Developing Grid-Aware Applications with DRMAA on Globus-based Grids

José Herrera Sanz
Eduardo Huedo Cuesta
Rubén Santiago Montero
Ignacio Martín Llorente

Advanced Computing Laboratory
Centro de Astrobiología
Associated to *NASA Astrobiology Institute*
CSIC-INTA

Distributed Systems Architecture and Security Group
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Outline

- Motivation
- The Grid*W*ay Framework
- DRMAA: Distributed Resource Management Application API
- Development Model
- Experiences:
  - HighThroughput Computing Application
  - Master-Worker Optimization Loop
- A Case of Study: The NAS Grid Benchmarks
  - Helical Chain
  - Visualization Pipe
- Conclusions

Motivation

Grid

- The deployment of existing applications across the **Grid** continues requiring a **high level of expertise** and a significant amount of effort, mainly due to the characteristics of the Grid: **complexity**, **heterogeneity**, **dynamism** and **high fault rate**

.

Grid*Way*

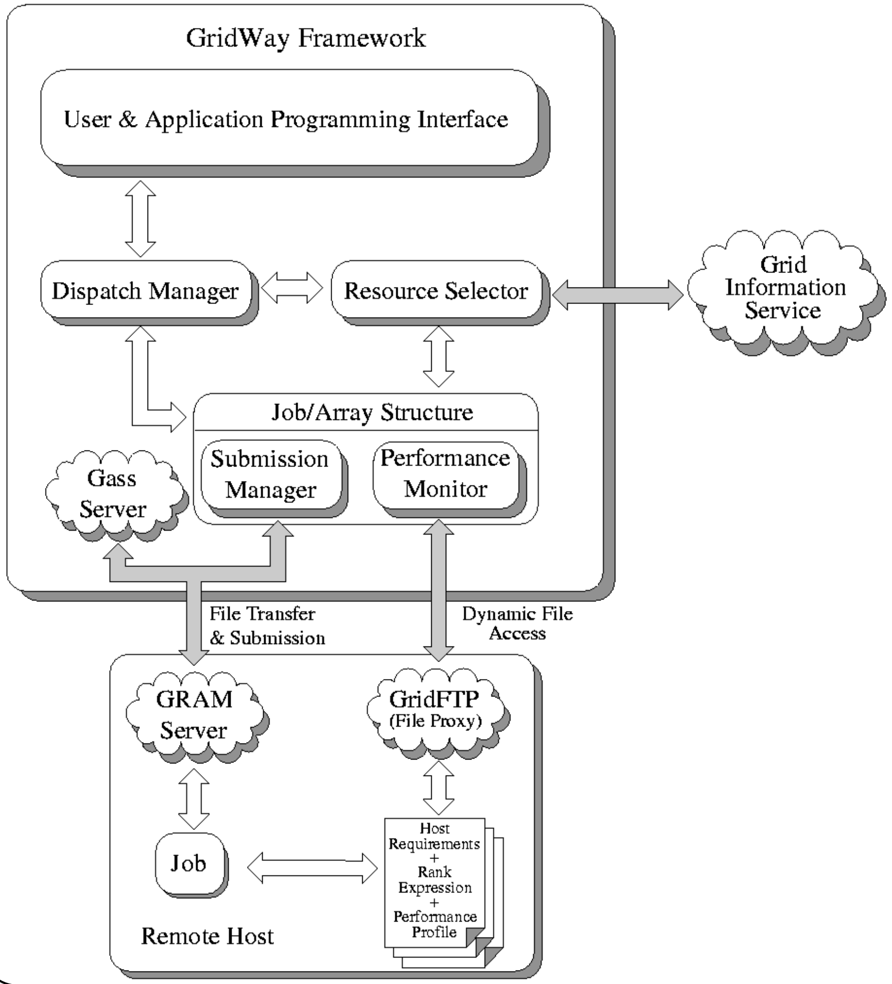
- To deal with the **characteristics** of the **Grid**, we have developed **GridWay** : a **Globus submission framework** that allows an easier and more efficient execution of jobs on dynamic Grid environments.

DRMAA

- Distributed Resource Management Application is an API specification for job submission, monitoring and control that provides a high level interface with Distributed Resource Management Systems (DRMS).
- DRMAA could aid scientists and engineers to express their computational problems by providing a portable direct interface to DRMS.

The Grid*W*ay Framework

GridWay provides an easier and more efficient execution (*submit & forget*) on heterogeneous and dynamic Grid.



Characteristics :

- **Dynamic Scheduler**: GridWay periodically adapts the scheduler to the available resources
- **Resource Selector**: Reflects the applications demands, in terms of requirements and preferences.
- **Adaptive Job Execution**: To migrate running applications to more suitable resources.
- **Fault tolerance** (callbacks) and **Job exit codes** (Job-manager).

DRMAA

Distributed Resource Management Application API

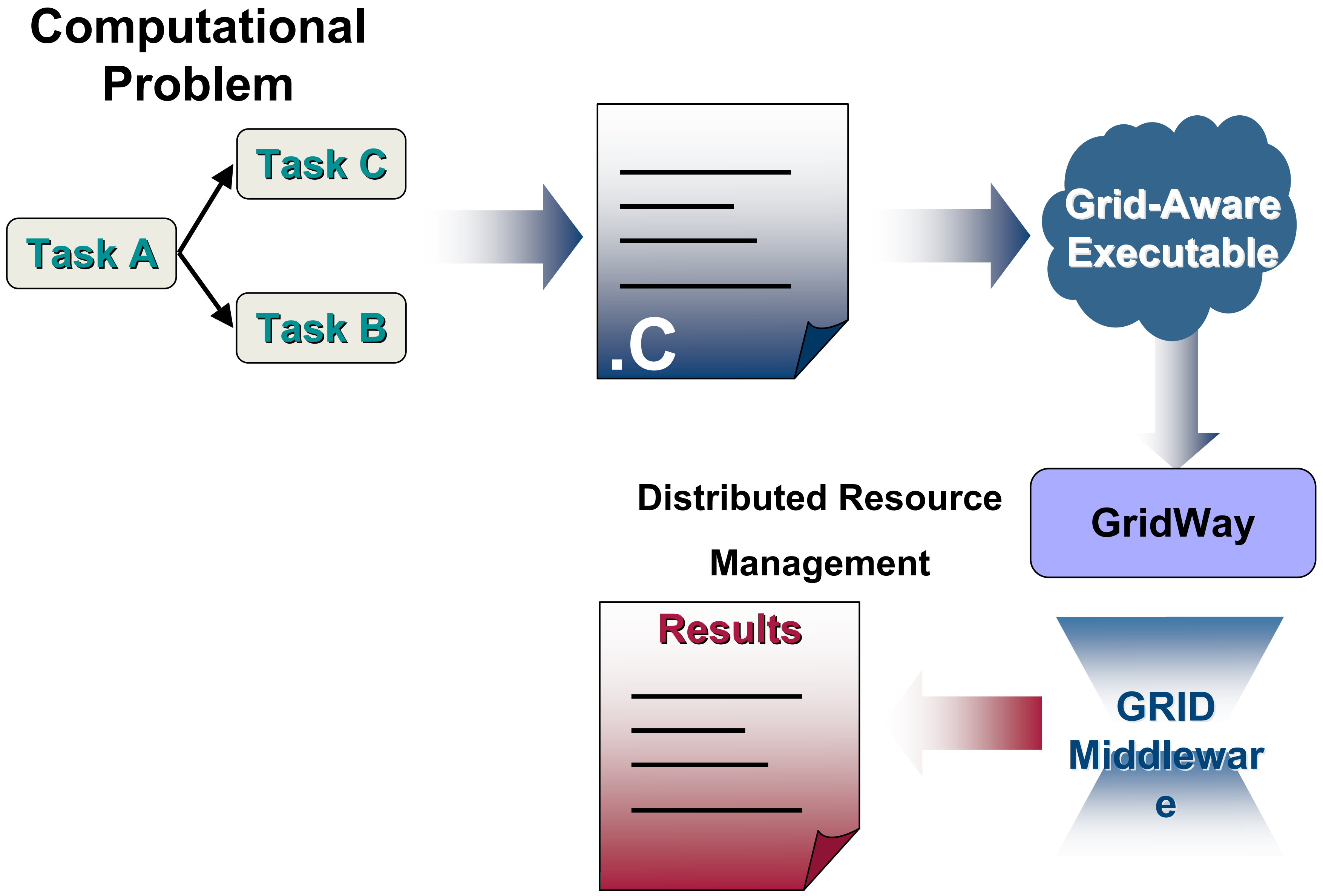
- The DRMAA specification constitutes a **homogenous interface** to different **DRMS** to handle job submission, monitoring and control, and retrieval of finished job status. Moreover, DRMAA has been developed by DRMAA-WG within the Global Grid Forum (GGF).

- 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 the **GridWay** framework.

Development Model



Experiences (1/5)

Testbed description:

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**
cepheus	Intel Pentium III	600 Mhz	Linux 2.4	256MB	1	fork
aquila	Intel Pentium III	666 Mhz	Linux 2.4	128 MB	1	fork

Objectives

- We next demonstrate the ability of the GridWay framework when executing different computational workloads distributed using DRMAA. The following examples resembles typical scientific problems whose structure is well suited to the Grid architecture.

The NAS Grid Bechmarks

- They present as a data flow graph encapsulating an instance of NAS Parallel Benchmarks code in each graph node, which communicates with other nodes sending/receiving initialization data.

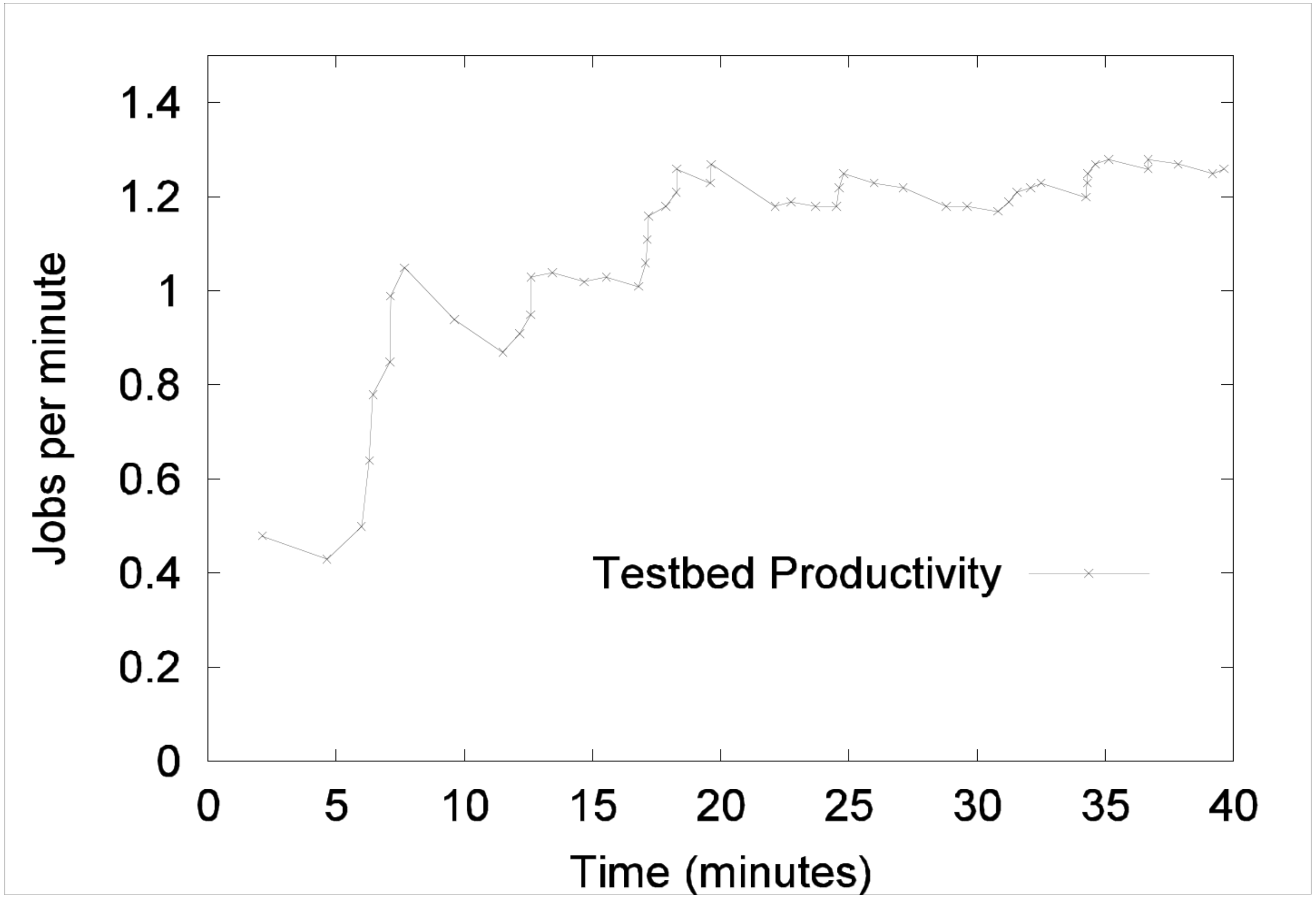
Experiences (2/5)

High-Throughput Computing Application

- It constitutes multiple independent runs of the same program, but with different input parameters.

```
rc = drmaa_init(contact, err);
// Execute initial job and wait for it
rc = drmaa_run_job(job_id, jt, err);
rc = drmaa_wait(job_id, &stat, timeout,
rusage, err);
// Execute n jobs simultaneously and wait
rc = drmaa_run_bulk_jobs(job_ids,jt,1,
JOB_NUM,1,err);
rc = drmaa_synchronize(job_ids, timeout,
1, err);
// Execute final job and wait for it
rc = drmaa_run_job(job_id, jt, err);
rc = drmaa_wait(job_id, &stat, timeout,
rusage, err);
rc = drmaa_exit(err_diag);
```

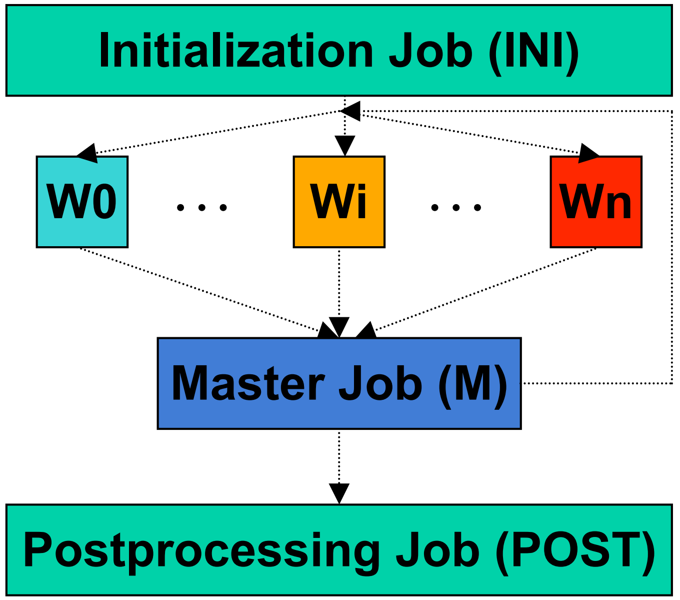
Experiences (3/5)



Experiences (4/5)

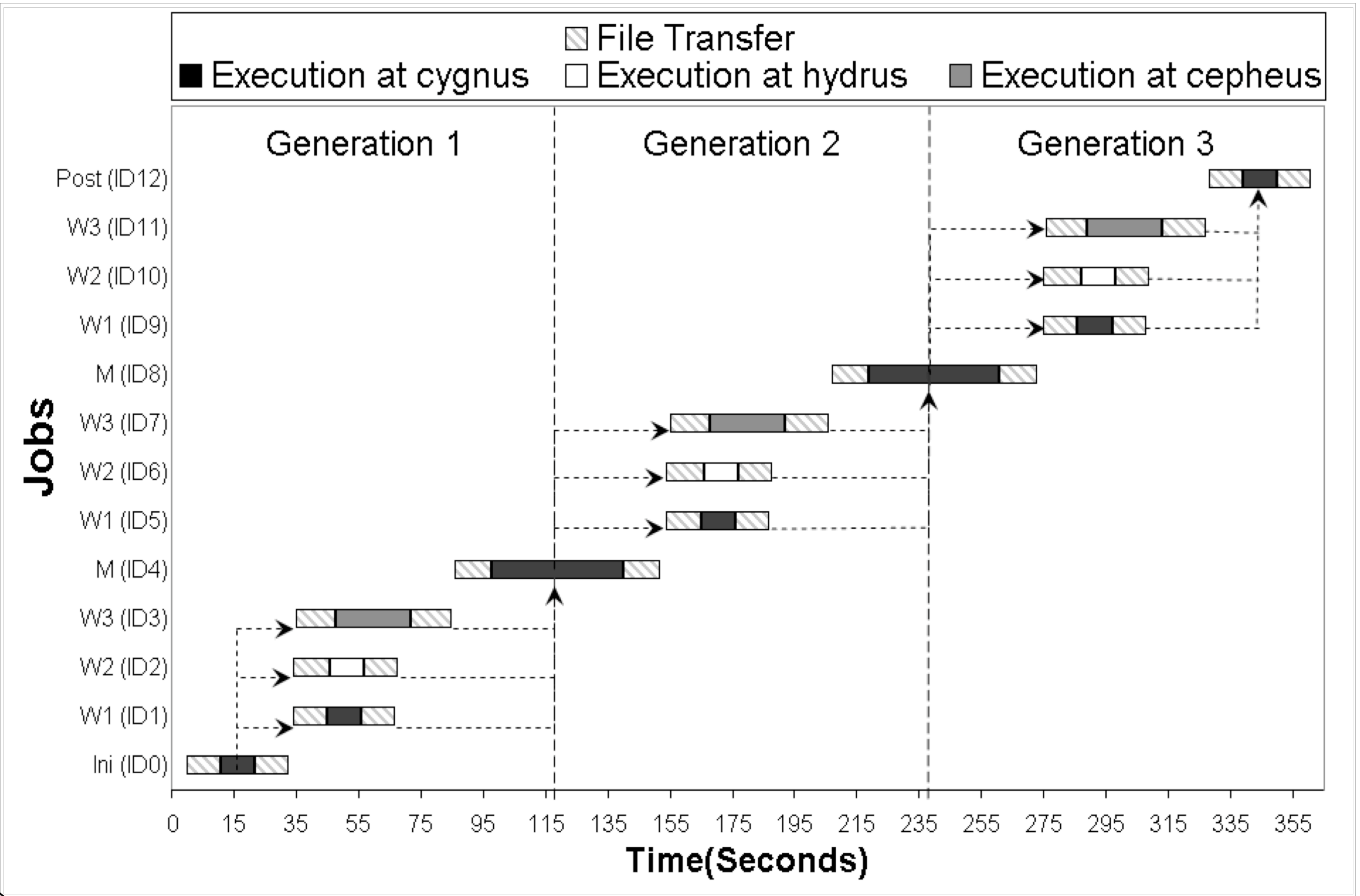
Master-Worker Optimization Loop

- A Master process assigns a description (input files) of the task to be performed by each Worker.
- Once all the Workers are completed, the Master process performs some computations in order to evaluate a stop criterion or to assign new tasks to more workers.



```
// Execute initial job and wait for it
rc = drmaa_run_job(job_id, jt, err_diag);
rc = drmaa_wait(job_id, &stat, timeout, rusage,
err_diag);
while (exitstatus != 0) {
// Execute n Workers concurrently and wait
rc = drmaa_run_bulk_jobs(job_ids, jt, 1,
JOB_NUM, 1, err_diag);
rc = drmaa_synchronize(job_ids, timeout, 1,
err_diag);
// Execute the Master, wait and get exit code
rc = drmaa_run_job(job_id, jt, err_diag);
rc = drmaa_wait(job_id, &stat, timeout, rusage,
err_diag);
rc = drmaa_wexitstatus(&exitstatus,
stat, err_diag);}
```

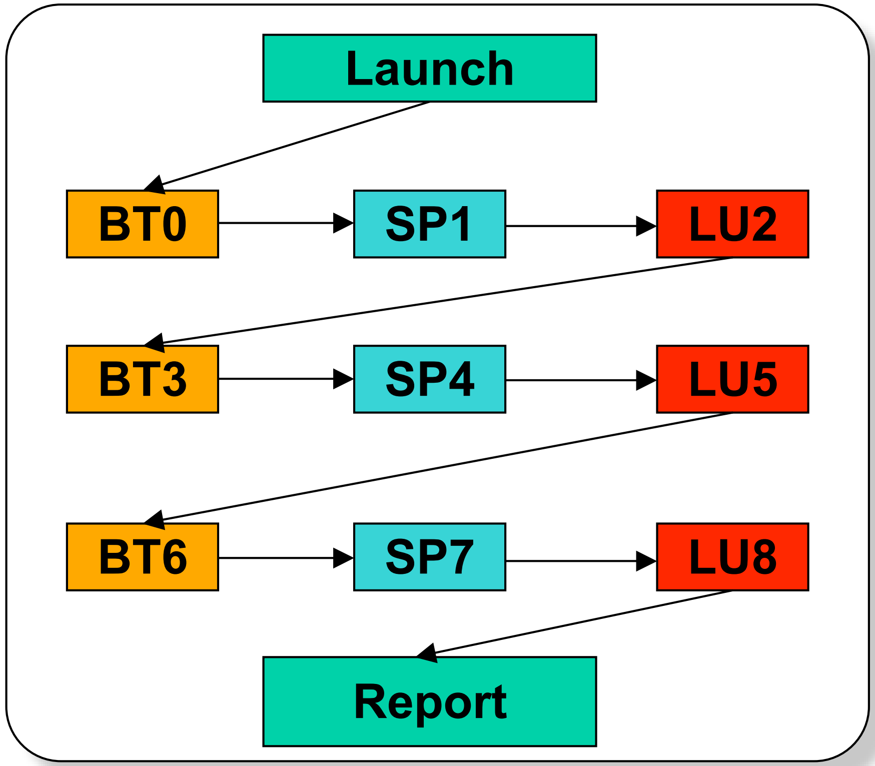
Experiences (5/5)



The NAS Grid Benchmarks (1/4)

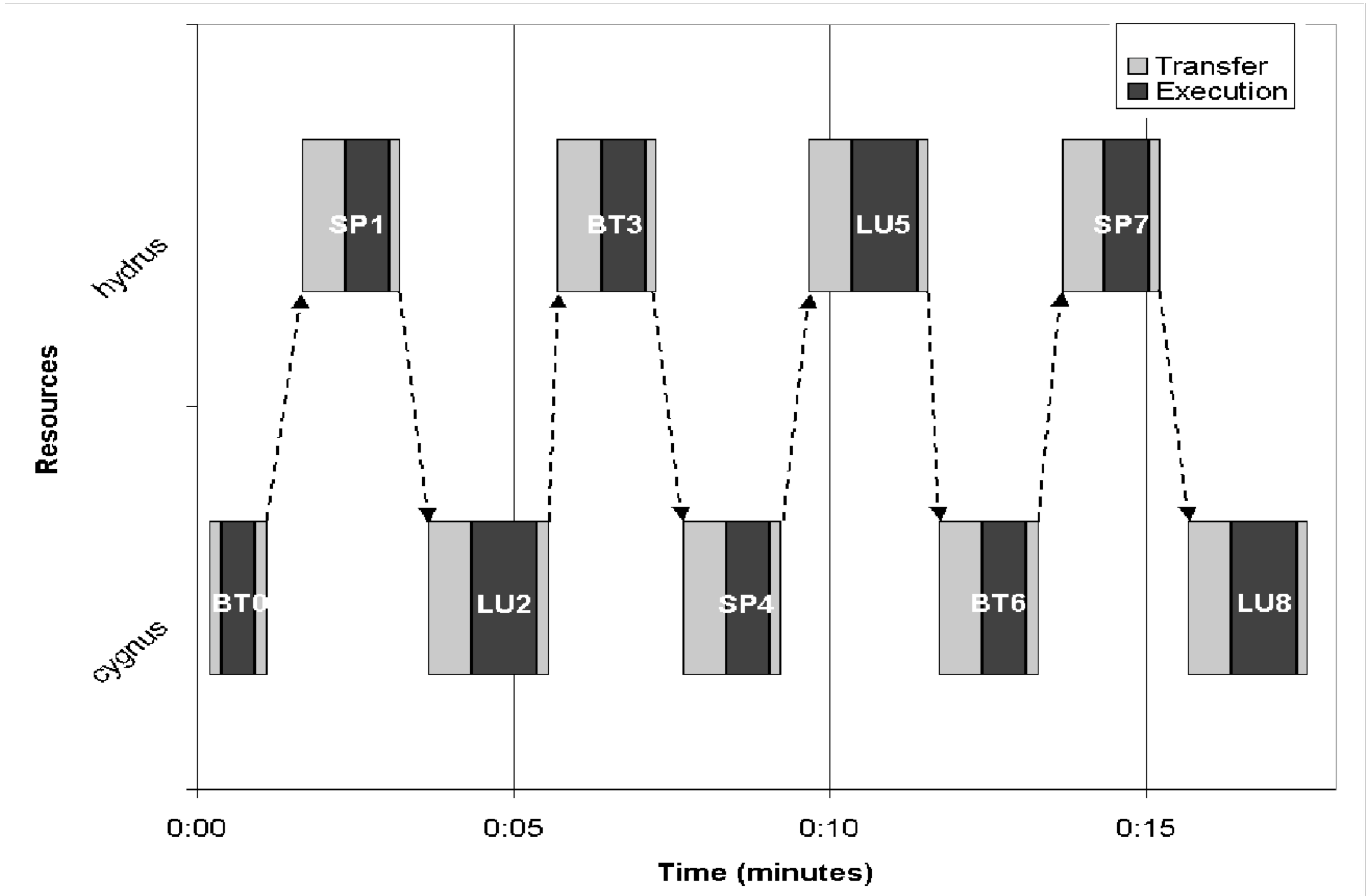
Helical Chain

- Represents long chains of repeating processes, such as a set of flow computations that are executed one after the other, as is customary when breaking up long running simulations into series of task, or in computational pipelines.



```
// Initialization
jobs[0].jt = bt;
jobs[1].jt = sp;
jobs[2].jt = lu;
jobs[3].jt = bt;
jobs[4].jt = sp;
jobs[5].jt = lu;
jobs[6].jt = bt;
jobs[7].jt = sp;
jobs[8].jt = lu;
drmaa_init(contact, err);
// Submit all jobs consecutively
for (i = 0; i<9; i++) {
drmaa_run_job(job_id, jobs[i].jt,
err);
drmaa_wait(job_id, &stat, timeout,
rusage, err);
}
drmaa_exit(err_diag);
```

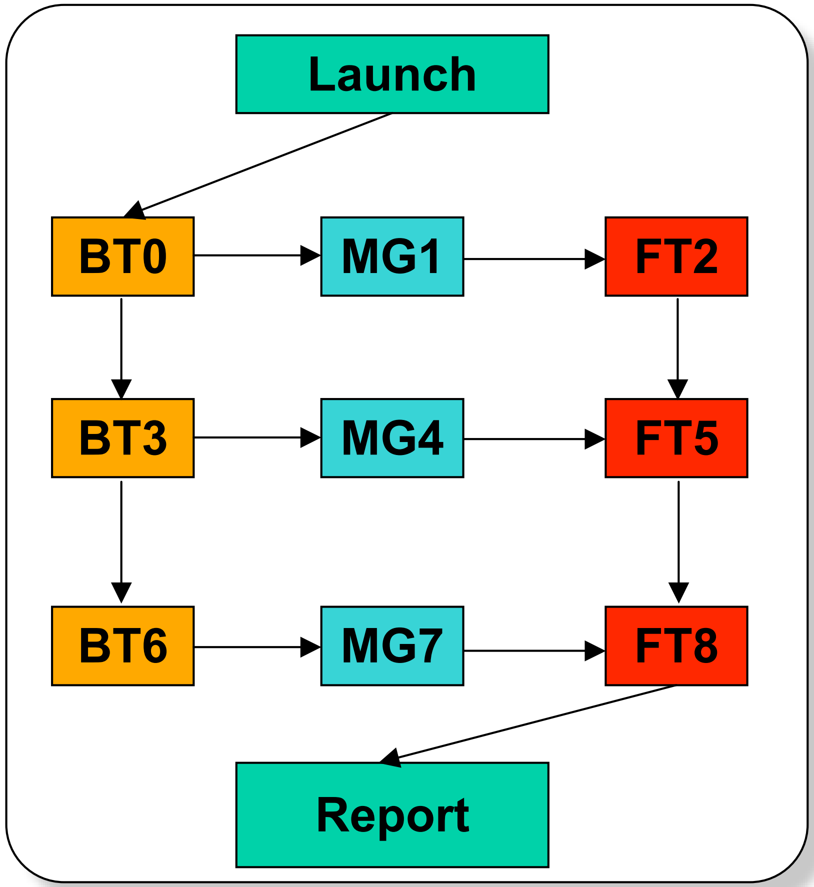
The NAS Grid Benchmarks (2/4)



The NAS Grid Benchmarks (3/4)

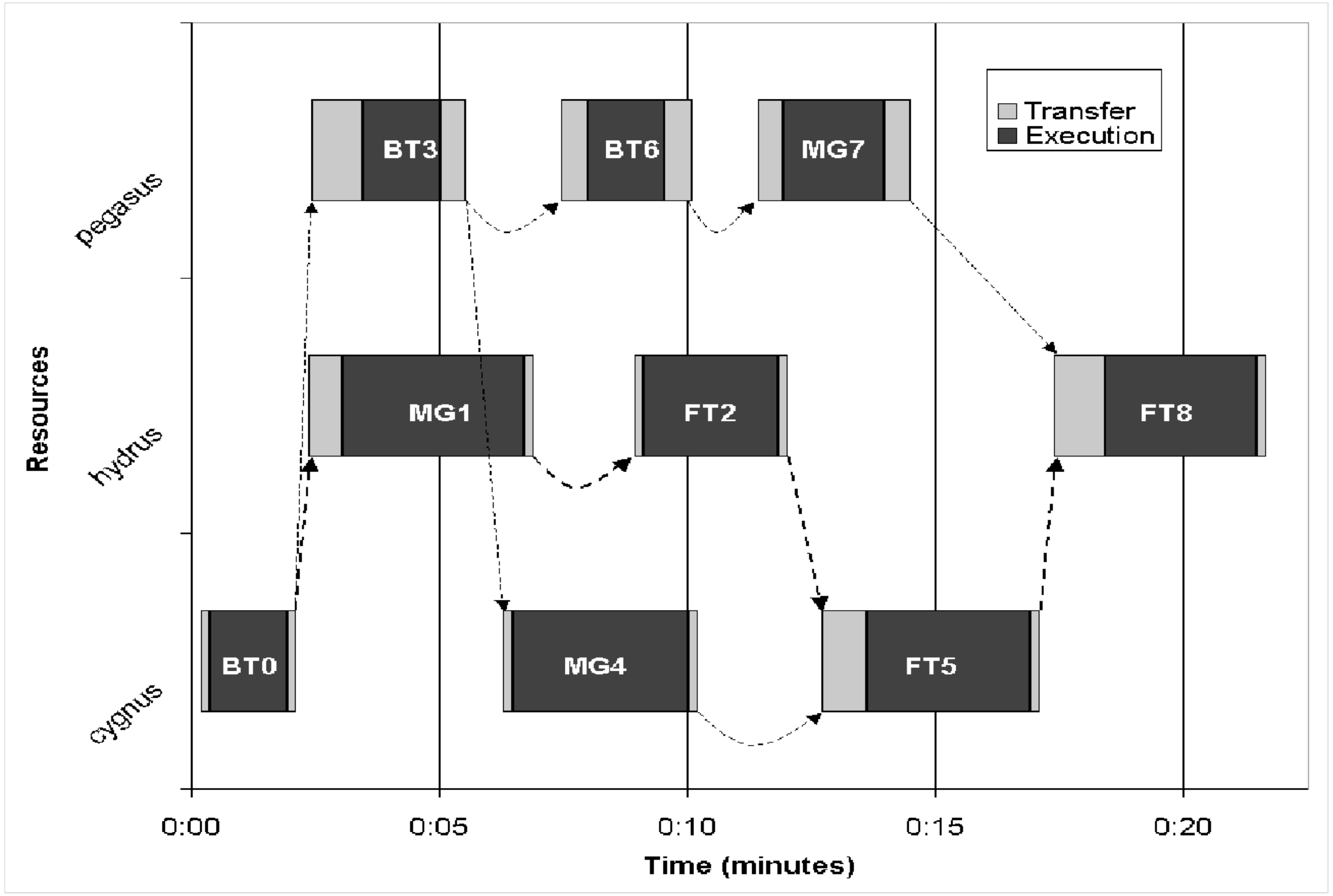
Visualization Pipe

- Represents chain of compound processes, like those encountered when visualizing flow solutions as the simulation progresses.



```
// Initialization
jobs[0].jt = bt; jobs[0].dep = "";
jobs[1].jt = mg; jobs[1].dep = "0";
jobs[2].jt = ft; jobs[2].dep = "1";
jobs[3].jt = bt; jobs[3].dep = "0";
jobs[4].jt = mg; jobs[4].dep = "3";
jobs[5].jt = ft; jobs[5].dep = "2 4";
jobs[6].jt = bt; jobs[6].dep = "3";
jobs[7].jt = mg; jobs[7].dep = "6";
jobs[8].jt = ft; jobs[8].dep = "5 7";
// Loop until all jobs are finished
while (there_are_jobs_left(jobs)) {
// Submit jobs with dependencies solved
for (i = 0; i<num_jobs; i++)
if (is_job_ready(jobs, i))
drmaa_run_job(jobs[i].id, jobs[i].jt, err);
// Wait any submitted job to finish
job_id = "DRMAA_JOB_IDS_SESSION_ANY";
drmaa_wait(job_id, &stat, timeout, rusage,
err);
set_job_done(jobs, job_id);}
```

The NAS Grid Benchmarks (4/4)



Conclusions

DRMAA

- We have presented the implementation of DRMAA on top of the Grid*W*ay framework and Globus.
- DRMAA aid the rapid development and distribution across the Grid of typical scientific applications.
- The execution of typical scientific applications, demonstrates the functionality, robustness and efficiency of this environment.
- The use of standard interfaces allows the comparison between different Grid implementations.
- DRMAA would help users, making Grid applications portable across DRMS adhered to the standard.