
  - *Execution source* executes an administrator-supplied program to collect information

Aggregator Framework (cont)

- 2) Common configuration mechanism
  - Maintain information about which aggregator sources to use and their associated parameters
  - Specify what data to get, and from where
- 3) Aggregator services are self-cleaning
  - Each registration has a lifetime
  - If a registration expires without being refreshed, it and its associated data are removed from the server.

Aggregator Framework



Information Providers

- Data sources for any aggregator service (eg. Index, Trigger)
- WSRF-compliant service
  - WS-ResourceProperty for Query source
  - WS-Notification mechanism for Subscription source
- Other services/data sources
  - Executable program that obtains data via some domain-specific mechanism for Execution source.

Information Providers: Cluster Data

- Interfaces to both Hawkeye and Ganglia
  - Not WS so these are Execution Sources
  - Basic host data (name, ID), processor information, memory size, OS name and version, file system data, processor load data
  - Some condor/cluster specific data
- GRAM – GT4 Job Submission Interface
  - Queue information, number of CPUs available and free, job count information, some memory statistics and host info for head node of cluster

Information Providers: GT4 Services

- Reliable File Transfer Service (RFT)
  - Service status data, number of active transfers, transfer status, information about the resource running the service
- Community Authorization Service (CAS)
  - Identifies the VO served by the service instance
- Replica Location Service (RLS)
  - Note: not a WS
  - Location of replicas on physical storage systems (based on user registrations) for later queries
- Every WS built using GT4 core
  - ServiceMetaDataInfo element includes start time, version, and service type name

WebMDS

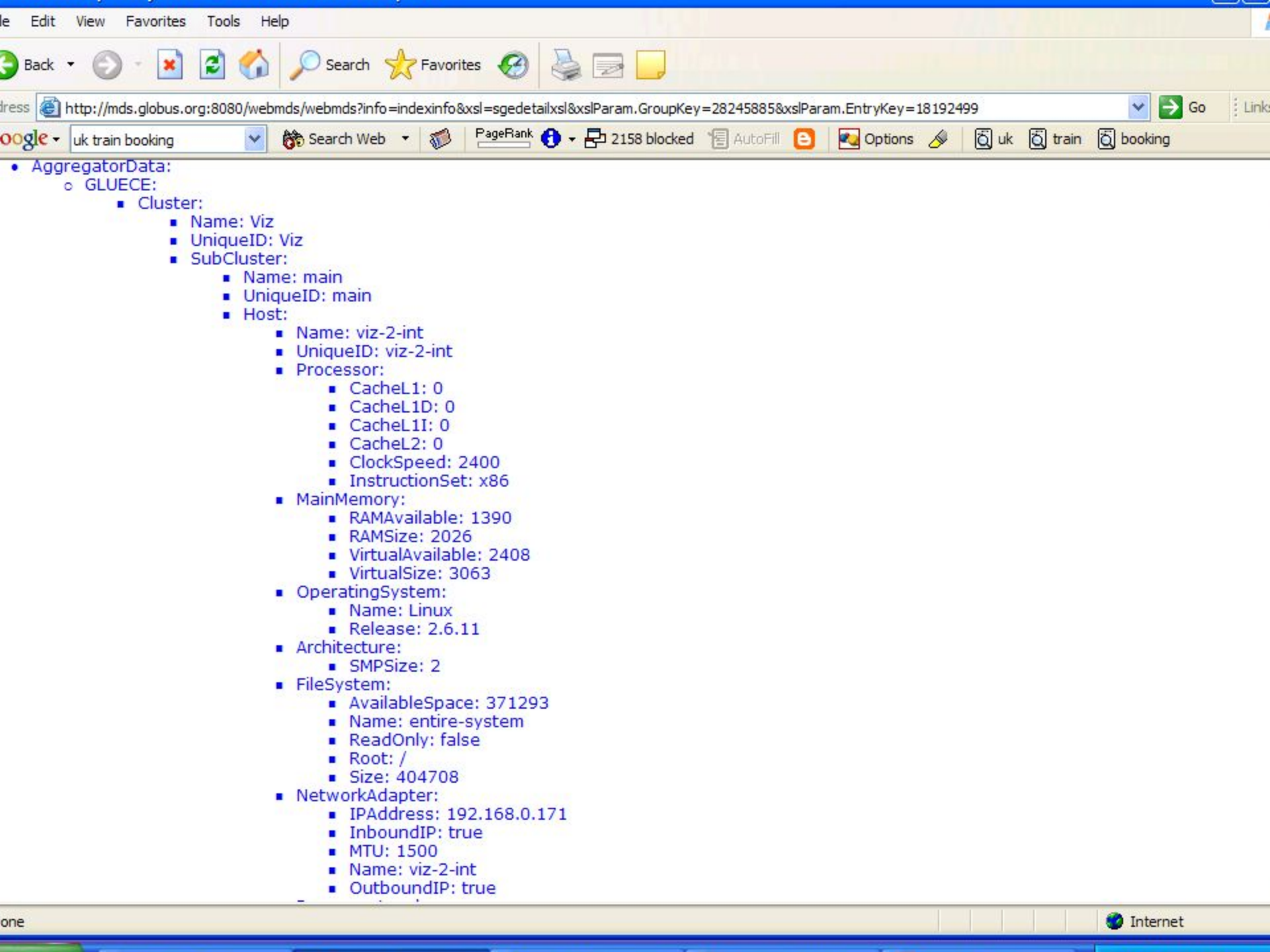
- Web-based interface to WSRF resource property information
- User-friendly front-end to the Index Service
- Uses standard resource property requests to query resource property data
- XSLT transforms to format and display them
- Customized pages are simply done by using HTML form options and creating your own XSLT transforms

erviceGroup Overview

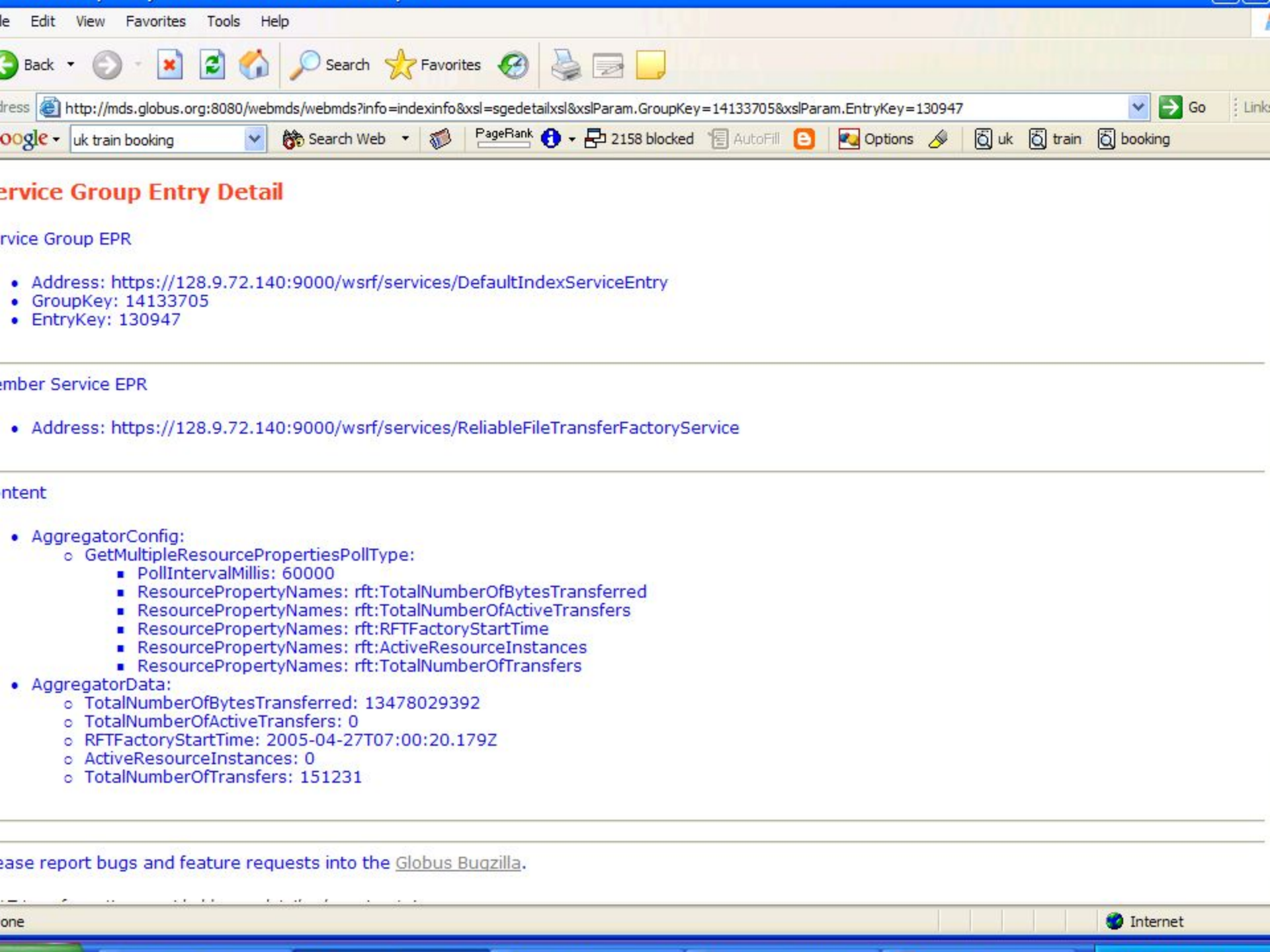
s page provides a brief overview of Web Services and/or WS-Resources that are members of a WS-ServiceGroup.

s WS-ServiceGroup has 4 direct entries, 33 in whole hierarchy.

esource Type	ID	Information	
Unknown	128.9.72.106	Aggregator entry with no content from `https://128.9.72.106:8443/wsrf/services/ReliableFileTransferFactoryService`	detail
GRAM	128.9.72.106	0 queues, submitting to 0 cluster(s) of 0 host(s).	detail
erviceGroup	128.9.72.140	This WS-ServiceGroup has 11 direct entries, 29 including descendants.	detail
erviceGroup	128.9.72.178	This WS-ServiceGroup has 4 direct entries, 4 including descendants.	detail
RFT	128.9.72.178	0 active transfer resources, transferring 0 files. 40.55 GB transferred in 173769 files since start of database.	detail
GRAM	128.9.72.178	0 queues, submitting to 1 cluster(s) of 10 host(s).	detail
GRAM	128.9.72.178	1 queues, submitting to 1 cluster(s) of 10 host(s).	detail
GRAM	128.9.72.178	2 queues, submitting to 1 cluster(s) of 10 host(s).	detail
erviceGroup	128.9.72.106	This WS-ServiceGroup has 3 direct entries, 3 including descendants.	detail
GRAM	128.9.72.106	0 queues, submitting to 0 cluster(s) of 0 host(s).	detail
GRAM	128.9.72.106	1 queues, submitting to 0 cluster(s) of 0 host(s).	detail
RFT	128.9.72.106	0 active transfer resources, transferring 0 files. 8.28 GB transferred in 8595 files since start of database.	detail
erviceGroup	128.9.64.179	This WS-ServiceGroup has 4 direct entries, 4 including descendants.	detail
GRAM	128.9.64.179	1 queues, submitting to 1 cluster(s) of 15 host(s).	detail
GRAM	128.9.64.179	5 queues, submitting to 1 cluster(s) of 15 host(s).	detail
RFT	128.9.64.179	0 active transfer resources, transferring 0 files. 63.16 GB transferred in 108704 files since start of database.	detail
GRAM	128.9.64.179	0 queues, submitting to 1 cluster(s) of 15 host(s).	detail
erviceGroup	128.9.128.168	This WS-ServiceGroup has 3 direct entries, 3 including descendants.	detail
GRAM	128.9.128.168	0 queues, submitting to 0 cluster(s) of 0 host(s).	detail
RFT	128.9.128.168	0 active transfer resources, transferring 0 files. 10.52 GB transferred in 23489 files since start of database.	detail


Edit View Favorites Tools Help
Back Search Favorites
http://mds.globus.org:8080/webmds/webmds?info=indexinfo&xsl=sgedetailxsl&xslParam.GroupKey=28245885&xslParam.EntryKey=18192499
Go
uk train booking
Search Web
PageRank
2158 blocked
AutoFill
Options
uk train booking
AggregatorData:
GLUECE:
Cluster:
Name: Viz
UniqueID: Viz
SubCluster:
Name: main
UniqueID: main
Host:
Name: viz-2-int
UniqueID: viz-2-int
Processor:
CacheL1: 0
CacheL1D: 0
CacheL1I: 0
CacheL2: 0
ClockSpeed: 2400
InstructionSet: x86
MainMemory:
RAMAvailable: 1390
RAMSize: 2026
VirtualAvailable: 2408
VirtualSize: 3063
OperatingSystem:
Name: Linux
Release: 2.6.11
Architecture:
SMPSize: 2
FileSystem:
AvailableSpace: 371293
Name: entire-system
ReadOnly: false
Root: /
Size: 404708
NetworkAdapter:
IPAddress: 192.168.0.171
InboundIP: true
MTU: 1500
Name: viz-2-int
OutboundIP: true
Internet

- Last5Min: 0
- ComputingElement:
  - Name: normal
  - UniqueID: normal
  - Info:
    - TotalCPUs: 8
  - State:
    - EstimatedResponseTime: 0
    - FreeCPUs: 8
    - RunningJobs: 0
    - Status: enabled
    - TotalJobs: 0
    - WaitingJobs: 0
    - WorstResponseTime: 0
  - Policy:
    - MaxCPUTime: 0
    - MaxRunningJobs: 0
    - MaxTotalJobs: 0
    - MaxWallClockTime: 0
    - Priority: 0
- ComputingElement:
  - Name: low
  - UniqueID: low
  - Info:
    - TotalCPUs: 8
  - State:
    - EstimatedResponseTime: 0
    - FreeCPUs: 8
    - RunningJobs: 0
    - Status: enabled
    - TotalJobs: 0
    - WaitingJobs: 0
    - WorstResponseTime: 0
  - Policy:
    - MaxCPUTime: 0
    - MaxRunningJobs: 0
    - MaxTotalJobs: 0
    - MaxWallClockTime: 0
    - Priority: 0



ervice Group Entry Detail

rvice Group EPR

- Address: https://128.9.72.140:9000/wsrf/services/DefaultIndexServiceEntry
- GroupKey: 14133705
- EntryKey: 130947

---

ember Service EPR

- Address: https://128.9.72.140:9000/wsrf/services/ReliableFileTransferFactoryService

---

ntent

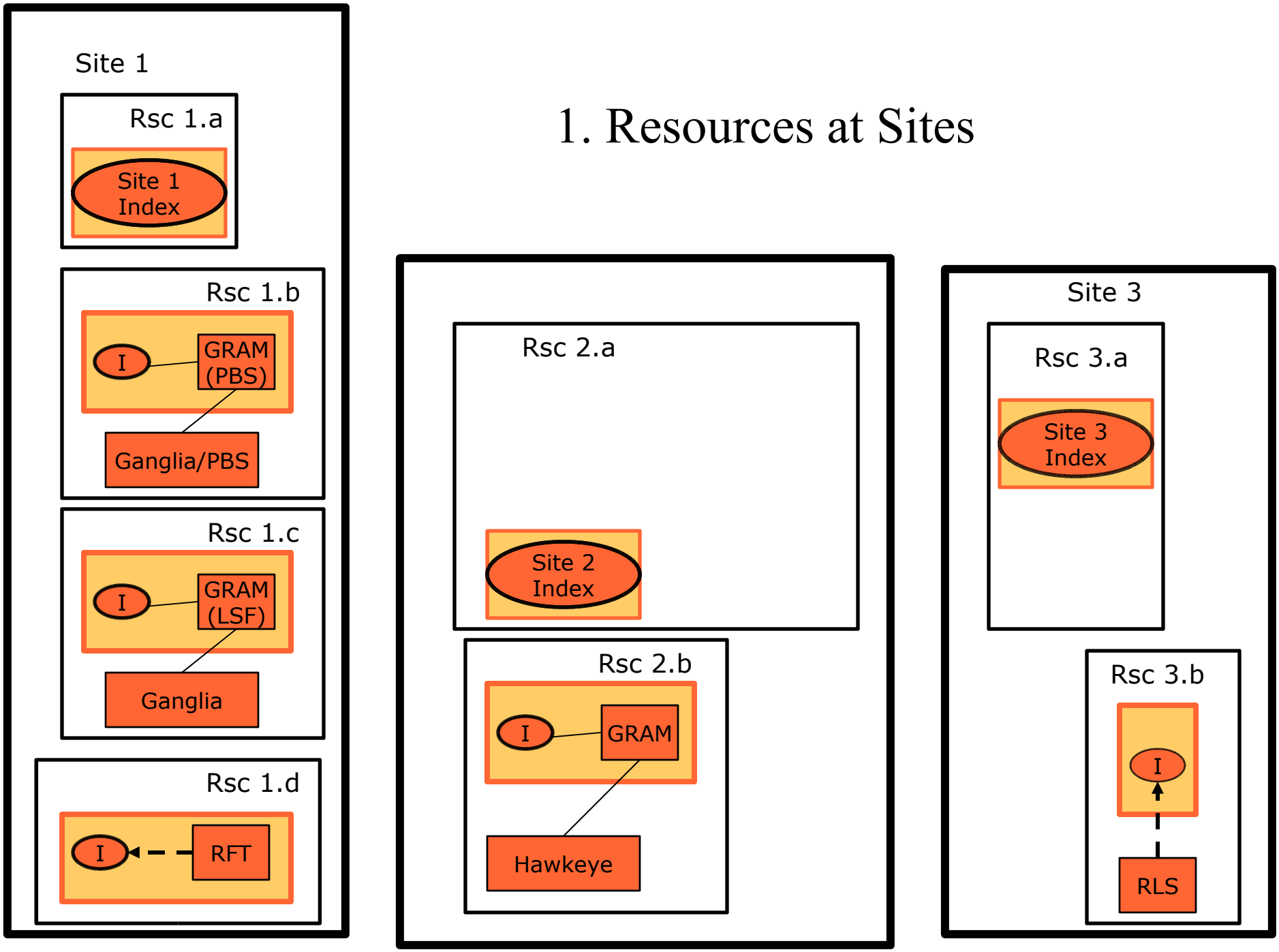
- AggregatorConfig:
  - GetMultipleResourcePropertiesPollType:
    - PollIntervalMillis: 60000
    - ResourcePropertyNames: rft:TotalNumberOfBytesTransferred
    - ResourcePropertyNames: rft:TotalNumberOfActiveTransfers
    - ResourcePropertyNames: rft:RFTFactoryStartTime
    - ResourcePropertyNames: rft:ActiveResourceInstances
    - ResourcePropertyNames: rft:TotalNumberOfTransfers
- AggregatorData:
  - TotalNumberOfBytesTransferred: 13478029392
  - TotalNumberOfActiveTransfers: 0
  - RFTFactoryStartTime: 2005-04-27T07:00:20.179Z
  - ActiveResourceInstances: 0
  - TotalNumberOfTransfers: 151231

---

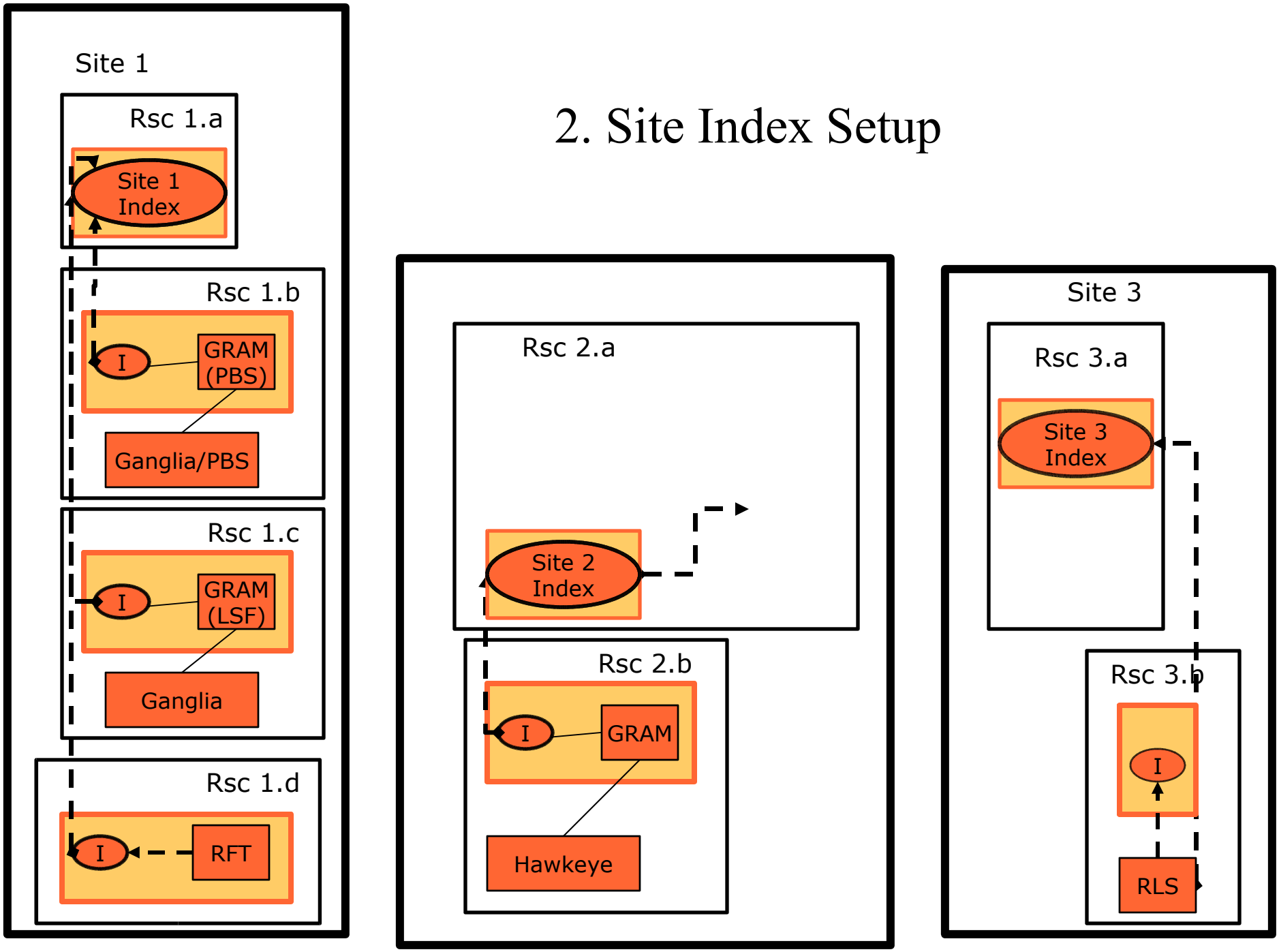
ease report bugs and feature requests into the Globus Bugzilla.

Any questions before I walk through a sample deployment?

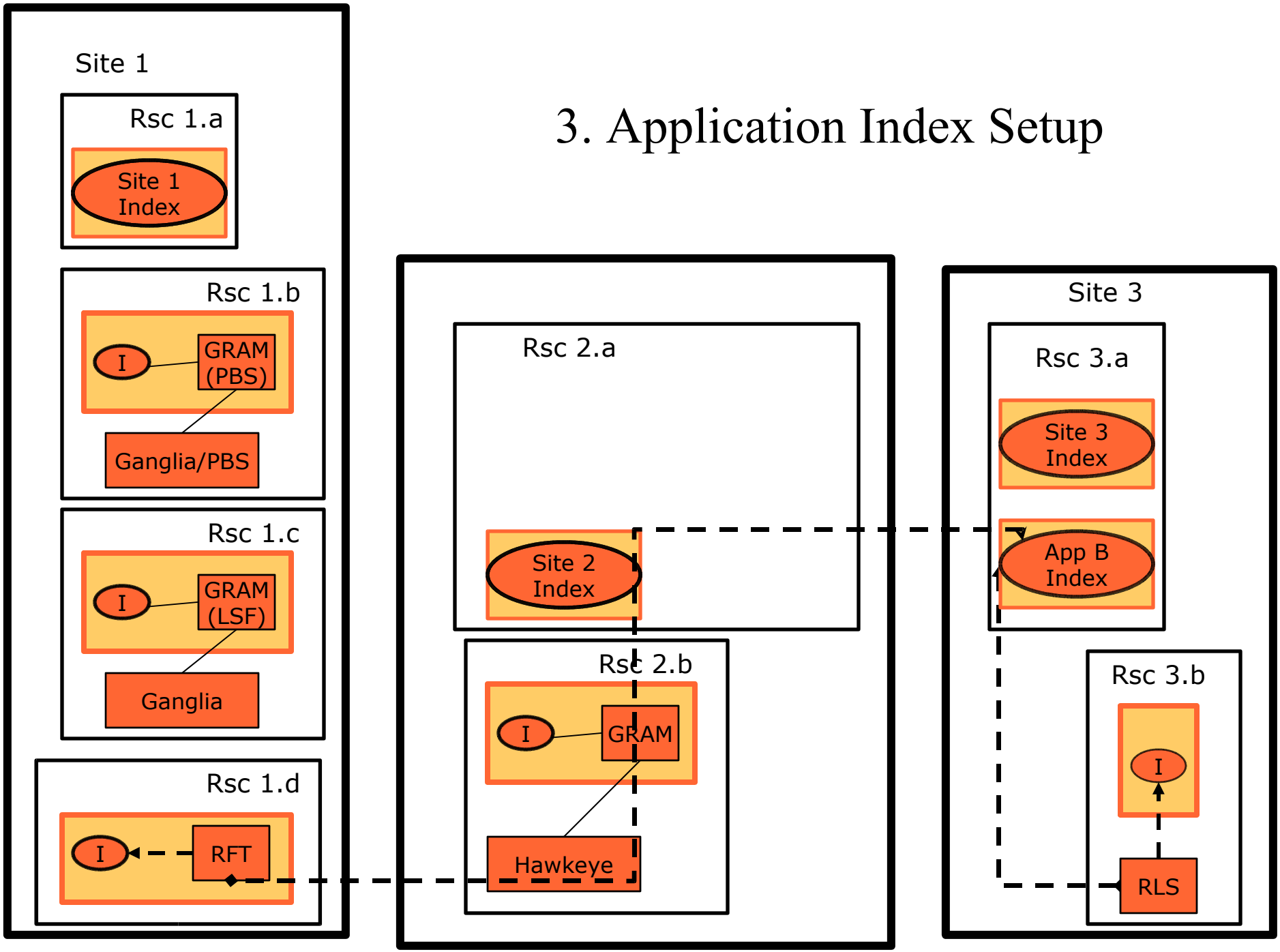
1. Resources at Sites



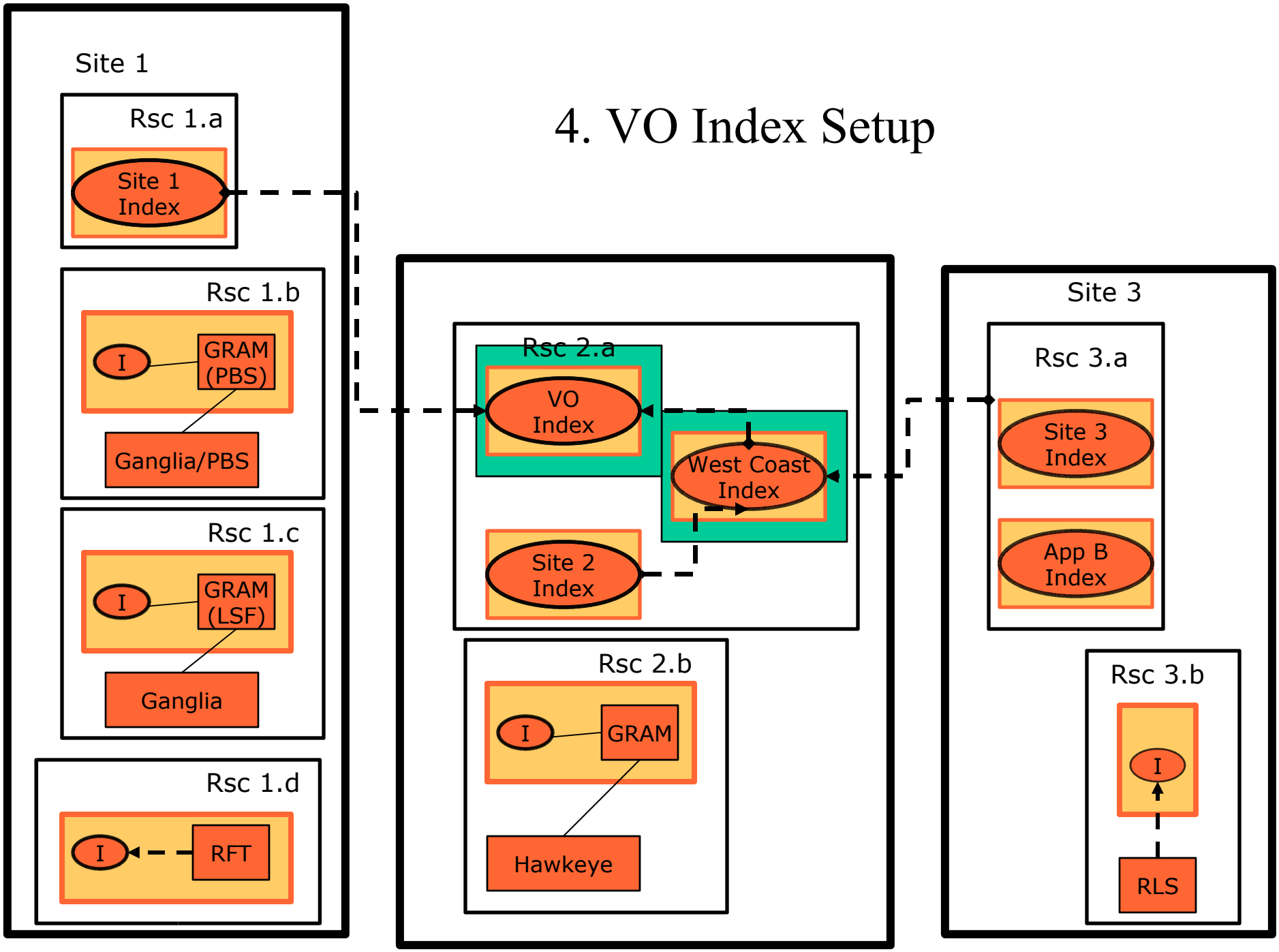
2. Site Index Setup

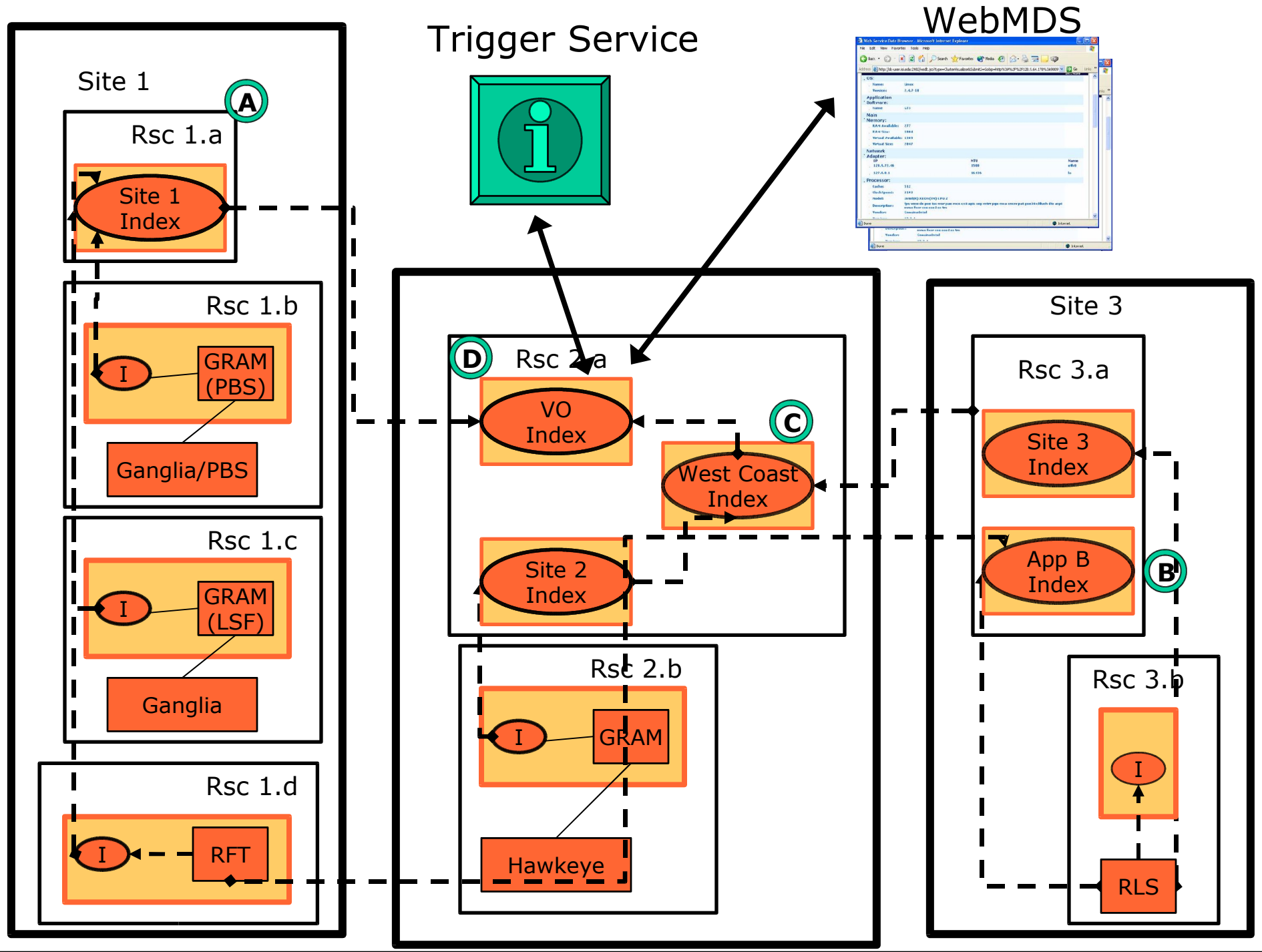


3. Application Index Setup



4. VO Index Setup




Trigger Service
WebMDS
Site 1
A
Rsc 1.a
Site 1 Index
Rsc 1.b
I
GRAM (PBS)
Ganglia/PBS
Rsc 1.c
I
GRAM (LSF)
Ganglia
Rsc 1.d
I
RFT
D
Rsc 2.a
VO Index
C
West Coast Index
Site 2 Index
Rsc 2.b
I
GRAM
Hawkeye
Site 3
Rsc 3.a
Site 3 Index
App B Index
B
Rsc 3.b
I
RLS

With this deployment, the project can…

- Discover needed data from services in order to make job submission or replica selection decisions by querying the VO-wide Index
- Evaluate the status of Grid services by looking at the VO-wide WebMDS setup
- Be notified when disks are full or other error conditions happen by being on the list of administrators
- Individual projects can examine the state of the resources and services of interest to them

Some Performance Numbers

- Basic Index Server Performance
  - How long does one response take?
  - How many responses per minute are possible?
  - How long does the service stay up while being used before failing?
- The set up
  - 5 client nodes (ned0–ned4), dedicated
    - > dual CPU 1133MHz Pentium III machines with 1.5GB of RAM
  - 1 server node (dc-user2), shared
    - > dual Intel (hyperthreaded) Xeon, 2.20GHz with 1GB of RAM
  - Interconnected by Gigabit Ethernet, same physical switch

Index Server Performance (3.9.4)

Index Size	1 client		2 Clients		25 Clients		100 Clients	
	Sing. clt q/sec	Resp. Time (msec)	Sing. clt q/sec	Resp. Time (msec)	Sing. clt q/sec	Resp. Time (msec)	Sing. clt q/sec	Resp. Time (msec)
10	24	40	22	44	4.5	245	0.85	1243
30	15	64	10	93	n/a	n/a	n/a	n/a
100	5	190	4	265	0.78	1334	0.19	5824

Index Server Performance

- As the MDS4 Index grows, query rate and response time both slow, although sublinearly
- Response time slows due to increasing data transfer size
  - Full Index is being returned
  - Response is re-built for every query
- Real question – how much over simple WS-N performance?

MDS4 compared to other systems

Monitoring System	1 client		10 Clients		50 Clients		100 Clients	
	Sing. cli. q/sec	Resp. Time (msec)	Sing. cli. q/sec	Resp. Time (msec)	Sing. cli. q/sec	Resp. Time (msec)	Sing. cli. q/sec	Resp. Time (msec)
MDS2 w/cache	0.88	129	0.45	147	0.92	153	0.93	182
MDS2 w/o cache	0.45	1219	0.15	5534	0.77	29,175	0.91	40,410
R-GMA	0.92	61	0.03	277	0.24	3230	0.89	9734
Hawkeye	0.93	79	0.02	106	0.12	113	0.68	463
MDS-4	24	40	*16.8*	n/a	*3.29*	n/a	0.85	1243

Index Server Stability

- Zero-entry index on same server
- Ran queries against it for 1,225,221 seconds (just over 2 weeks)
  - (server machine was accidentally rebooted)
- Processed 93,890,248 requests
  - Avg 76 per second
  - Average query round-trip time of 13ms
- No noticeable performance or usability degradation over the entire duration of the test

Summary

- MDS4 is a WS-based Grid monitoring system that uses current standards for interfaces and mechanisms
- Available as part of the GT4 release
  - Final April 29!
- Initial performance results aren't awful – we need to do more work to determine bottlenecks

Where do we go next?

- Extend MDS4 information providers
  - More data from GT4 WS
    - > GRAM, RFT, CAS
  - More data from GT4 non-WS components
    - > RLS, GridFTP
  - Interface to other data sources
    - > Inca, GRASP
  - Interface to archivers
    - > PinGER, NetLogger
- Additional scalability testing and development
- Additional clients

Other Possible Higher Level Services

- Archiving service
  - The next thing we're doing
  - Looking at Xindice as a possibility
- Site Validation Service
- Prediction service (ala NWS)
- What else do you think we need?

More Deployment Experience

- Open Science Gird (OSG) currently deploys 8-12 different monitoring tools
- Can MDS4 act as a common framework?

Summary

- Current monitoring systems
  - Insights into performance issues
- What we really want for monitoring and discovery is a combination of all the current systems
- Next-generation information architecture
  - WS-RF
  - MDS4 plans
- Additional work needed!

Thanks

- MDS4 Team: Mike D'Arcy (ISI), Laura Pearlman (ISI), Neill Miller (UC), Jennifer Schopf (ANL)
- Students: Xuehai Zhang (UC), Jeffrey Freschel (UW)
- Testbed/Experiment support and comments
  - John Mcgee, ISI; James Magowan, IBM-UK; Alain Roy and Nick LeRoy at University of Wisconsin, Madison;Scott Gose and Charles Bacon, ANL; Steve Fisher, RAL; Brian Tierney and Dan Gunter, LBNL.
- This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under contract W-31-109-Eng-38, and NSF NMI Award SCI-0438372. This work also supported by DOESG SciDAC Grant, iVDGL from NSF, and others.

For More Information

- Jennifer Schopf
  - jms@mcs.anl.gov
  - http://www.mcs.anl.gov/~jms
- Globus Toolkit MDS4
  - http://www.globus.org/mds
- Scalability comparison of MDS2, Hawkeye, R-GMA
  - www.mcs.anl.gov/~jms/Pubs/xuehaijeff-hpdc2003.pdf
  - Journal paper in the works – email if you want a draft
- Monitoring and Discovery in a Web Services Framework: Functionality and Performance of the Globus Toolkit's MDS4
  - Submitted to SC '05, online later this week