Experiences on Adaptive Grid Scheduling of Parameter Sweep Applications

Eduardo Huedo Cuesta (huedoce@inta.es)
Rubén Santiago Montero
Ignacio Martín Llorente

Laboratorio de Computación Avanzada, Simulación y Aplicaciones Telemáticas
Centro de Astrobiología
CSIC – INTA

Grupo de Arquitectura de Sistemas Distribuidos y Seguridad
Dpto. de Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Introduction

Parameter sweep applications (**PSA**) comprise the execution of a high number of tasks, each of which performs a given calculation over a subset of parameter values.

Reliable and **efficient scheduling and execution** of PSAs is challenging because of the **dynamic** and **heterogeneous** nature of Grids.

In this work, we present a **scheduling algorithm** that combines:

- **Adaptive scheduling** to reflect the dynamic Grid characteristics
- **Adaptive execution** to migrate running jobs to *better* resources and provide fault tolerance
- **Re-use of common files** between tasks to reduce the file transfer overhead

We will show the benefits of this **scheduling algorithm** in the execution of an existing CFD application to provide both:

- **fault tolerance** and
- **performance improvement**

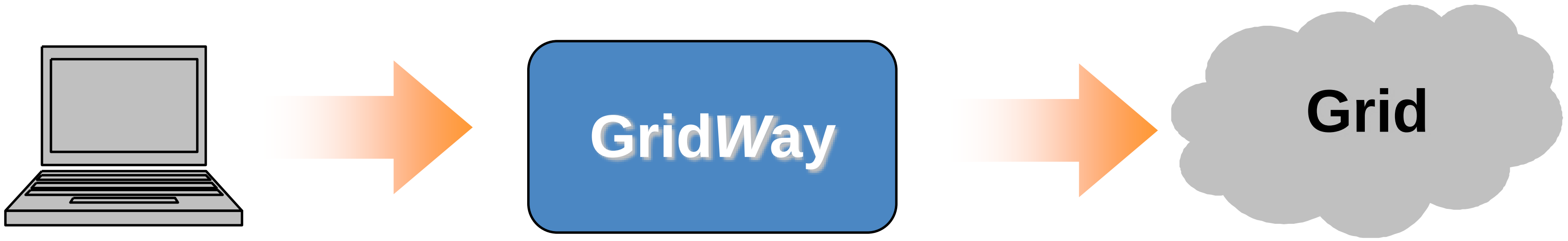
using the **Grid*Way*** tool.

Grid Scheduling

Scheduling steps:

•Where does it execute?	**Resource selection**
•What does it need?	**Job preparation**
•How does it start?	**Job submission**
•How is it performing?	**Job monitoring**
•Could it perform better?	**Job migration**
•What does it produce?	**Job termination**

The **GridWay** tool provides an **easier** and more **efficient** execution (***submit & forget***) on **heterogeneous** and **dynamic** Grids



Adaptive Job Execution

Execution steps:

- ***Prolog*** prepares the remote system and stages the input files.
- ***Wrapper*** executes the actual job and obtains its exit code.
- ***Epilog*** stages the output files and cleans up the remote system.

Re-use of common files

Adaptation to changing conditions is achieved through **dynamic rescheduling** of jobs:

- Periodically, to discover ***better resources***
- When a resource or its network connection **fail**
- When the job is **cancelled**
- When the job remains **suspended** too much time
- When a **performance degradation** is detected
- When the application **demands** change

$\left.\right\}$ **Grid** (first four items)

$\left.\right\}$ **Application** (last two items)

Job rescheduling can lead to its **migration** to a more suitable resource.

Experimental Testbed: UCM-CAB

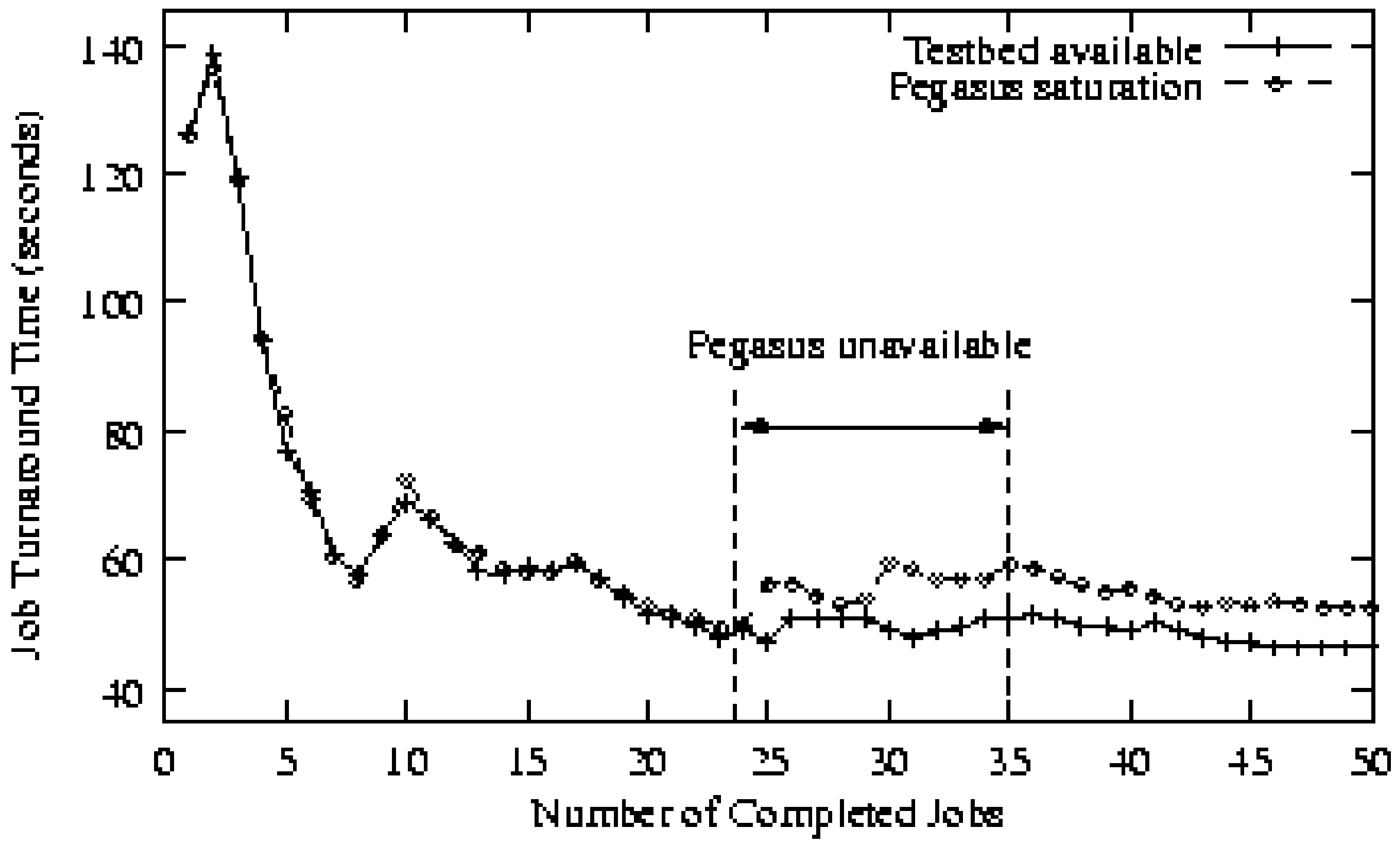
Testbed description:

Host	Nodes	Speed	OS	Memory	VO
ursa	1 x Sun UltrsaSPARC IIe	500MHz	Solaris 8	256MB	UCM
draco	1 x Sun UltrsaSPARC I	167MHz	Solaris 8	128MB	UCM
pegasus	1 x Intel Pentium 4	2.4MHz	Linux 2.4	1GB	UCM
solea	2 x Sun UltraSPARC II	296MHz	Solaris 8	256MB	UCM
babieca	4 x Alpha EV6	466MHz	Linux 2.2	1GB	CAB

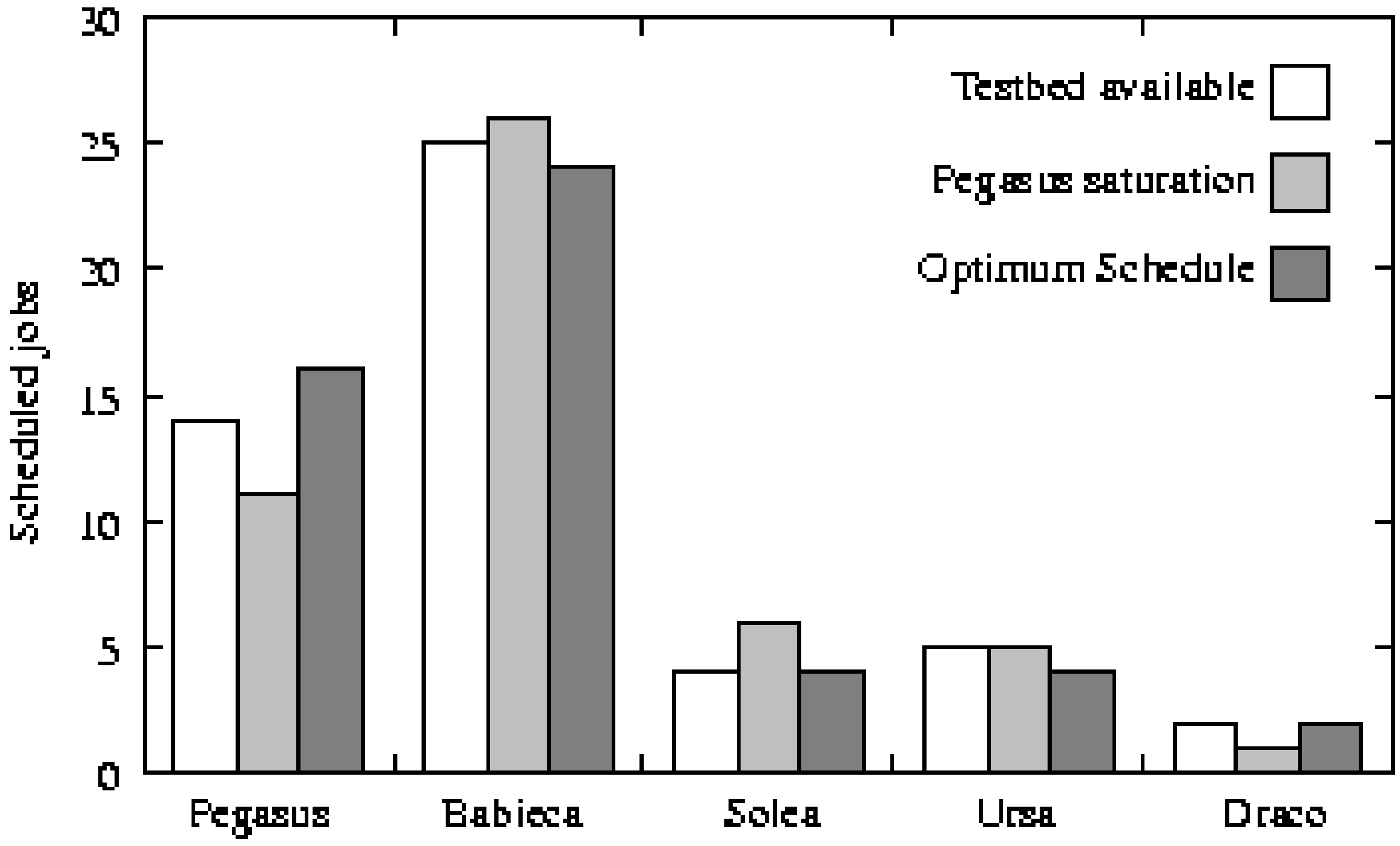
Experiment:

- CFD application to calculate the flow over a flat plate for a range of Reynolds numbers from 10^2 to 10^4 (50 tasks).

Results


Testbed available
Pegasus saturation
Pegasus unavailable
Job Turnaround Time (seconds)
Number of Completed Jobs
140
120
100
80
60
40
0
5
10
15
20
25
30
35
40
45
50

Results



Scheduled jobs
0
5
10
15
20
25
30
Testbed available
Pegasus saturation
Optimum Schedule
Pegasus
Babieca
Solea
Ursa
Draco

Conclusions and Future Work

We have extended the capabilities of the **Grid*W*ay** framework to enhance the **performance** and **reliability** of PSAs.

Grid*W*ay achieves the robust and efficient execution of PSAs by combining: **adaptive scheduling**, **adaptive execution** and **re-use of common files**.

The experimental results are promising because they show how generic PSAs can efficiently harness the highly distributed computing resources provided by a Grid.

We have also applied this techniques to a **bioinformatics application** to study large number of proteins sequences.