October 20, 2005
Grid Workshop

Technologies for Grid Computing

Ignacio Martín Llorente
asds.dacya.ucm.es/nacho


Grupo de Arquitectura de Sistemas Distribuidos
Departamento de Arquitectura de Computadores y Automática
Universidad Complutense de Madrid



Laboratorio de Computación Avanzada, Simulación y Aplicaciones Telemáticas
Centro de Astrobiología CSIC/INTA
Asociado al NASA Astrobiology Institute

Objectives of the Presentation

- **Introduce grid computing** as the technology to enable secure remote operation over multiple administration domains with different resource management systems and access policies
- Describe **the Globus Toolkit and the GridWay meta-scheduler** as the components required to deploy computing grids infrastructures

12:00 **Data Management Technologies for Grid using Globus**, Dr. Félix García, UC3M

Contents

Contents

Goal of Parallel and Distributed Computing Platforms

- **Efficient execution** of computational or data-intensive applications

Types of Computing Environments

High Performance Computing (HPC) Environments

- Reduce the execution time of a single distributed or shared memory parallel application
- Performance measured in floating point operations per second
- Sample areas: CFD, climate modeling…

High Throughput Computing (HTC) Environments

- Improve the number of executions per unit time
- Performance measured in number of jobs per second
- Sample areas: HEP, Bioinformatics, Monte-Carlo simulation, Financial models…

Types of Computing Platforms

Centralized Computing

SMP servers

MPP servers

Distributed Computing

Dedicated clusters

Non-dedicated clusters

High Performance Computing Servers

- Shared (SMP) or distributed memory (MPP) computing architectures

Application profile

- Efficient execution of both HPC and HTC applications

Advantages

- High bandwidth and low latency Interconnection network
- Uniform environment and single view of the system provided by the operating system

Disadvantages

- Low scalability limits (for SMPs)
- Complex programming models (for HPC on MPPs)
- High cost

Batch queuing system

NQE

Dedicated Clusters

- Clusters of homogeneous dedicated PCs or workstations interconnected by system area networks (Giganet, Myrinet…)

Application profile

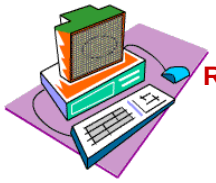
- Efficient execution of HTC and coarse-grain HPC applications

Advantages

- More cost effective for HTC applications
- Higher scalability limits

Disadvantages

- Require distributed memory programming model (message passing library such as MPI) for HPC applications



Resource management system

PBS

Non-Dedicated Clusters

- Clusters of heterogeneous non-dedicated PCs or workstations interconnected by local area networks (Fast ethernet…)

Application profile

- Only execution of HTC applications

Advantages

- Minimum cost for HTC applications
- Higher scalability limits

Disadvantages

- Low bandwidth and high latency interconnection network
- Require adaptation management capabilities to use idle times in the dynamic resources

Management of Computing Platforms

Computing platforms are managed by different types of **Distributed Resource Management (DRM) Systems**:

- Batch queuing systems for servers
- Resource management systems for dedicated clusters
- Workload management systems for non-dedicated clusters

DRM Systems Capabilities

DRM systems **share many capabilities**:

- Batch queuing
- Job scheduling
- Resource management

DRM Systems Benefits

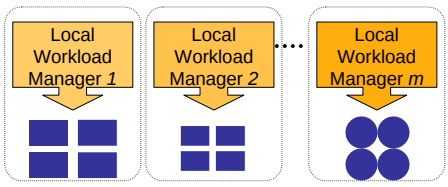
Their benefits in **cost minimization** and **performance maximization** are mainly due to greater utilization of underlying resources

DRM Systems

Independent Suppliers	Open Source	OEM Proprietary
Platform Computing **LSF**	*Altair* **Open PBS**	*IBM* **Load Leveler**
Altair **PBS Pro**	*University of Wisconsin* **Condor**	*Cray* **NQE**
	Sun Microsystems **SGE**	

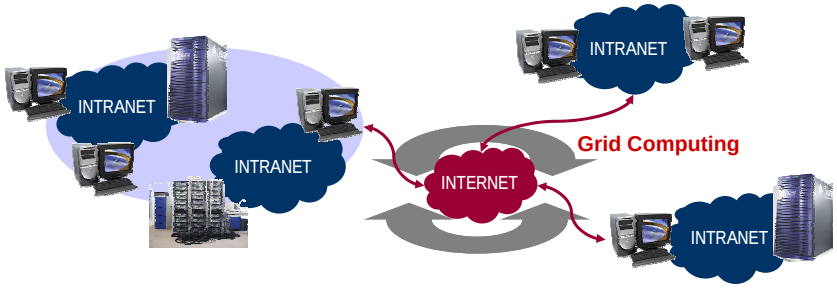
Non-interoperable Computing Vertical Silos within the Organization

- DRM systems do not provide a common interface and security framework, and so **their integration is not possible**
- Such lack of interoperability involves the existence within an organization of **independent computational platforms** (*vertical silos*) responsible for distinct functions that:
  - Require **specialized administration skills**
  - **Increase operational costs**
  - Generates **overprovisioning** and **global load unbalance**



Unsuitable to Build Multiple Organization Infrastructures

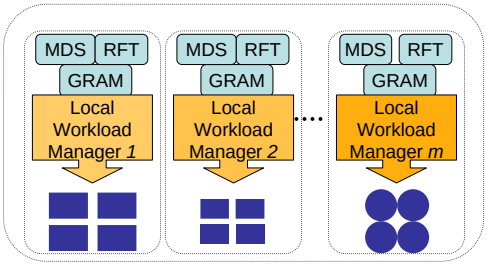
- Such technologies are also unsuitable to build computational infrastructures where resources are scattered across several administrative domains, **each with its own security policy and DRM system**.



Contents

Grid Infrastructure

- A grid infrastructure offers a **common layer to integrate these non-interoperable computational platforms** (*vertical silos*) by defining a consistent set of abstraction and interfaces for access to, and management of, shared resources
- The **Grid services** include, among others, resource monitoring and discovery, resource allocation & management, a security infrastructure, and file transfer



Grid Philosophy

A grid is a system that...

1) ...coordinates resources that are not subject to a centralized control...

2) ...using standard, open, general-purpose protocols and interfaces...

3) ...to deliver nontrivial qualities of services.

Ian Foster

What is the Grid? A Three Point Checklist (2002)

The Globus Toolkit, a de facto Standard in Grid Computing

Globus allows secure remote operation **over** multiple administration domains **with different** resource management systems **and** access policies**.**

Globus is a new **high performance computing technology** to efficiently solve certain application profiles

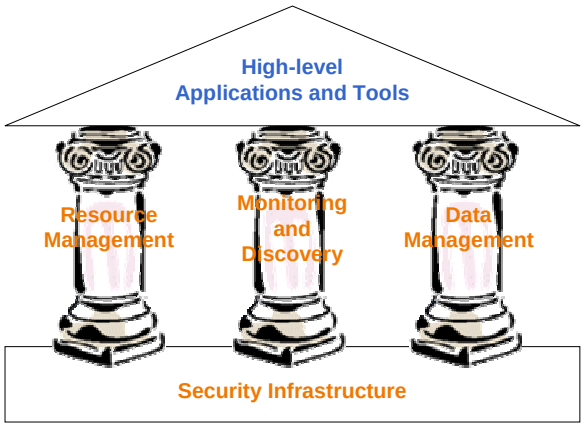
Globus is…

- a set of services, commands, libraries and APIs
- a *software* infrastructure, or ***middleware***.

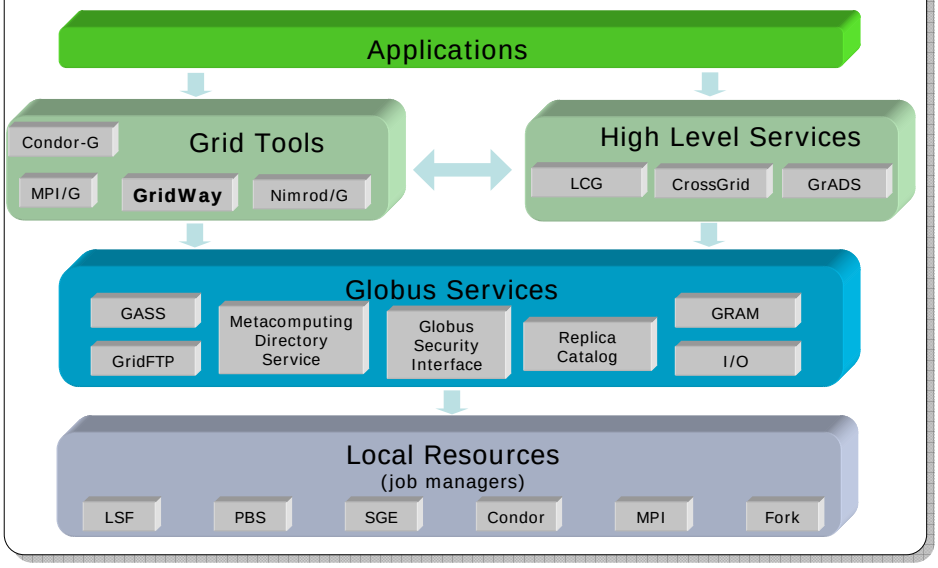
Globus is NOT...

- a scheduler, a resource broker or an application
- an end-user tool.

Globus Toolkit Main Components



Layers in a Globus-based Grid Infrastructure



Stages in the Evolution of Grid Computing

Source: Platform Computing, "The Evolution Of Grid: The Three Stages of Grid Computing". Available at http://www.platform.com/grid/evolution.asp

Description of the Stages

Grid Infrastructures in Research Projects

	Enterprise Grid	Partner Grid	Utility Grid
Infrastructure	Internal resources managed by different DRM systems that could be geographically distributed	Resources scattered across several organizations or administrative domains managed by different DRM systems	Resources provided by dedicated service providers
Objective	Enable diverse resource sharing to improve internal collaboration and achieve a better return from their IT investment	Provide large-scale, secure and reliable sharing of resources among partner organizations and supply-chain participants	Supply resources on demand
Benefits	•Cost minimization •Performance maximization	•Access to a higher computing performance to satisfy peak demands •Provide support to face collaborative projects	•Flexibility to adjust capacity •Access to unlimited computational capacity •Transform IT costs from fixed to variable

Partner Grid Infrastructures in Research Projects

- **Loosely Coupled Grids**
  - ✓Dynamic, heterogeneous and autonomous resources interconnected by public networks

CrossGrid, EGEE Enabling Grids for E-sciencE, IrisGrid, NorduGrid, GridLab

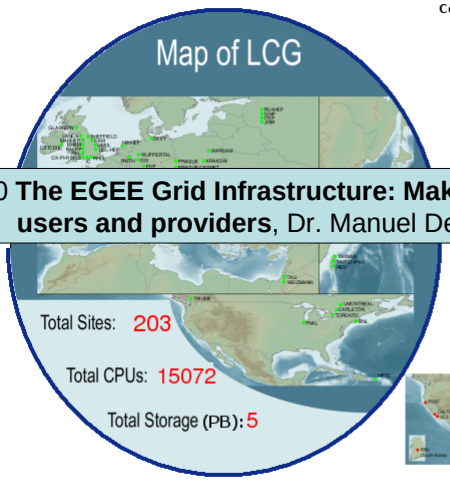
- **Tightly Coupled Grids**
  - ✓Static, homogeneous and dedicated resources interconnected by dedicated networks

TeraGrid, Information Power Grid

Application Profile

- **HTC applications and complex flows**
  - ✓They usually require heavy file transferring
- **HPC applications are not suitable for loosely coupled grids** due to the dynamic and heterogeneous performance provided by the computing and interconnection resources

www.eu-**egee**.org



Collaborating with LCG

15:00 **The EGEE Grid Infrastructure: Making the Grid easier for users and providers**, Dr. Manuel Delfino, PIC-UAB

September 2005

CrossGrid
http://www.crossgrid.org

EU-DataGrid
http://www.eu-datagrid.org

FlowGrid
http://www.unizar.es/flowgrid

Damien
http://www.hlrs.de/organization/pds/projects/damien/

iAstro: Cost Action
http://main.cs.qub.ac.uk/~fmurtagh/iastro/

CESGA-CESCA Grid
http://www.cesga.es/Novas/defaultL.html?2003/2003_07_28.html&2/

Contents

User on a Globus Grid

Where do I execute my job?	resource selection
What do I need (files, ...)?	job preparation
How do I execute my job?	job submission
How is my job doing?	job monitoring
Can I use a better host?	job migration
How do I retrieve job output?	job termination

Goal

To provide an **unattended** and more **efficient** execution of jobs (***submit & forget***) on ***heterogeneous***, ***dynamic*** and ***loosely coupled*** Grids.

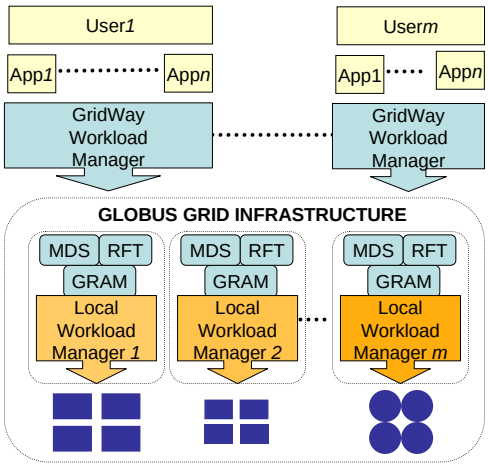
Design Guidelines

- **Easy to Adapt** (modular design)
- **Easy to Scale** (decentralized architecture)
- **Easy to Deploy** (user, standard services)

Features

- Adaptive scheduling
- Adaptive execution
- Self-adaptive applications
- Fault-tolerance

GridWay on Top of Globus Provides Decoupling between Applications and the Underlying Local Management System



GridWay Functionality

Adaptive Scheduling

- **Dynamic Scheduling** (resource characteristics and availability)
- A better resource is discovered, ex. new resource added or became free (**opportunistic migration**)

Adaptive Execution (*on-request migration*)

- A performance degradation is detected **(performance contract violation)**
- The application preferences or requirements changes **(self-migration)**
- The **user** requests a migration

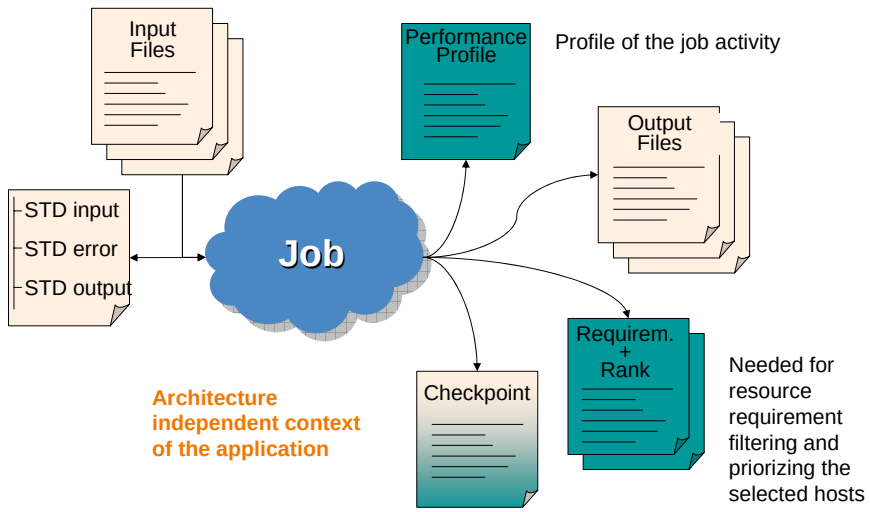
Fault Tolerance

- The **remote host** fails (container failures, OS crash...)
- **Network** failure
- The job is canceled.

Globus Resource Management Ecosystem

		GridWay	Condor/G	Nimrod/G	LCG
Adpative Scheduling	Dynamic Scheduling	Discover & Selection	Static	Selection	Discover & Selection
	Opportunistic Migration	✓	✗	✗	✗
Adaptive Execution	Performance Contracts	✓	✗	✗	✗
	Self scheduling	✓	✗	✗	✗
	User Migration	✓	✗	✗	✗
Fault Tolerance	Resource Failure	✓	✓	✓	✓
	Network Failure	✓	✓	✓	✓
	Job cancellation	✓	✓	✓	✓

The Job Model



Job Template

```
# Scheduling Variables
DISCOVERY_TIME    = 60
DISCOVERY_TIMEOUT = 1200

# Performance Variables
POLL_TIMEOUT       = 60
SUSPENSION_TIMEOUT = 3000

# Behaviour in case of failure
ON_FAILURE        = reschedule
NUMBER_OF_RETRIES = 3

# Executable Variables
EXECUTABLE_FILE      =   /bin/ls
EXECUTABLE_ARGUMENTS =  -la
INPUT_FILES        =
OUTPUT_FILES    =
RESTART_FILES =

# Standard Streams
STDIN_FILE  =   /dev/null
STDOUT_FILE =   stdout_file.${GW_JOB_ID}
STDERR_FILE =   stderr_file.${GW_JOB_ID}

# Execution Modules
RESOURCE_SELECTOR  = scripts/rs_round_robin.sh

# Driver specific information and user-defined and module specific variables
...
```


User Interface (Unix-like)

- **gwps:** display job information and status

```
JID AID TID DM          SM          GSM         STIME ETIME EXETIME EXIT HOST    TEMPLATE
0   --  --  submitted   prologue    --          --:-- --:-- --:--   --   columba SP.A
1   --  --  zombie      done        --          27:37 28:07 00:30   0    ursa    BT.A
7   --  --  pending     done        --          --:-- --:-- --:--   --   draco   SP.A
```

- **gwhistory:** display job execution history

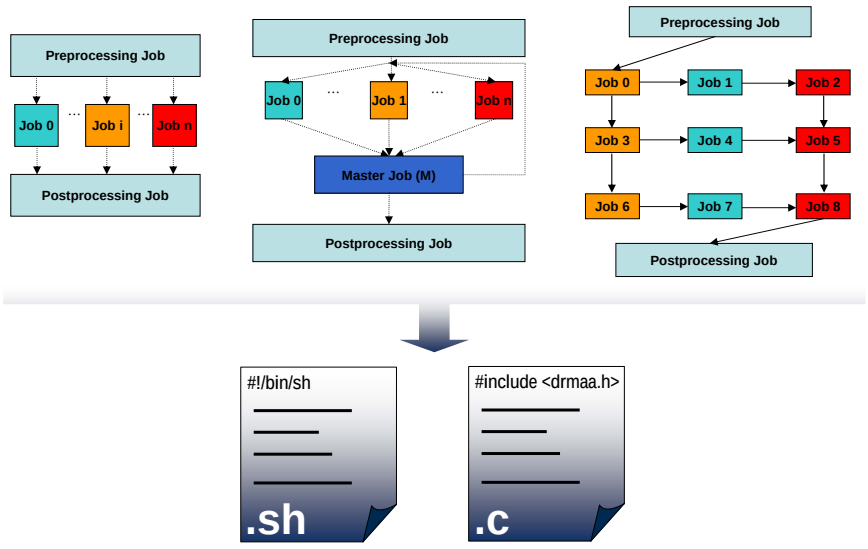
```
HOST                  RANK STIME      ETIME      EXETIME     MIGRATION_REASON
columba.dacya.ucm.es 100  --:--      --:--      --:--       --
ursa.dacya.ucm.es/jobmanager-grd 50   27:41      27:52       00:11       discovery timeout
```

- **gwkill:** signals a job (kill, stop, resume, reschedule)
- **gwsubmit:** submits a job, or an array job
- **gwwait:** waits for zombie state of a job (any, all, set)

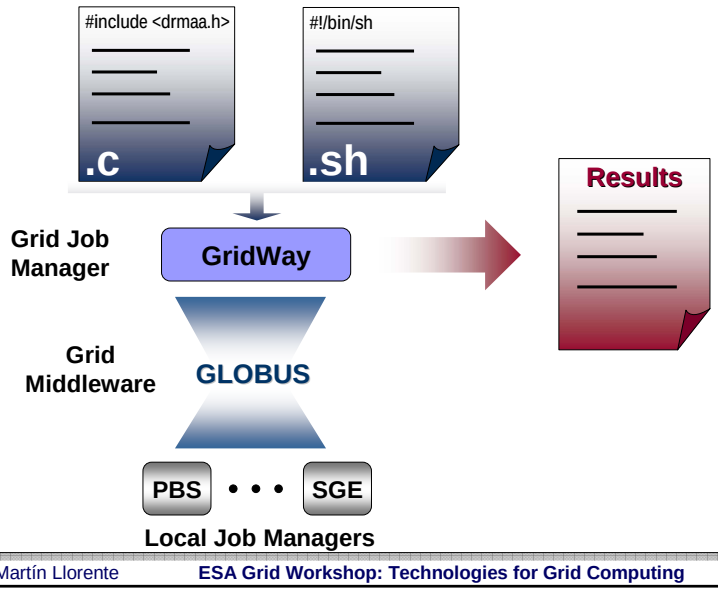
Client API

Handles **job submission**, **monitoring** and **control**, and retrieval of **finished job status**. ***(DRMAA by GGF Working Group)***

Complex Application Definition

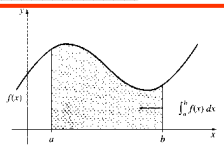


Complex Application Execution

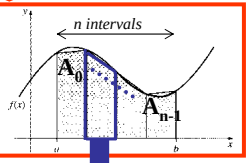


Numerical Approximation

$$\pi = \int_0^1 \frac{4}{1+x^2} dx$$

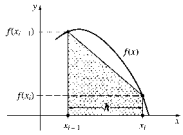


$$\pi = \sum_{i=0}^{n-1} A_i$$



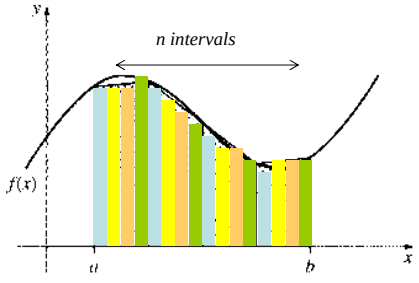
Numerical Integration

Area in Interval *i*

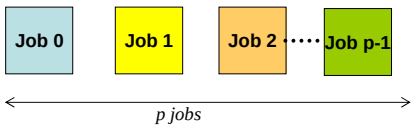


$$A_i = f(x_i + \frac{h}{2})h$$

Interval Scheduling



Cyclic distribution



The Job

Template

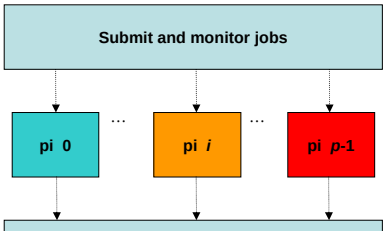
```
ON_FAILURE = reschedule
EXECUTABLE_FILE=pi
EXECUTABLE_ARGUMENTS="${GW_TASK_ID} ${GW_TOTAL_TASKS} 1000000000"
STDOUT_FILE=stdout.${GW_TASK_ID}
STDERR_FILE=stderr.${GW_TASK_ID}
RESOURCE_SELECTOR  = scripts/rs_round_robin.sh
GW_HOST_LIST=$HOME/DRMAA_Tutorial/pi_script/host.list
```

pi *rank p n*

```
…
int main (int nargs, char** args)
{
    …
    rank  = atoi(args[1]);
    total = atoi(args[2]);
    n     = atoi(args[3]);
        h = 1.0/n;
    l_sum = 0.;
    for (i = rank; i < n; i += total)
    {
        x = (i+0.5)*h;
        l_sum += 4.0/(1.0+x*x);
    }
    l_sum *= h;
    printf("%g\n",l_sum);
    return 0;
}
```


Command Interface

```
gwsubmit –n 20 –t job_template
gwps –cd 1
gwwait –A 0
```



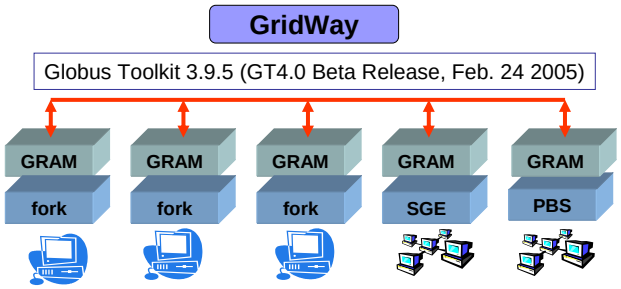

```
my $i = 0;
my $content;
my $pi = 0;
while ($i < $num_of_files)
{
        open (FILE,"stdout.$i");
        $content = <FILE>;
        close (FILE);
        $pi = $pi + $content;
        $i++;
}
print "Pi is: $pi\n";
```

Distributed Resource Management Application API

- The DRMAA specification constitutes a **homogenous interface** to different **DRMS** to handle job submission, monitoring and control, and retrieval of finished job status. Moreover, DRMAA has been developed by DRMAA-WG within the Global Grid Forum (GGF).
- The DRMAA standard represents a suitable and portable framework to express this kind of distributed computations.
- Some DRMAA interface routines:
  - Initialization and finalization routines: **`drmaa_init`** and **`drmaa_exit`**.
  - Job submission routines: **`drmaa_run_job`** and **`drmaa_run_bulk_jobs`**.
  - Job control and monitoring routines: **`drmaa_control`**, **`drmaa_synchronize`**, **`drmaa_wait`** and **`drmaa_job_ps`**.
- DRMAA interface routines has been implemented within SGE, Condor and **GridWay**

A Research TestBed

					DRMS	
Name	Processor	Speed	O.S.	Nodes	pre-WS	WS
ursa	Intel P4 HT	3.2 Ghz	Linux 2.4	1	fork	fork
draco	Intel P4 HT	3.2 Ghz	Linux 2.4	1	fork	fork
cygnus	Intel P4	2.5 Ghz	Linux 2.4	1	fork	fork
hydrus	Intel P4 HT	3.2 Ghz	Linux 2.4	4	PBS	PBS
aquila	Intel P3	600 Mhz	Linux 2.4	2	**SGE**	**fork**



Computing π on pre-WS Components

Resource Selector: Round Robin
Number of Jobs: 20
Number of Intervals: 10^9
Execution Time on `hydrus`: 12 minutes

Execution Time on the Testbed: 3 minutes

```
gwps -cd 1
```

```
JID AID TID DM   SM   EM   STIME    ETIME    EXETIME XFRTIME EXIT TEMPLATE        HOST
0   0   0   zomb done done 13:25:41 13:26:58 0:00:22 0:00:28 0    pi_template     draco.dacya.ucm.es
1   0   1   zomb done done 13:25:41 13:26:59 0:00:24 0:00:27 0    pi_template     ursa.dacya.ucm.es
2   0   2   zomb done done 13:25:41 13:27:14 0:00:38 0:00:28 0    pi_template     hydrus.dacya.ucm.es/jobmanager-pbs
3   0   3   zomb done done 13:25:41 13:27:04 0:00:26 0:00:30 0    pi_template     hydrus.dacya.ucm.es/jobmanager-pbs
4   0   4   zomb done done 13:25:41 13:27:02 0:00:25 0:00:29 0    pi_template     hydrus.dacya.ucm.es/jobmanager-pbs
5   0   5   zomb done done 13:25:41 13:27:03 0:00:25 0:00:30 0    pi_template     hydrus.dacya.ucm.es/jobmanager-pbs
6   0   6   zomb done done 13:25:41 13:27:20 0:00:38 0:00:34 0    pi_template     cygnus.dacya.ucm.es
7   0   7   zomb done done 13:25:41 13:28:14 0:01:33 0:00:33 0    pi_template     aquila.dacya.ucm.es/jobmanager-sge
8   0   8   zomb done done 13:25:41 13:27:58 0:01:21 0:00:29 0    pi_template     aquila.dacya.ucm.es/jobmanager-sge
9   0   9   zomb done done 13:25:41 13:27:54 0:00:21 0:00:25 0    pi_template     draco.dacya.ucm.es
10  0   10  zomb done done 13:25:41 13:27:54 0:00:22 0:00:24 0    pi_template     ursa.dacya.ucm.es
11  0   11  zomb done done 13:25:41 13:28:00 0:00:24 0:00:28 0    pi_template     hydrus.dacya.ucm.es/jobmanager-pbs
12  0   12  zomb done done 13:25:41 13:27:59 0:00:23 0:00:28 0    pi_template     hydrus.dacya.ucm.es/jobmanager-pbs
13  0   13  zomb done done 13:25:41 13:27:57 0:00:24 0:00:25 0    pi_template     hydrus.dacya.ucm.es/jobmanager-pbs
14  0   14  zomb done done 13:25:41 13:28:28 0:00:22 0:00:28 0    pi_template     hydrus.dacya.ucm.es/jobmanager-pbs
15  0   15  zomb done done 13:25:41 13:28:39 0:00:32 0:00:29 0    pi_template     cygnus.dacya.ucm.es
16  0   16  zomb done done 13:25:41 13:28:52 0:00:22 0:00:22 0    pi_template     draco.dacya.ucm.es
17  0   17  zomb done done 13:25:41 13:28:53 0:00:22 0:00:23 0    pi_template     ursa.dacya.ucm.es
18  0   18  zomb done done 13:25:41 13:28:56 0:00:24 0:00:24 0    pi_template     hydrus.dacya.ucm.es/jobmanager-pbs
19  0   19  subm eplg actv 13:25:41 --:--:-- 0:00:23 0:00:25 --   pi_template     hydrus.dacya.ucm.es/jobmanager-pbs
```


Computing π on WS Components

Resource Selector: Round Robin
Number of Jobs: 20
Number of Intervals: 10^9
Execution Time on `hydrus`: 12 minutes

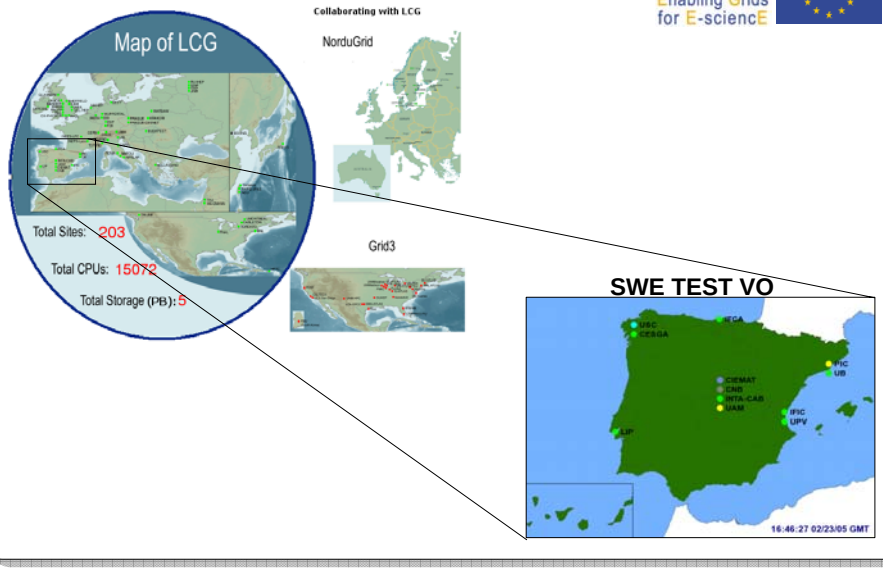
Execution Time on the Testbed: 7 minutes
(fewer resources than pre-WS execution)

```
gwps -cd 1
```

```
JID AID TID DM   SM   EM   STIME    ETIME    EXETIME XFRTIME EXIT TEMPLATE        HOST
0   0   0   zomb done done 20:16:14 20:18:08 0:00:42 0:00:55 0    pi_template     draco.dacya.ucm.es
1   0   1   zomb done done 20:16:14 20:18:13 0:00:37 0:01:05 0    pi_template     ursa.dacya.ucm.es
2   0   2   subm wrap actv 20:16:14 --:--:-- 0:06:33 0:00:56 --   pi_template     hydrus.dacya.ucm.es/PBS
3   0   3   zomb done done 20:16:14 20:18:50 0:00:50 0:01:29 0    pi_template     hydrus.dacya.ucm
4   0   4   zomb done done 20:16:14 20:18:50 0:00:50 0:01:28 0    pi_template     hydrus.
5   0   5   zomb done done 20:16:14 20:18:50 0:00:54 0:01:25 0    pi_template
6   0   6   zomb done done 20:16:14 20:18:36 0:00:46 0:01:19 0    pi_templ
7   0   7   zomb done done 20:16:14 20:20:14 0:01:54 0:01:49 0
8   0   8   zomb done done 20:16:14 20:20:37 0:00:34 0:01
9   0   9   zomb done done 20:16:14 20:19:51 0:00
10  0   10  zomb done done 20:16:14 20:20
11  0   11  zomb done done 20:16
12  0   12  zomb done do
13  0   13  zomb
14  0
1
1
17  0   17  zomb done done 20:16:14 20:22:26 0:00:36 0:00:49 0    pi_template     hydrus.dacya.ucm.es/PBS
18  0   18  zomb done done 20:16:14 20:22:41 0:00:33 0:01:04 0    pi_template     ursa.dacya.ucm.es
19  0   19  zomb done done 20:16:14 20:23:25 0:00:50 0:01:13 0    pi_template     cygnus.dacya.ucm.es
```

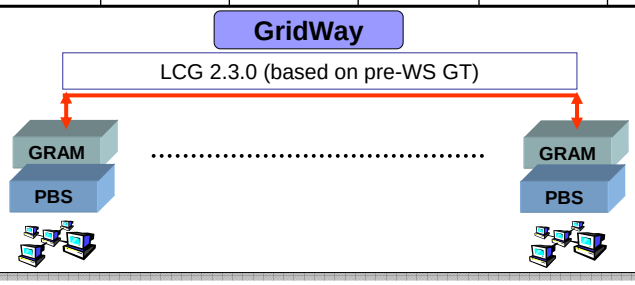
GridWay is able to simultaneously harness pre-WS and WS

A Production TestBed



A Production TestBed

Name	Site	Node	Processor	Speed	O.S.	Nodes	DRMS
ramses	UPV	Valencia	2 x Intel PIII	900 Mhz	Linux 2.4	12	PBS
ce00	CAB-INTA	Madrid	Intel P4 HT	2.8 Ghz	Linux 2.4	8	PBS
mallarme	CNB	Madrid	4 x Xeon	2.0 Ghz	Linux 2.4	16	PBS
ce2	CESGA	Santiago	2 x Intel PIII	1.0 Ghz	Linux 2.4	6	PBS
ce01	LIP	Lisboa	2 x Intel PIII	800 Mhz	Linux 2.4	8	PBS
gtbcg12	IFCA	Santander	2 x Intel PIII	1.2 Ghz	Linux 2.4	34	PBS
lcg2ce	IFIC	Valencia	AMD Athlon	1.2 Ghz	Linux 2.4	117	PBS
ce01	PIC	Barcelona	Intel P4 HT	3.4 Ghz	Linux 2.4	20	PBS



Computing π on a Production Testbed

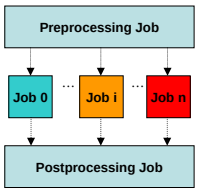
Resource Selector: Round Robin
Number of Jobs: 20
Number of Intervals: 10^9
Execution Time on hydrus: 12 minutes

Execution Time on the Testbed: 6 minutes

```
gwps -cd 1
```

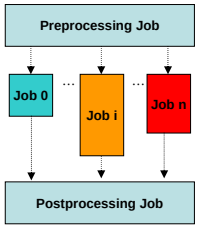
```
JID AID TID DM   SM   EM   STIME    ETIME    EXETIME XFRTIME EXIT TEMPLATE        HOST
40  2   0   zomb done done 16:01:52 16:05:02 0:01:57 0:00:50 0    pi_template     ce00.inta.es/jobmanager-torque
41  2   1   zomb done done 16:01:52 16:05:00 0:01:36 0:01:09 0    pi_template     ce00.inta.es/jobmanager-torque
42  2   2   zomb done done 16:01:52 16:05:04 0:01:48 0:01:01 0    pi_template     ce00.inta.es/jobmanager-torque
43  2   3   zomb done done 16:01:52 16:05:04 0:01:30 0:01:19 0    pi_template     ce00.inta.es/jobmanager-torque
44  2   4   zomb done done 16:01:52 16:05:49 0:02:39 0:00:55 0    pi_template     lcg2ce.ific.uv.es/jobmanager-lcgpbs
45  2   5   zomb done done 16:01:52 16:05:56 0:02:42 0:00:59 0    pi_template     lcg2ce.ific.uv.es/jobmanager-lcgpbs
46  2   6   zomb done done 16:01:52 16:05:56 0:02:45 0:00:56 0    pi_template     lcg2ce.ific.uv.es/jobmanager-lcgpbs
47  2   7   zomb done done 16:01:52 16:05:53 0:02:45 0:00:53 0    pi_template     lcg2ce.ific.uv.es/jobmanager-lcgpbs
48  2   8   zomb done done 16:01:52 16:04:11 0:01:08 0:00:48 0    pi_template     ramses.dsic.upv.es/jobmanager-torque
49  2   9   zomb done done 16:01:52 16:03:58 0:00:57 0:00:46 0    pi_template     ramses.dsic.upv.es/jobmanager-torque
50  2   10  zomb done done 16:01:52 16:04:19 0:01:06 0:00:58 0    pi_template     ramses.dsic.upv.es/jobmanager-torque
51  2   11  zomb done done 16:01:52 16:04:04 0:00:52 0:00:57 0    pi_template     ramses.dsic.upv.es/jobmanager-torque
52  2   12  zomb done done 16:01:52 16:05:58 0:01:00 0:00:43 0    pi_template     ramses.dsic.upv.es/jobmanager-torque
53  2   13  zomb done done 16:01:52 16:05:57 0:01:00 0:00:42 0    pi_template     ramses.dsic.upv.es/jobmanager-torque
54  2   14  zomb done done 16:01:52 16:05:58 0:01:03 0:00:40 0    pi_template     ramses.dsic.upv.es/jobmanager-torque
55  2   15  zomb done done 16:01:52 16:06:28 0:01:04 0:00:39 0    pi_template     ramses.dsic.upv.es/jobmanager-torque
56  2   16  zomb done done 16:01:52 16:07:59 0:01:54 0:00:50 0    pi_template     ce00.inta.es/jobmanager-torque
57  2   17  zomb done done 16:01:52 16:07:56 0:01:54 0:00:47 0    pi_template     ce00.inta.es/jobmanager-torque
58  2   18  zomb done done 16:01:52 16:07:58 0:01:49 0:00:54 0    pi_template     ce00.inta.es/jobmanager-torque
59  2   19  subm eplg actv 16:01:52 --:--:-- 0:01:47 0:00:57 --   pi_template     ce00.inta.es/jobmanager-torque
```


Different Execution Profile of the Use Cases



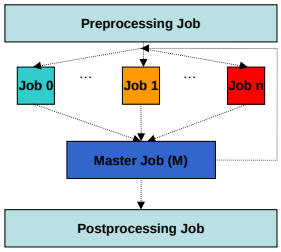
HTC Synchronous

Computational Proteomics



HTC Asynchronous

Impact Cratering Simulation



Master-slave

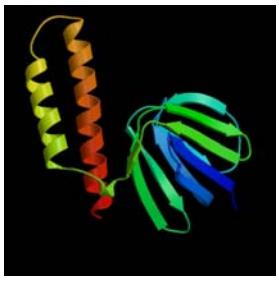
Genetic Algorithm

The Application: Computational Proteomics

- **Synchronous High Throughput Computing with 80 tasks**
  - **Each task performs protein structure prediction and thermodynamic studies** from aminoacid sequences (PDB) by means of *threading* methods
  - Application to a family of 80 **orthologous proteins**

–MTYHLDVVSAEQQMFSGLVEKIQVTGSEGELGIYPGHAPLLTAIKPGMIRIVK
QHGHEEFIYLSGGILEVQPGNVTVLADTAIRGQDLDEARAMEAKRKAEEHISS
SHGDVDYAQASAELAKAIAQLRVIELTKK

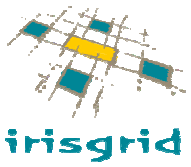
ATP Sintase (ε chain)



Triose Phosphate Isomerase

The Grid Infrastructure: IRISGrid and EGEE

Testbed	Site	Resource	Processor	Speed	Nodes	RM
IRISGrid	RedIRIS	heraclito	Intel Celeron	700MHz	1	Fork
		platon	2×Intel PIII	1.4GHz	1	Fork
		descartes	Intel P4	2.6GHz	1	Fork
		socrates	Intel P4	2.6GHz	1	Fork
	DACYA-UCM	aquila	Intel PIII	700MHz	1	Fork
		cepheus	Intel PIII	600MHz	1	Fork
		cygnus	Intel P4	2.5GHz	1	Fork
		hydrus	Intel P4	2.5GHz	1	Fork
	LCASAT-CAB	babieca	Alpha EV67	450MHz	30	PBS
	CESGA	bw	Intel P4	3.2GHz	80	PBS
	IMEDEA	llucalcari	AMD Athlon	800MHz	4	PBS
	DIF-UM	augusto	4×Intel Xeon**	2.4GHz	1	Fork
		caligula	4×Intel Xeon**	2.4GHz	1	Fork
		claudio	4×Intel Xeon**	2.4GHz	1	Fork
	BIFI-UNIZAR	lxsrv1	Intel P4	3.2GHz	50	SGE
EGEE	LCASAT-CAB	ce00	Intel P4	2.8GHz	8	PBS
	CNB	mallarme	2×Intel Xeon	2.0GHz	8	PBS
	CIEMAT	lcg02	Intel P4	2.8GHz	6	PBS
	FT-UAM	grid003	Intel P4	2.6GHz	49	PBS
	IFCA	gtbcg12	2×Intel PIII	1.3GHz	34	PBS
	IFIC	lcg2ce	AMD Athlon	1.2GHz	117	PBS
	PIC	lcgce02	Intel P4	2.8GHz	69	PBS



7 sites and 195 CPUs

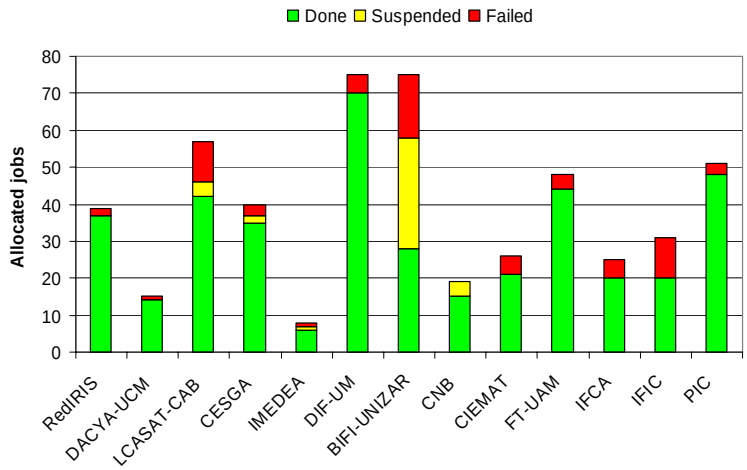
7 sites and 333 CPUs

Total: 13 sites and 528 CPUs. Limitation of 4 running jobs per resource (64 CPUs)

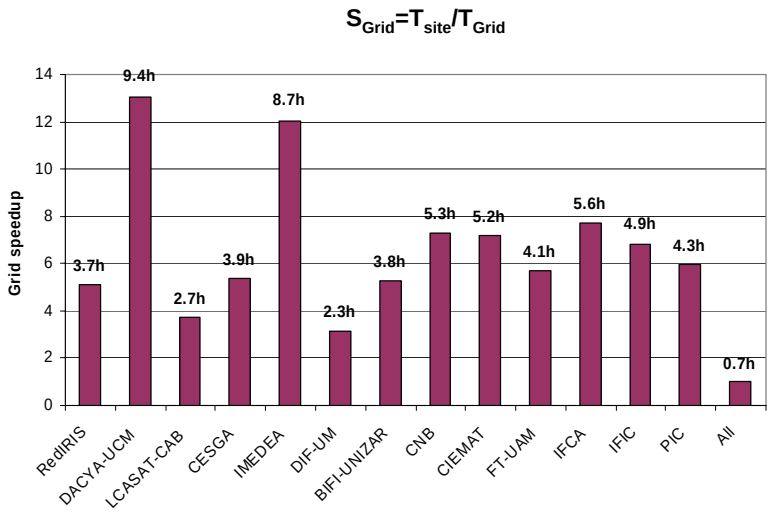
The Results: Dynamic Throughput

The Results: Scheduling

- **Aggregated schedule** performed during the five experiments



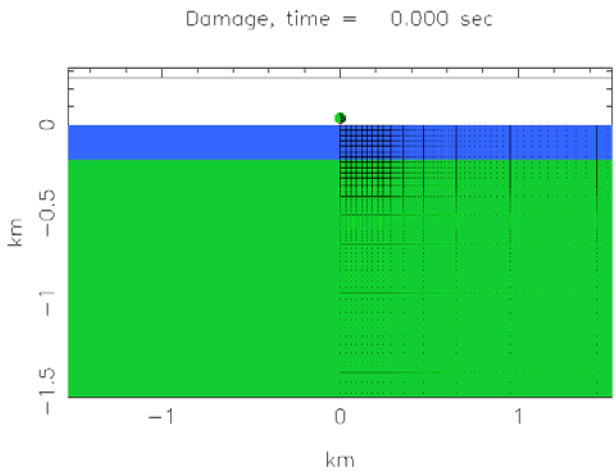
The Results: Grid Speedup per Site



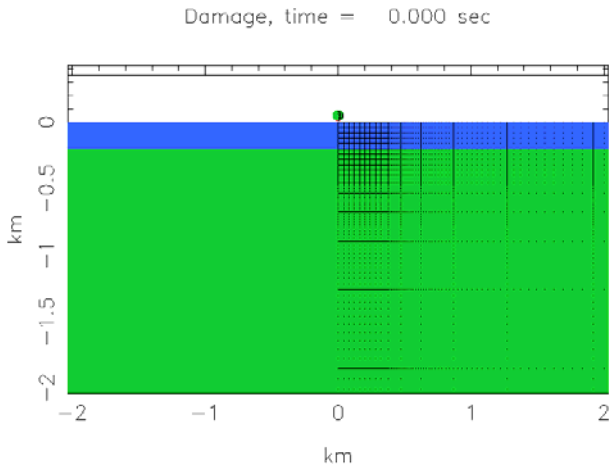
The Application: Impact Cratering Simulation

- **Impact cratering is a geological process of special interest** in Astrobiology that affects the surface of nearly all celestial bodies.
- Marine-target impact cratering simulation plays an important role in the **study of past martian seas**. A water layer at the target influences lithology and morphology of the resultant crater.
- **Asynchronous High Throughput Computing with 72 tasks**
  - The application analyzes the **threshold impactor diameter** for cratering the seafloor of an hypothetical martian sea
  - The **search space** of input parameters includes the projectile diameter itself (8 cases), the water depth (3 cases) and the impactor velocity (3 cases)

D= 60m, H= 200m, V= 10Km/s



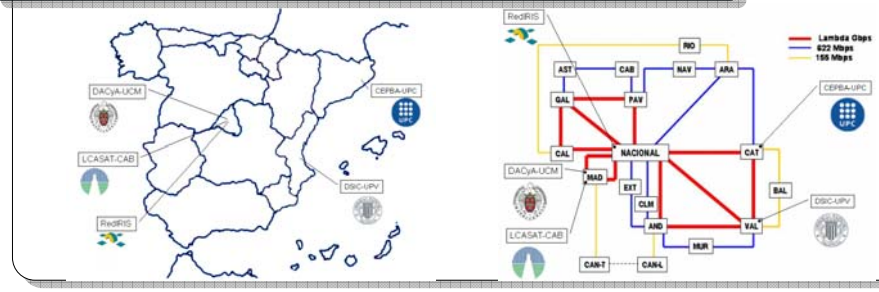
D= 80m, H= 200m, V= 10Km/s



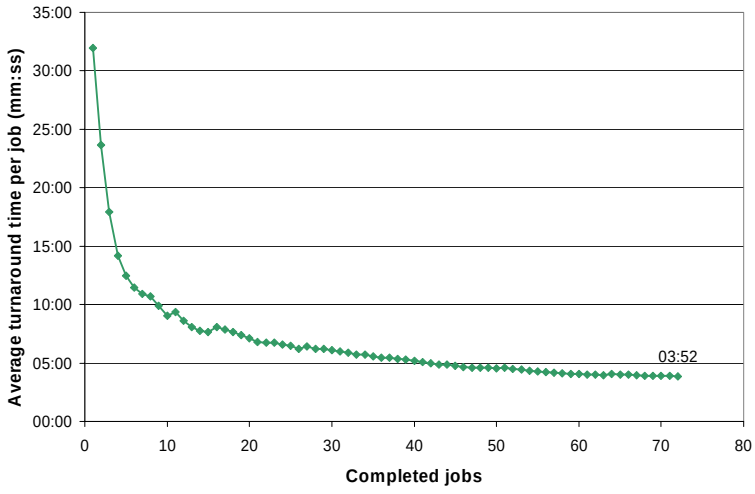
The Grid Infrastructure

Name	Site	Nodes	Model	Speed	Mem	OS	Job mgr.
hydrus	DACyA-UCM	1	Intel P4	2.5GHz	512MB	Linux 2.4	fork
cygnus		1	Intel P4	2.5GHz	512MB	Linux 2.4	fork
aquila		1	Intel PIII	700MHz	128MB	Linux 2.4	fork
babieca	LCASAT-CAB	5	Alpha EV67	450MHz	256MB	Linux 2.2	PBS
platon	RedIRIS	2	Intel PIII	1.4GHz	512MB	Linux 2.4	fork
heraclito		1	Intel Celeron	700MHz	256MB	Linux 2.4	fork
ramses	DSIC-UPV	5	Intel PIII	900MHz	512MB	Linux 2.4	PBS
khafre	CEPBA-UPC	4	Intel PIII	700MHz	512MB	Linux 2.4	fork

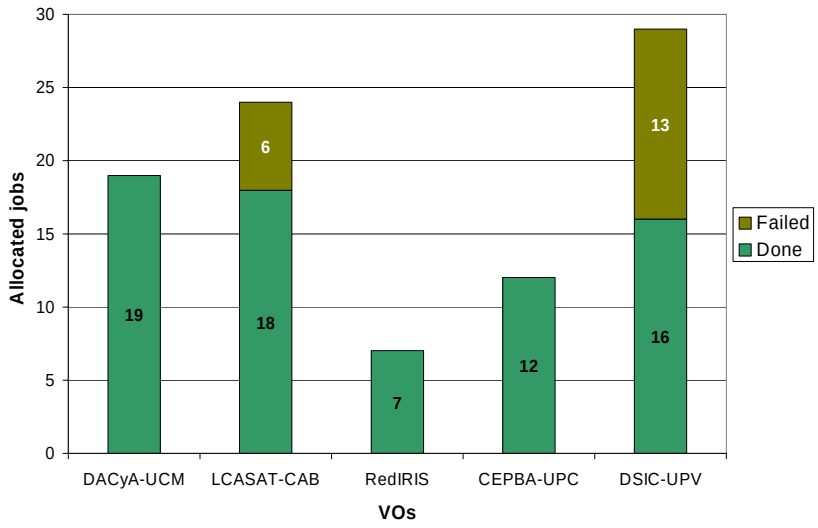
Geographical Distribution and Interconnecting Network



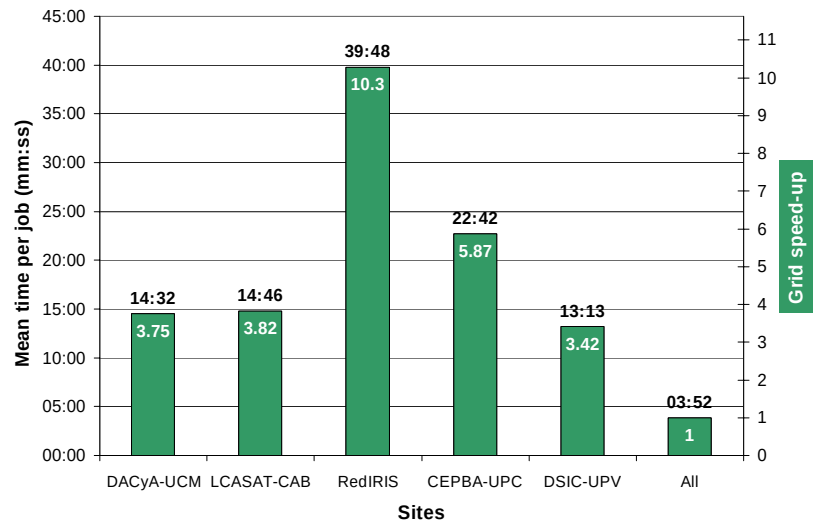
The Results: Turnaround Time per Job



The Results: Scheduling



The Results: Grid Speedup per Site



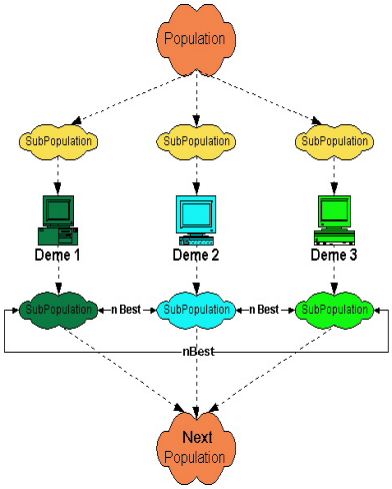
The Application: Multi-deme Genetic Algorithm

- We use a **fully connected multi-deme genetic algorithm**, all demes exchange individuals every generation.
- Its functionality and efficiency are evaluated in the solution of the **One-Max problem**.
  - One-Max problem is a **classical benchmark** problem for genetic algorithm computations, and it tries to evolve an initial matrix of zeros in a matrix of ones.

The Grid Infrastructure

Host	Model	Hz	OS	Memory	Nodes	GRAM
babieca	Alpha DS10	466Mhz	Linux 2.2	256MB	5	PBS
hydrus	**Intel Pentium 4**	**2.5 Ghz**	**Linux 2.4**	**512MB**	**1**	**fork**
cygnus	**Intel Pentium 4**	**2.5 Ghz**	**Linux 2.4**	**512MB**	**1**	**fork**
aquila	Intel Pentium III	666 Mhz	Linux 2.4	128 MB	1	fork

Algorithm Schema



Algorithm Execution

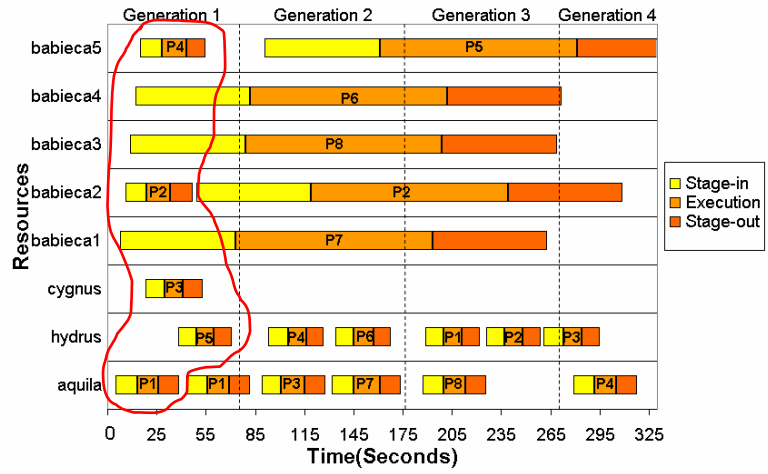
- Initial **population is uniformly distributed** among available nodes.
- Sequential **GA is locally executed over each subpopulations**.
- **Worst individuals** of each subpopulation **are exchanged with the best ones** of the rest.
- **New population is generated** to perform the next iteration.

Algorithm Optimization

- **The iteration time is given by the slowest machine.**
- Solution ⇨ ***Dynamic Connectivity***:
  - We allow **asynchronous communication pattern** between a fixed number of demes
  - Minimum number of demes in each iteration

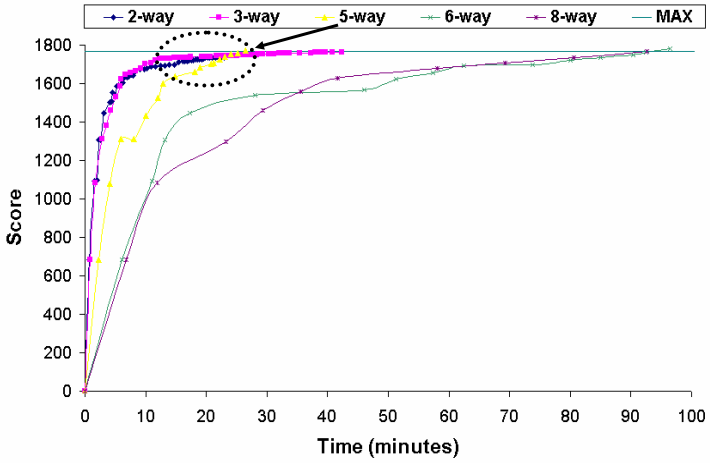
The Results: Execution Profile

- **Execution profile** of 4 generations of the GOGA, with a 5-way dynamic connectivity

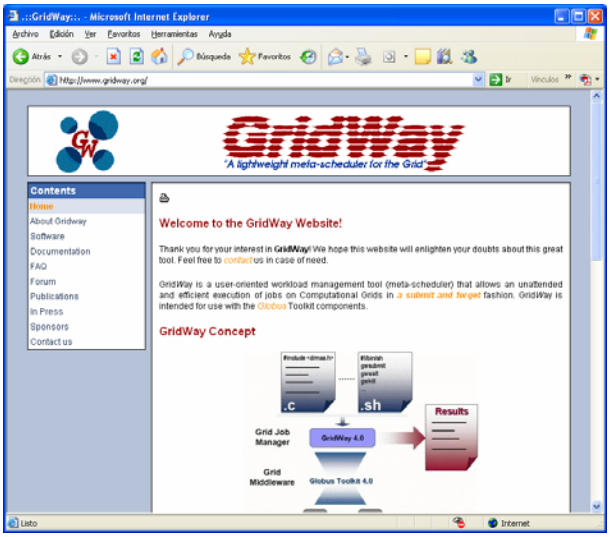


The Results: Executions with Different Degrees of Dynamic Connectivity

- 5 different executions of GOGA, with different degrees of *dynamic connectivity*: 2-way, 3-way, 5-way, 6-way and 8-way.



Information and download at http://www.GridWay.org
Open source license



Additional information about GridWay…

Grid Ecosystem at **Globus** site

Tutorial at **IBM** site

Installation on Solaris at **Sun Microsystems** site

DRMAA support and scheduling use case at **GGF** site

Thank you
for your attention!