"Coordinated Use of Globus Pre-WS and WS Resource Management Services with GridWay"

Eduardo Huedo (huedoce@inta.es)

Rubén S. Montero

Ignacio M. Llorente

Advanced Computing Laboratory

Distributed Systems Architecture group

Center for Astrobiology (INTA-CSIC)

Complutense University of Madrid

2nd GADA Workshop / OTM 2005 Conferences

Objectives

- Present the modular architecture of **Grid*Way*** which allows the coordinated use of different Grid infrastructures, although based on different middleware and service technologies.
- The proposed architecture eases:
  - the **gradual migration** from pre-WS Grid services to WS ones, and
  - the **long-term coexistence** of both.
- Its suitability will be demonstrated with the evaluation of the **coordinated use** of two Grid infrastructures:
  - a **research testbed** based on Globus WS Grid services, and
  - a **production testbed** based on Globus pre-WS Grid services, as part of the LCG middleware.

From Pre-WS to WS Grid Services

- Pre-WS Grid services are based on **proprietary interfaces**, although usually implemented over **standard protocols** (HTTP, LDAP, FTP...).
- WS Grid services are based on the **WS-Resource Framework** (WSRF), which is a set of conventions and usage patterns within the context of established **WS standards** (WS-Addressing, WS-Notification...).
- WSRF defines the **WS-Resource** construct as a composition of a **Web Service** and a **stateful resource**.
- WSRF makes easier/possible to **define** (and so standardize) a service architecture, like **OGSA**, since only **service interfaces** and **resource properties**, representing the resource state, have to be specified.

Globus Approach for Resource Management

- Pre-WS GRAM vs. WS GRAM
  - Better overall performance associated with pre-WS GRAM.
  - Better job status monitoring mechanism in WS GRAM:
    - use of a **Job State Monitor** (JSM) and **Scheduler Event Generator** (SEG) instead of a polling mechanism in the Job Manager.
  - More scalable/reliable file handling in WS GRAM:
    - use of the **Reliable File Transfer** (RFT) service with **GridFTP**, and
    - removal of **GASS** transfer and caching.
- WSRF-based Grid services in GT4 clearly outperforms heavy-weight OGSI-based Grid services in GT3.

GridWay Approach for Job Management

www.gridway.org

Easier and efficient execution in dynamic and heterogeneous grids in a ***submit & forget*** fashion.

Functionality:

- **Adaptive scheduling**
- **Adaptive execution**
- **High throughput apps.**
- **DRMAA standard**

Design Guidelines:

- **Adaptable/extensible** (modular design)
- **Scalable** (decentralized architecture)
- **Deployable** (user, standard services)
- **Applicable** (wide application range)

Grid*Way* Approach for Job Management

- Components of Grid*Way*:
  - **Request Manager** (RM): To handle client requests.
  - **Dispatch Manager** (DM): To perform job scheduling.
  - **Submission Manager** (SM): To perform the stages of job execution, including job migration.
  - **Execution Manager** (EM): To execute each job stage.
  - **Performance Monitor** (PM): To evaluate job performance.
- The framework has been designed to be **modular** to allow adaptability, extensibility and improvement of its capabilities.

GridWay Approach for Job Management

- The **Submission Manager** is responsible for the execution of the job during its lifetime, i.e. until it is done or stopped.
- It performs the following tasks:
  - **Preparation**: Submission of **Prolog** executable.
  - **Submission**: Submission of **Wrapper** executable and waiting for events.
  - **Cancellation**: Cancellation of the submitted job if a migration, stop or kill event is received.
  - **Finalization**: Submission of **Epilog** executable.
- It doesn't rely on the underlying middleware to perform preparation and finalization tasks.
- **Prolog** and **Epilog** are submitted to the cluster front-end node and **Wrapper** is submitted to a compute node. Therefore, no **middleware installation** nor **network connectivity** is required in the compute nodes (**"end-to-end" architecture**).

Grid*Way* Approach for Job Management

- The **Execution Manager** uses a **Middleware Access Driver** (MAD) module to **submit**, **monitor** and **control** the execution of **Prolog**, **Wrapper** and **Epilog** modules.
- The MAD module is an abstraction of the resource management middleware layer:
  - provides **basic operations**, like submitting, polling or cancelling jobs, and
  - receives **asynchronous notifications** about the state of each submitted job.
- Currently, there are two MADs available:
  - one, written in C, interfaces to **pre-WS GRAM** services, and
  - other, written in Java, interfaces to **WS GRAM** services.
- **Java Virtual Machine** (JVM) initialization time doesn't affect, since the JVM is initiated before the start of measurements.

Application (NGB ED)

- We have chosen the **Embarrassingly Distributed** (ED) benchmark from the **NAS Grid Benchmark** (NGB) suite.
- This benchmark represent the class of **Parameter Sweep Applications** (PSA), so important in the Grid.
- Problem sizes, in terms of mesh size, iterations and number of tasks, are defined as classes by NGB.
- We have used a **class A** problem but, instead of submitting 9 tasks (as NGB defines for class A) we submitted more tasks in order to have a real **high-throughput** application.

Research Testbed (UCM Grid)

- Based on Globus WS Grid services.

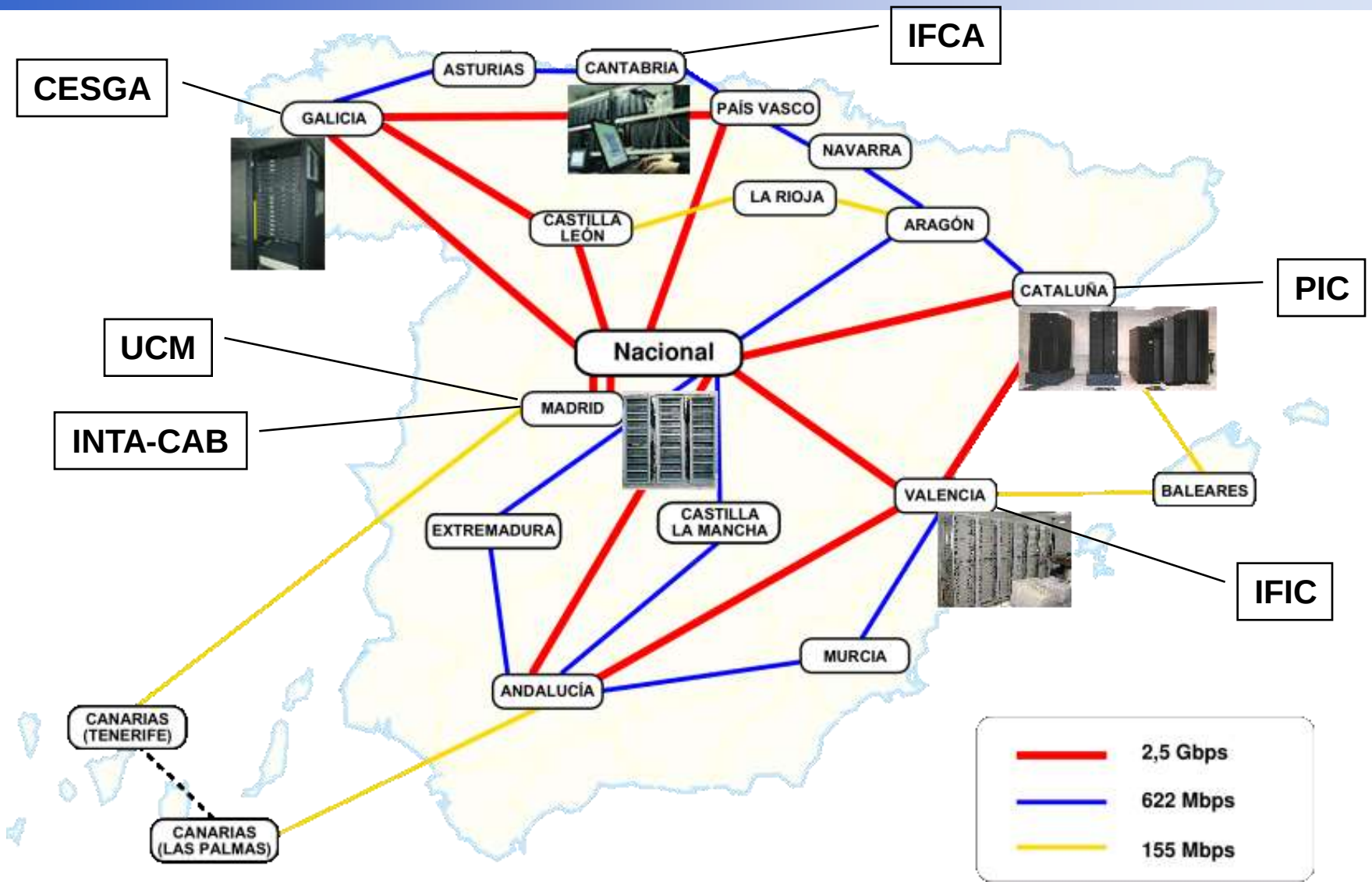
Name	Site	Location	Nodes	Processor	Speed	Memory per node	DRMS
cygnus	UCM	Madrid	1	Intel P4	2.5GHz	512MB	-
ursa	UCM	Madrid	1	Intel P4	3.2GHz	512MB	fork
draco	UCM	Madrid	1	Intel P4	3.2GHz	512MB	fork
hydrus	UCM	Madrid	4	Intel P4	3.2GHz	512MB	PBS
aquila	UCM	Madrid	2	Intel PIII	600MHz	250MB	SGE

Production Testbed (EGEE-ES)

- Based on Globus pre-WS Grid services, as part of the LCG middleware.
  - In a previous work, Grid*Way* was adapted to work over the LCG-2 middleware.

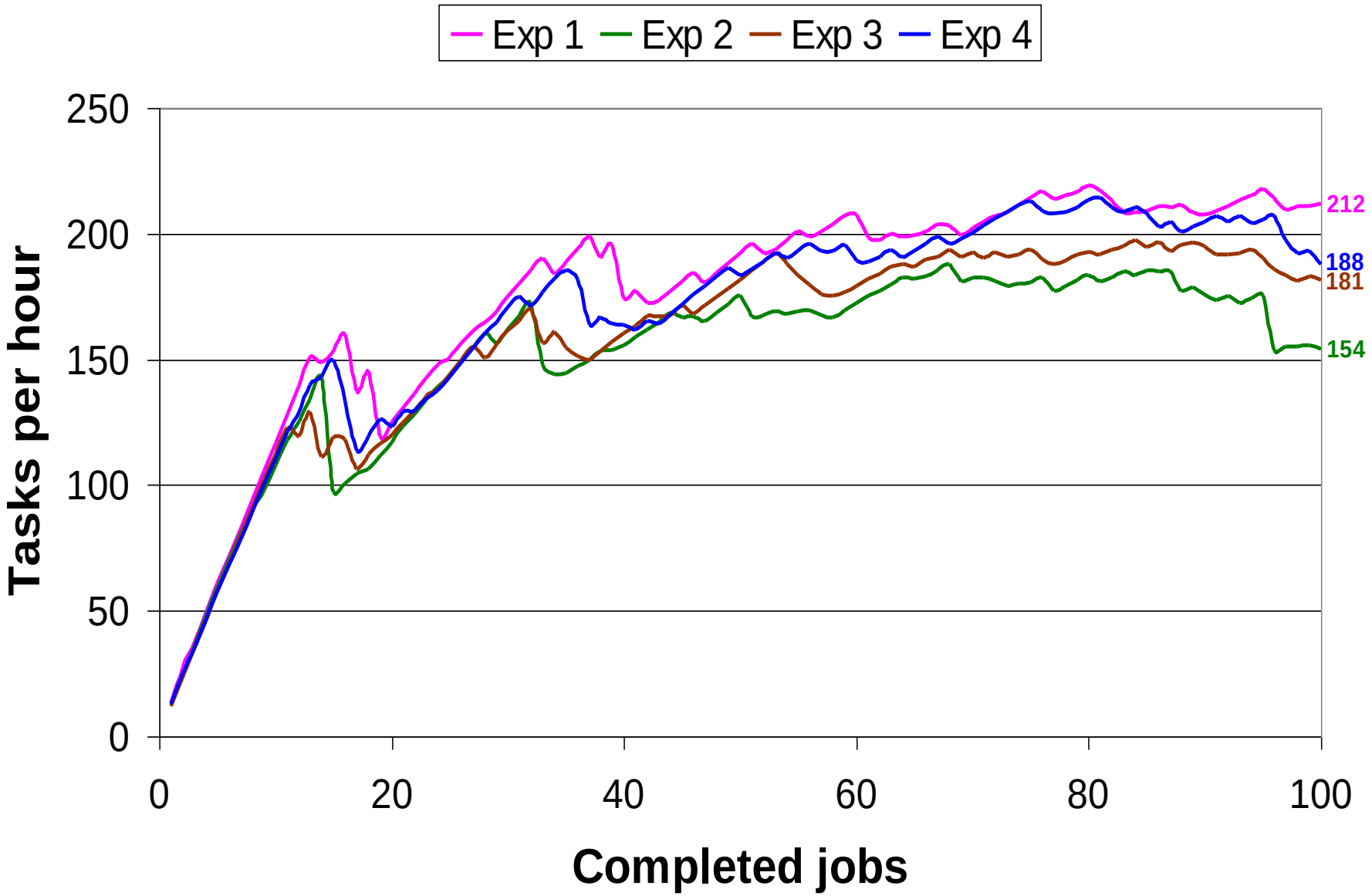
Name	Site	Location	Nodes	Processor	Speed	Memory per node	DRMS
egeece	IFCA	Cantabria	28	2×Intel PIII	1.2GHz	512MB	PBS
lcg2ce	IFIC	Valencia	117	AMD Athlon	1.2GHz	512MB	PBS
lcg-ce	CESGA	Galicia	72	Intel P4	2.5GHz	1GB	PBS
ce00	INTA-CAB	Madrid	4	Intel P4	2.8GHz	512MB	PBS
ce01	PIC	Cataluña	65	Intel P4	3.4GHz	512MB	PBS

Joint Testbed

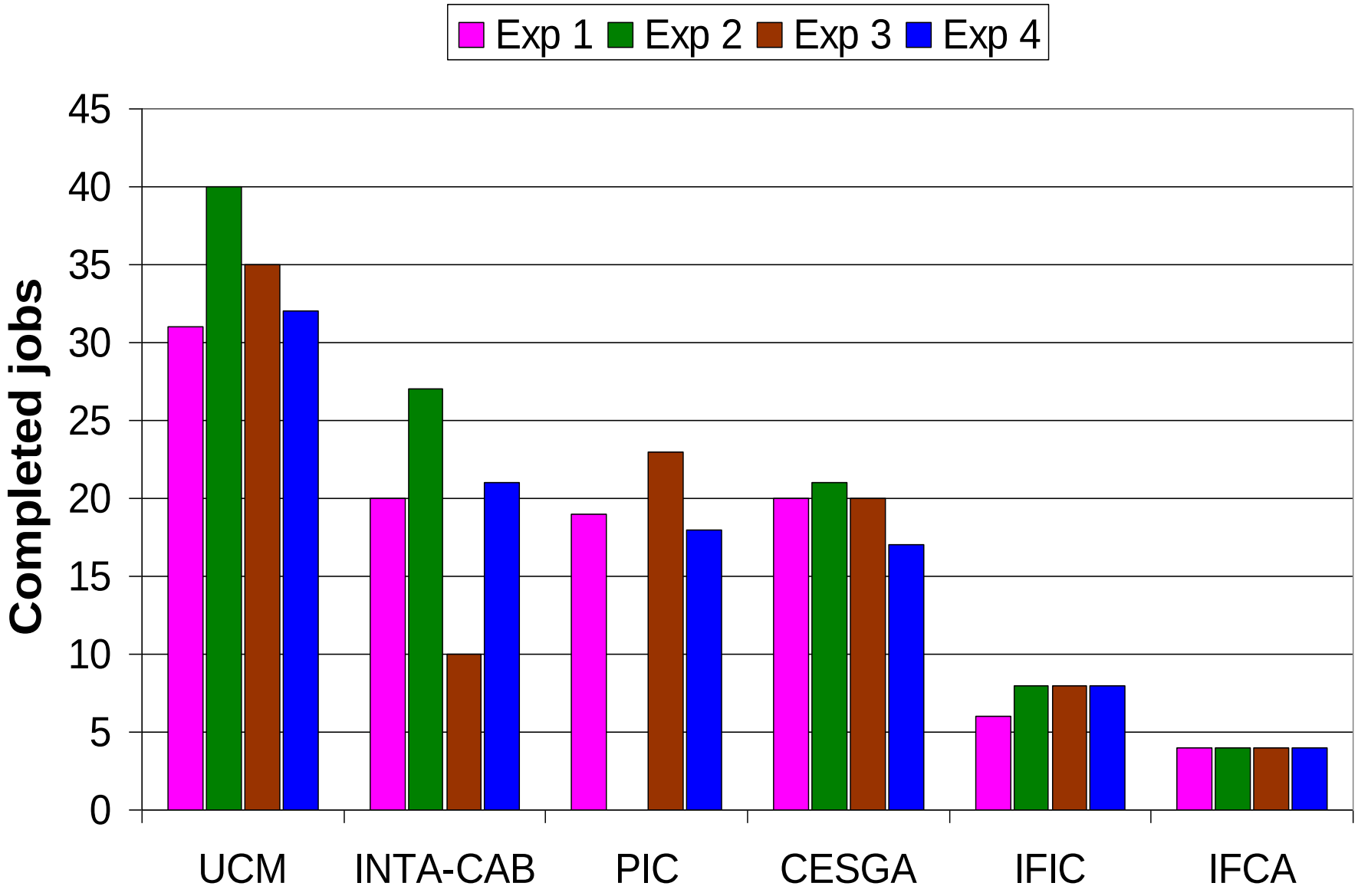


- Only four nodes simultaneously used on each resource.

Results: Dynamic Throughput over the Joint Testbed



Results: Scheduling over the Joint Testbed



Results: Comparison of Pre-WS and WS GRAM

Conclusions

- **Grid*Way***, as **user-level Grid middleware**, can work with Globus, as a standard **core Grid middleware**, over any **Grid fabric** in a *loosely-coupled* way.
- The **Grid*Way*** approach (the Grid way), based on a **modular, decentralized**, and **"end-to-end" architecture**, is appropriate for the Grid.
- Results resemble the advantages of *loosely-coupled* grids, since they allow:
  - a **straightforward resource sharing**, and
  - an **easier, scalable and compatible deployment**.