iGrid: a Relational Information Service

A novel resource & service discovery approach

Italo Epicoco, Ph.D.
University of Lecce, Italy

Italo.epicoco@unile.it

Outline of the talk

- Requirements & features
- Technologies
- Architecture
- Security
- Decentralized control & fault tolerance
- Information providers
- Relational schema
- Web service methods
- Performances
- iGrid Public Release

iGrid

GridLab Project

- The GridLab Project was born on the basis of two important
  - Co-development of grid infrastructure and grid-aware applications
  - Dynamic grid computing

The main goal of the GridLab project is to develop innovative grid computing technologies, which can be immediately and easily adopted and exploited by applications from many different research and engineering fields.

Requirements & features

iGrid Requirements

- Performance
- Scalability
- Security
- Uniformity
- Expressiveness
- Extensibility
- Dynamic data
- Flexible access
- Deployability
- Decentralized control

iGrid Features

- Web Service interface built using the gSOAP Toolkit
- Support for Globus Toolkit GSI security mechanism
  - Through our GSI plug-in for gSOAP Toolkit
- Distributed architecture
- Fault tolerance
- Based on the PostgreSQL relational DBMS
- Support for TLS binding to DBMS
- Support for heterogeneous DBMS
  - Through our GRelC library
- Easy to extend with new information providers
- Support for GAS authorization service (WP6)
- Support for MERCURY logging service (WP11)
- Platforms: linux, Mac OS X, tru64, irix
- Extreme Performance

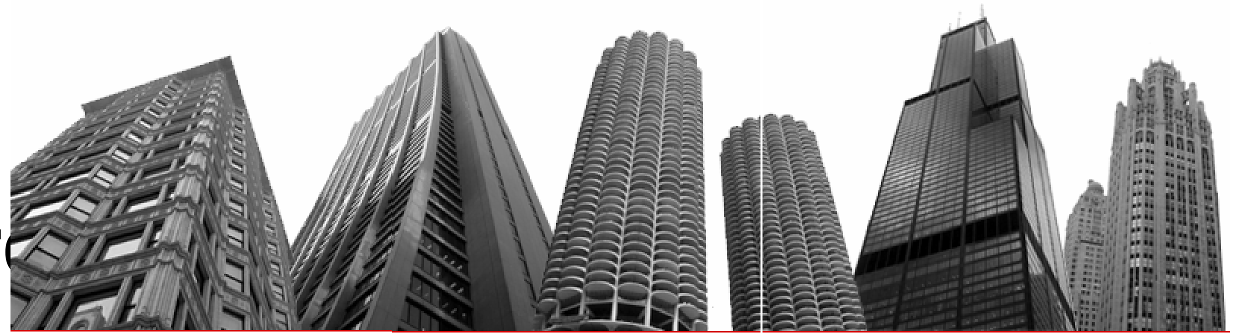
Technologies

iGrid Technologies

- gSOAP Toolkit
- Globus Toolkit GSI
- GSI plug-in for gSOAP
- PostgreSQL
- GRelC
- TLS
- libxml2

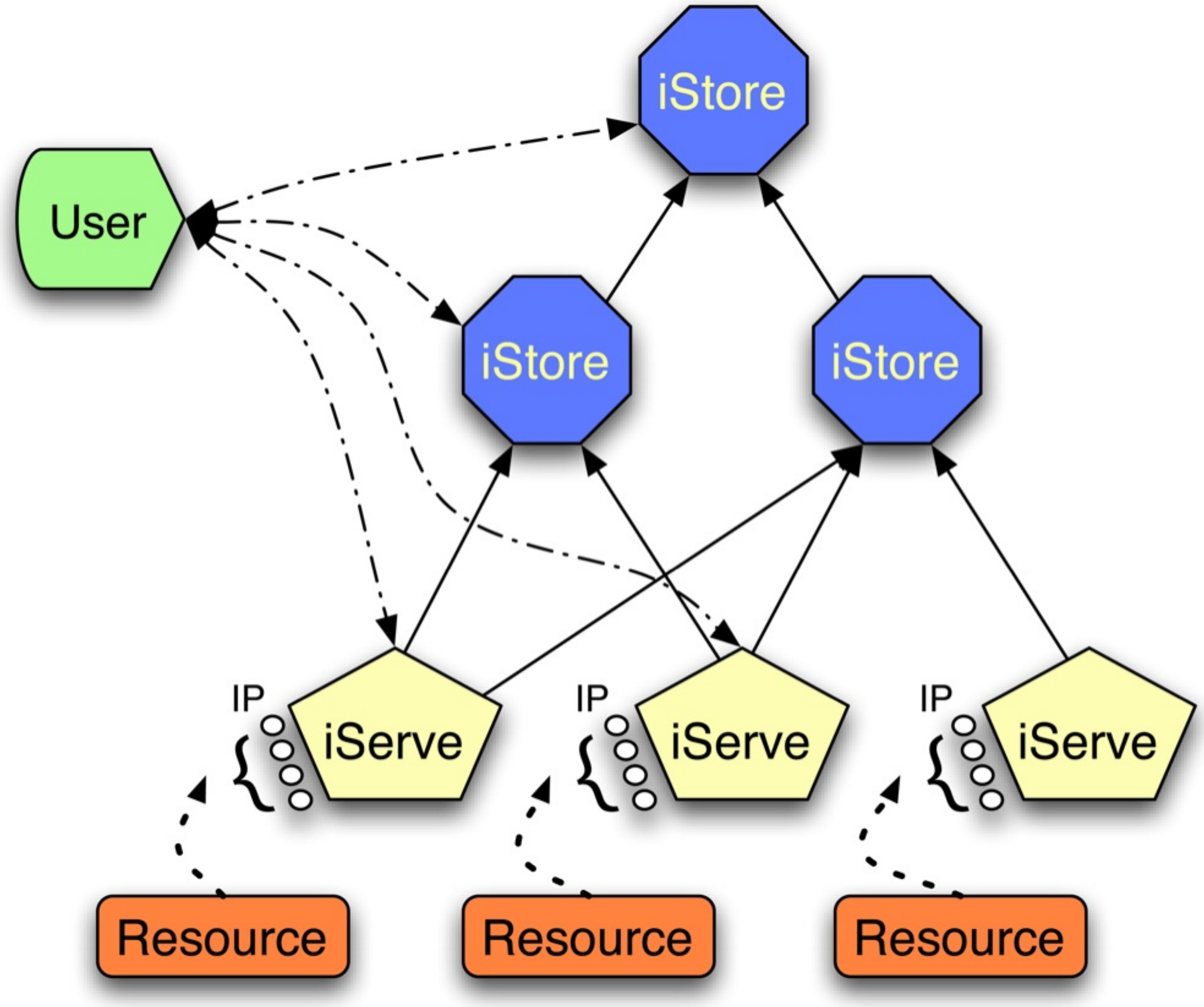
Architecture

iGrid Architectur



- Hierarchical, distributed architecture based on a server that can act as
  - iServe: collects information related to a specific computing resource
  - iStore: gathers the information related to registered iServes
- iGrid implements a push policy for data exchange
  - Information is extracted from resources synchronously and stored in a relational DBMS
  - Each user and /or service supplied information (asynchronously) is immediately stored on local DBMS and sent to iStores
- Information is always available and updated
  - On client request information is immediately retrieved from DBMS
  - No need for iStore to pull information from iServes

iGrid Architecture



iServe

- It extracts system information from local resource
- It stores user and/or service supplied information in the local DBMS
- Periodically, it collects and sends new information to remote iStores
- In case of iStores failure, iServe temporarily removes them from its iStore list
- Periodically, the iStore list is updated by adding previously removed iStore servers. The local database is dumped and sent to newly added iStores

iStore

- Acts as an iServe for its local resource
- It gathers information coming from registered and authorized iServes
- An iStore can be registered to other iStores, thus creating hierarchies

iGrid DBMS Maintenance

- Each kind of information is tagged with
  - creation date: it represents the UTC (GMT) time when the information has been acquired
  - time to live: it indicates how long information can be considered reliable
- iGrid removes expired information from DBMS
  - on each lookup, a data cleanup is performed. Stale information is never returned to clients
  - at startup the entire DBMS is cleaned up, removing stale information

Information dissemination

- The frequency of system information forwarding is based on the information itself, but we also allow defining a per information specific policy
- Currently, system information forwarding is based on the rate of change of the information itself. As an example, information that does not change frequently or change slowly (e.g. the amount of RAM installed) does not require a narrow update interval. Interestingly, this is true even for the opposite extreme, i.e., for information changing rapidly (e.g., CPU load), since it is extremely unlikely that continuous forwarding of this kind of information can be valuable for users, due to information becoming quickly inaccurate
- Finally, information whose rate of change is moderate is forwarded using narrow update intervals

Security

iGrid Security (I)

- Access to iGrid through a GSI enabled web service
  - using our GSI plug-in for gSOAP which provides:
    - code based on GSS APIs
    - mutual authentication
    - authorization
    - delegation of credentials
    - connection caching
    - support for both IPv4 and IPv6
    - support for development of both clients & servers
    - debugging framework
    - extensive error reporting

iGrid Security (II)

- GSI protects connections
  - from users/services to iServes
  - from iServes to iStores
  - from iStores to hierarchical iStores
- TLS protects connections
  - from an iServe or iStore to its PostgreSQL back-end
- This provides mutual authentication, confidentiality and anti-tampering
- And prevents snooping, spoofing, tampering, hijacking and capture replay network attacks

iGrid Security (III)

- Authorization can take advantage of
  - support for administrator defined ACL
  - support for GridLab GAS
  - fully customizable authorization mechanism through a GSI callback
- The iGrid daemon does not need to run as a privileged user
- We sanitize the environment on iGrid startup
- All user input is validated
- Special attention is devoted to the prevention of SQL injection attacks
  - an SQL SELECT query statement is escaped as needed
  - the query string is validated by an SQL parser which accepts only valid SELECT statements and rejects everything else

Decentralized control & fault tolerance

iGrid Decentralized Control

- Decentralized control in iGrid is achieved by means of distributed iServe and iStore services
- Both creation and maintenance of system/user/service information are delegated to the sites where resources are actually located, and administrators fully control how their site local policies are enforced
  - For instance, they decide
    - about users/services that must be trusted for both data lookup and registration
    - if an iServe must register to one or more iStores
    - if an iStore must accept incoming data from remote iServes
    - the frequency of system information dissemination from iServes to iStores on a per information provider basis
    - etc

iGrid Fault Tolerance

- In case of failure of an iStore, iServes temporarily remove the faulty iStore from their registration list. Periodically, the iStore list is updated by adding previously removed iStores when iStores are available again. In this case, the local database is dumped and immediately sent to newly added iStores
- Moreover, because of iGrid distributed and hierarchical architecture it is possible to implement a primary/backup approach by mirroring an iStore configuring all of the iServes to publish their information simultaneously on two or more different iStores

Information providers

iGrid Information Providers (I)

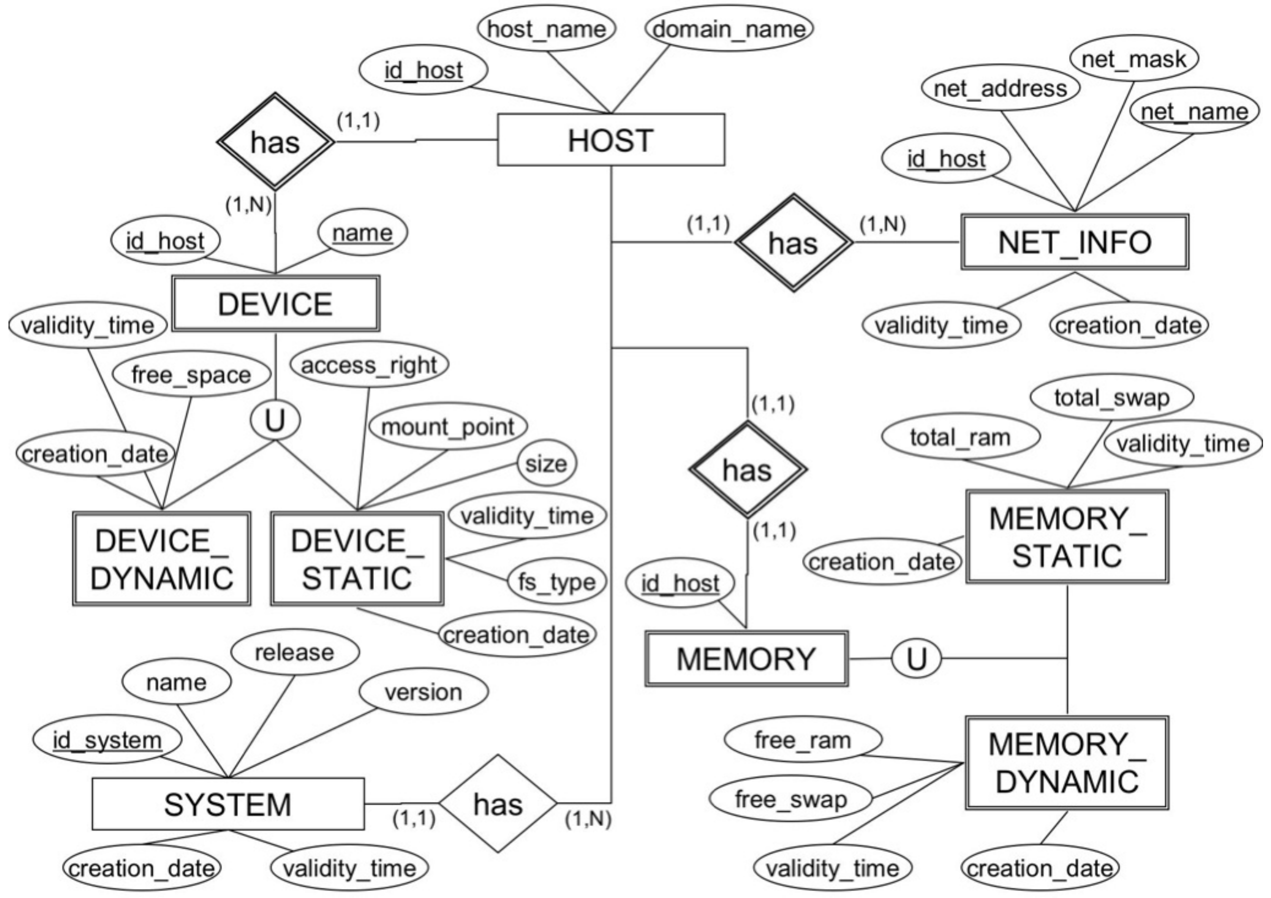
- System information
  - Cpu
  - Memory
  - Network
  - Filesystems
  - Users
  - Certification Authorities
  - Resource Managers

iGrid Information Providers (II)

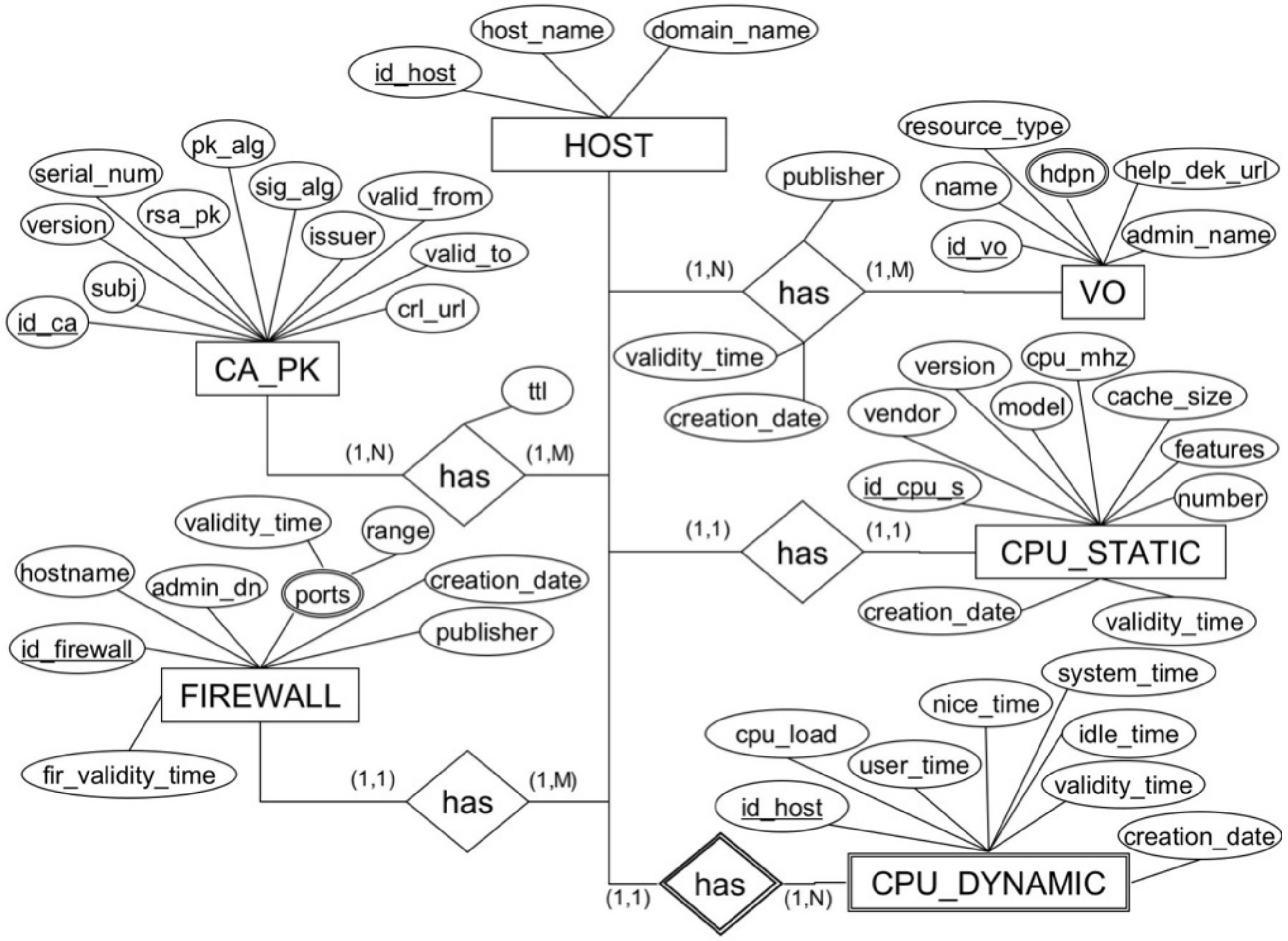
- User/Service supplied information
  - Services
  - Web services
  - Software
  - Firewalls
  - Virtual Organizations

Relational schema

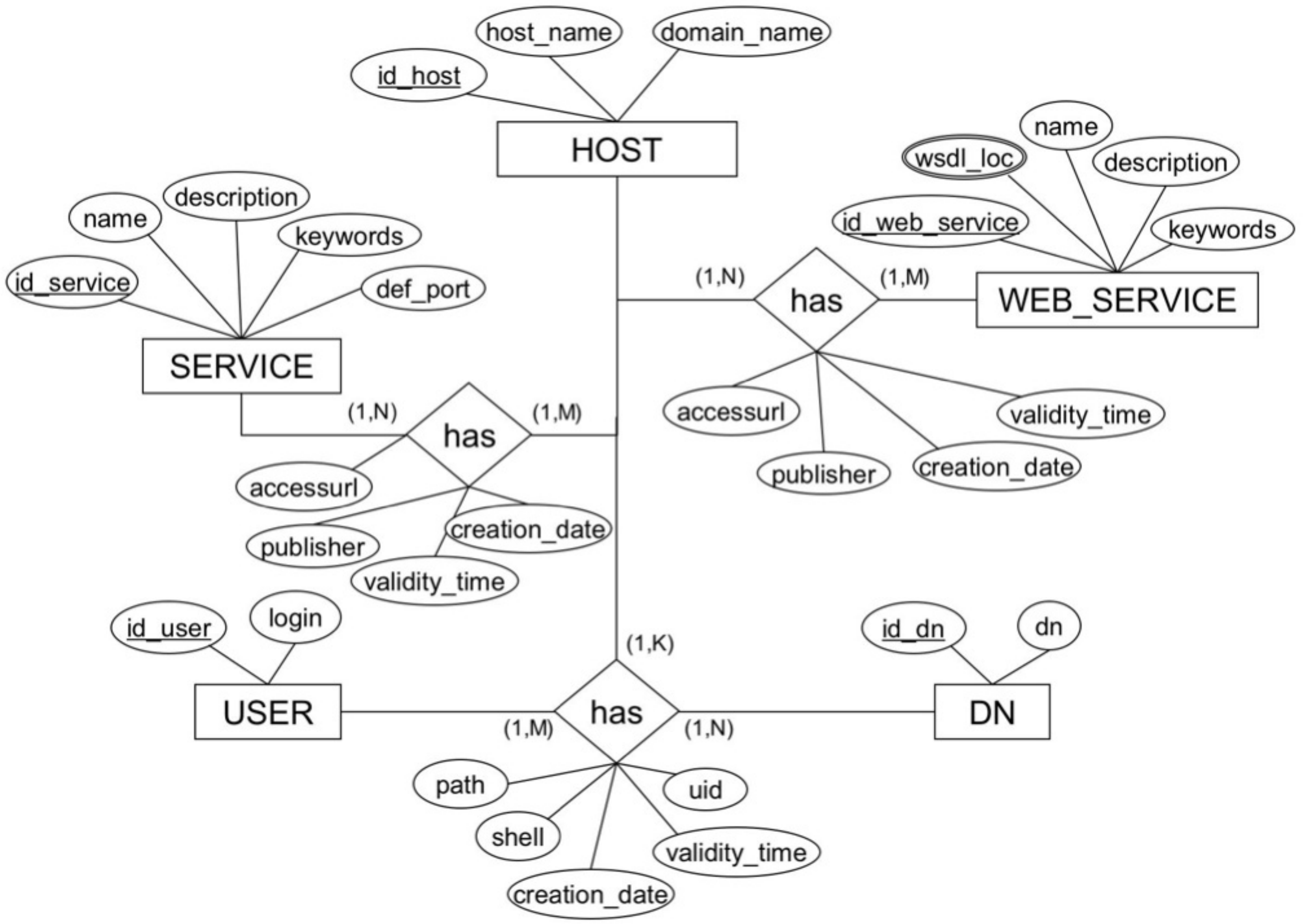
iGrid Relational schema (I)



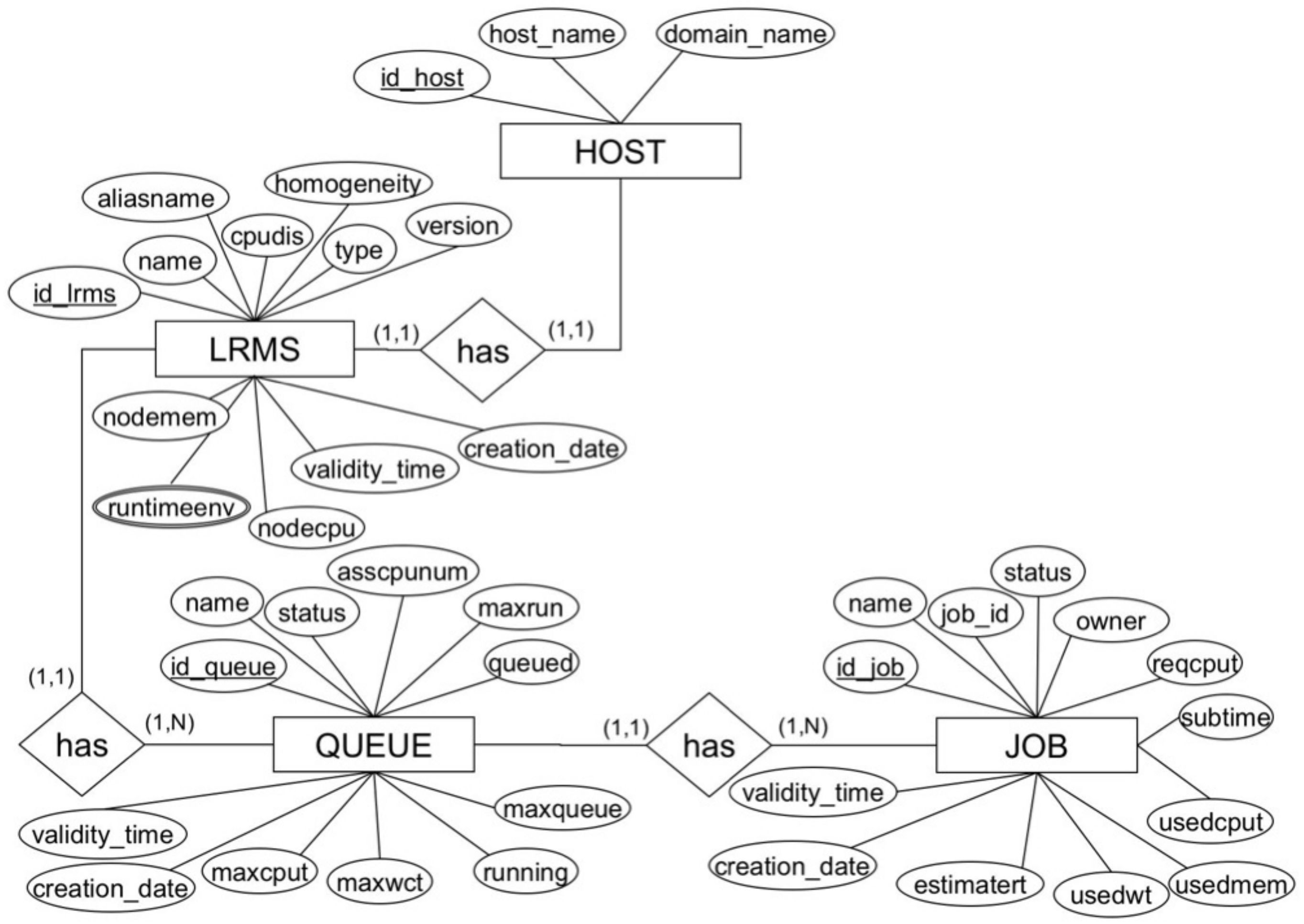
iGrid Relational schema (II)



iGrid Relational schema (III)



iGrid Relational schema (IV)



Web service methods

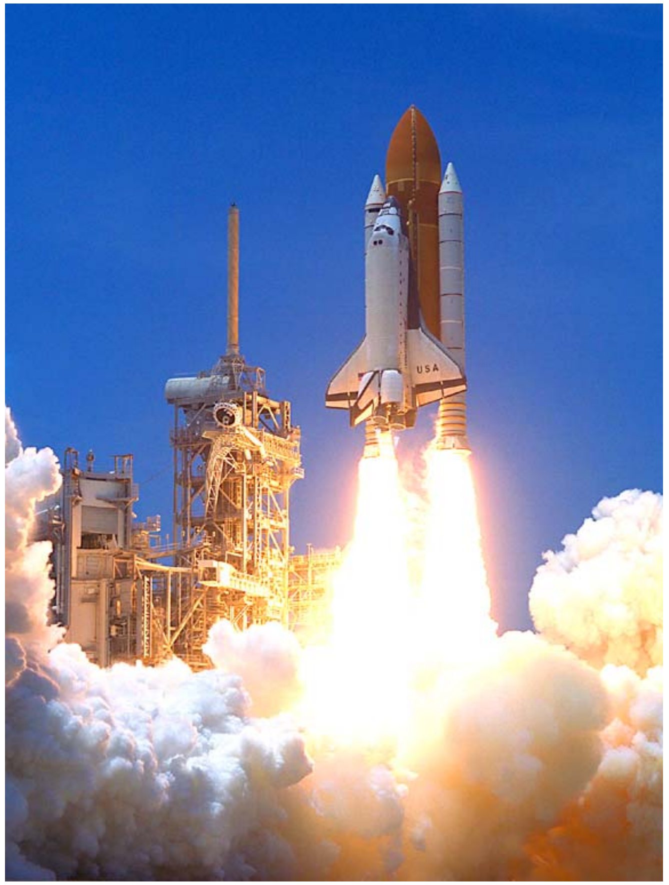
iGrid Web Service Methods

- SQL SELECT query
  - exact
  - range
  - approximate
  - etc
- Lookup
  - predefined set of search queries
- Register
- Unregister
- Update

Performances

iGrid Performances

- The performances of iGrid are extremely good. In what follows, we report the performance obtained for information retrieval on two testbeds
- There were no high speed, dedicated networks connecting the machines belonging to the testbeds, and Globus MDS2 servers were accessed without GSI authentication, while iGrid servers were accessed using GSI, and the PostgreSQL back-end using TLS
- The results were averaged over 50 runs



First Testbed

- Results, in seconds, are related to a testbed which comprises an iServe and a GRIS server hosted on one machine in Lecce, Italy, and an iStore and GIIS server hosted on another machine in Brno, Czech Republic. A query to both iStore and MDS2 GIIS was issued requesting all of the available information using clients in Lecce, Italy

Search for information related to all of the resources	Cache expired	Cache not expired
Globus MDS2 GIIS	77	7
iStore	0.6	

Second Testbed

- Results, in seconds, are related to iStore/GIIS servers running on a testbed which comprises four machines in Italy (three in Lecce and one in Bari, a city 200 Km far from Lecce) and one located in Poznan (Poland). The clients were in Poland, and the servers in Lecce. A total of four queries were issued
- The first query in this set requested all of the information of each registered resource
- The second one requested all of the information related to the machine in Poland
- The third one requested only the CPU information related to each registered resource
- The last query was a complex query involving many filtering operations to retrieve information about attributes related to CPU, memory, network and filesystems.

Second Testbed

iStore / GIIS	information related to all resources	information related to resource in Poland	information related to CPU of all resources	complex query
GIIS cache expired	25,88	23,52	22,42	18,59
GISS cache not expired	2,53	1,64	0,46	0,67
iStore	3,06	1,01	0,92	0,67

Second testbed

iServe / GRIS	All of the available information	CPU information
GRIS cache expired	23,88	3,36
GRIS cache not expired	1,05	0,39
iServe	0,95	0,81

iGrid Public Release

Work in progress

- We are now finalizing version 2.0 Public Release
- We are investigating a peer-to-peer approach
- Additional information dissemination protocols
- New information providers
- Additional improvements

Q & A