Loosely-Coupled Loop Scheduling in Computational Grids

José Herrera, Eduardo Huedo,
Rubén S. Montero and Ignacio M. Llorente

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

GridWay

Distributed Systems Architecture Group
Universidad Complutense de Madrid

Outline

- Motivation
- Self-Scheduling Schemes
- The GridWay Framework
- DRMAA: Distributed Resource Management Application API
- Development Model
- Loosely-Coupled Loop Scheduler
- A Simple Example
- Experimental Results
- Conclusions

Motivation

- **Loop distribution** is one of the most useful techniques to **reduce** the **execution time** of **parallel** applications.
- **MPICH-G2** have been used to **develop** the **self-scheduling** loops application in a **Grid** environment.
- Disadvantages:
  - All resources must be **allocated** to **begin execution** of the application.
  - It is necessary to **restart** the self-scheduling loop when a **resource fails**.
  - It is **no** possible to **join** new **resources** to a **running application**.
- **Our investigation**: A new approach to implement loop distribution in **Grid** using **DRMAA** API and **Grid*Way*** meta-scheduling framework.
- The **efficiency** and **reliability** of before schema to solve the **Mandelbrot** set problem is analyzed in a research testbed bases on the **Globus Toolkit 4.0.**

Self-Scheduling Schemes (1/2)

Introduction

- Two kinds of loop schedulers:
    - **Static**: The loop scheduling decision is made at compile-time.
    - **Dynamic**: The decision is made at execution-time.

- Two kinds of dynamic loop schedulers;
    - **Simple**: Also named self-scheduling schemes.
    - **Distributed**: The speed of cluster computers, the actual load of each node, etc.

- **Master-Worker paradigm**: The master node dynamically assigns tasks to the rest to the worker nodes. When a worker node ends, send the results to the master node.

- The different ways to compute the iterations assigned to each processor has given rise to different kinds of self-scheduling algorithms.

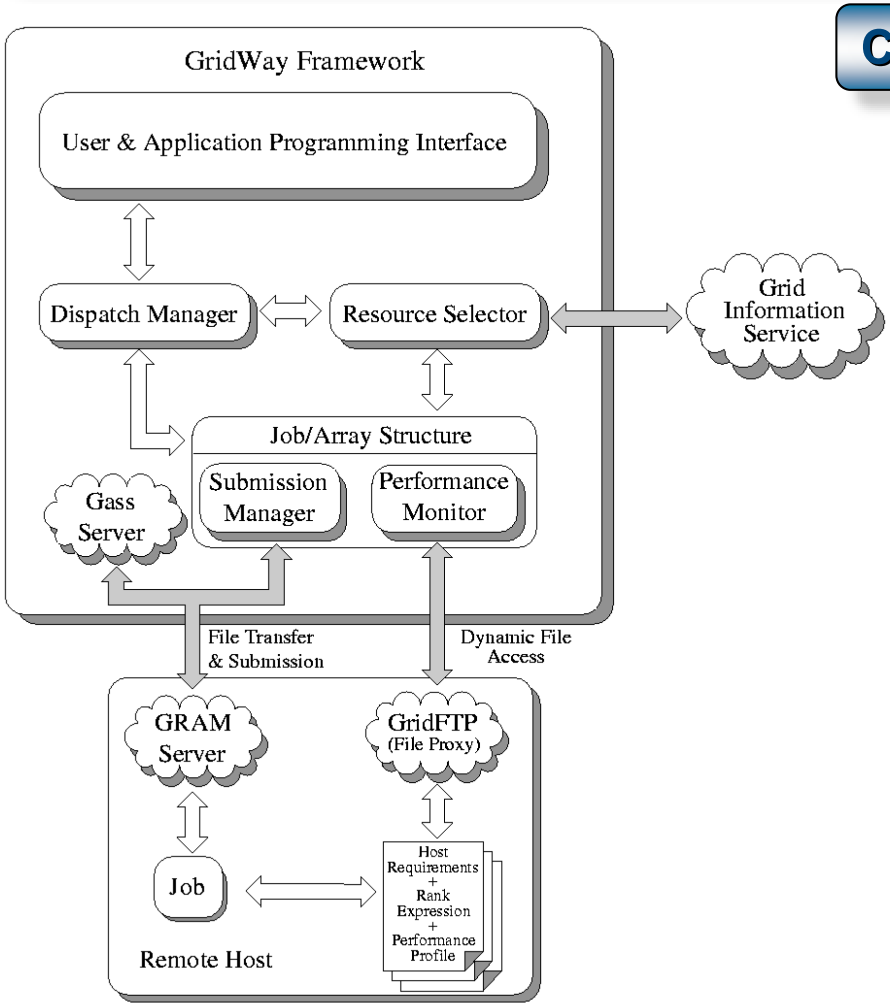
Self-Scheduling Schemes (2/2)

Kinds of Algorithms

- **Chunk Self-Scheduling(CSS).**
  - The chunk-size is fixed and is chosen by the user. When the chunk size is 1 it is named pure self-scheduling.
- **Guided Self-Scheduling (GSS).**
  - The chunk-size is decreasing. The user can choose the minimum chunk-size assigned to each processor.
- **Trapezoid Self-Scheduling (TSS).**
  - The chunk-size is linearly decreased a given amount.
- **Fixed Increase Self-Scheduling (FISS).**
  - During each phase, only a subset of the remaining loop iterations divided equally among the available processors. In each phase the chunk-size is linearly increased.

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

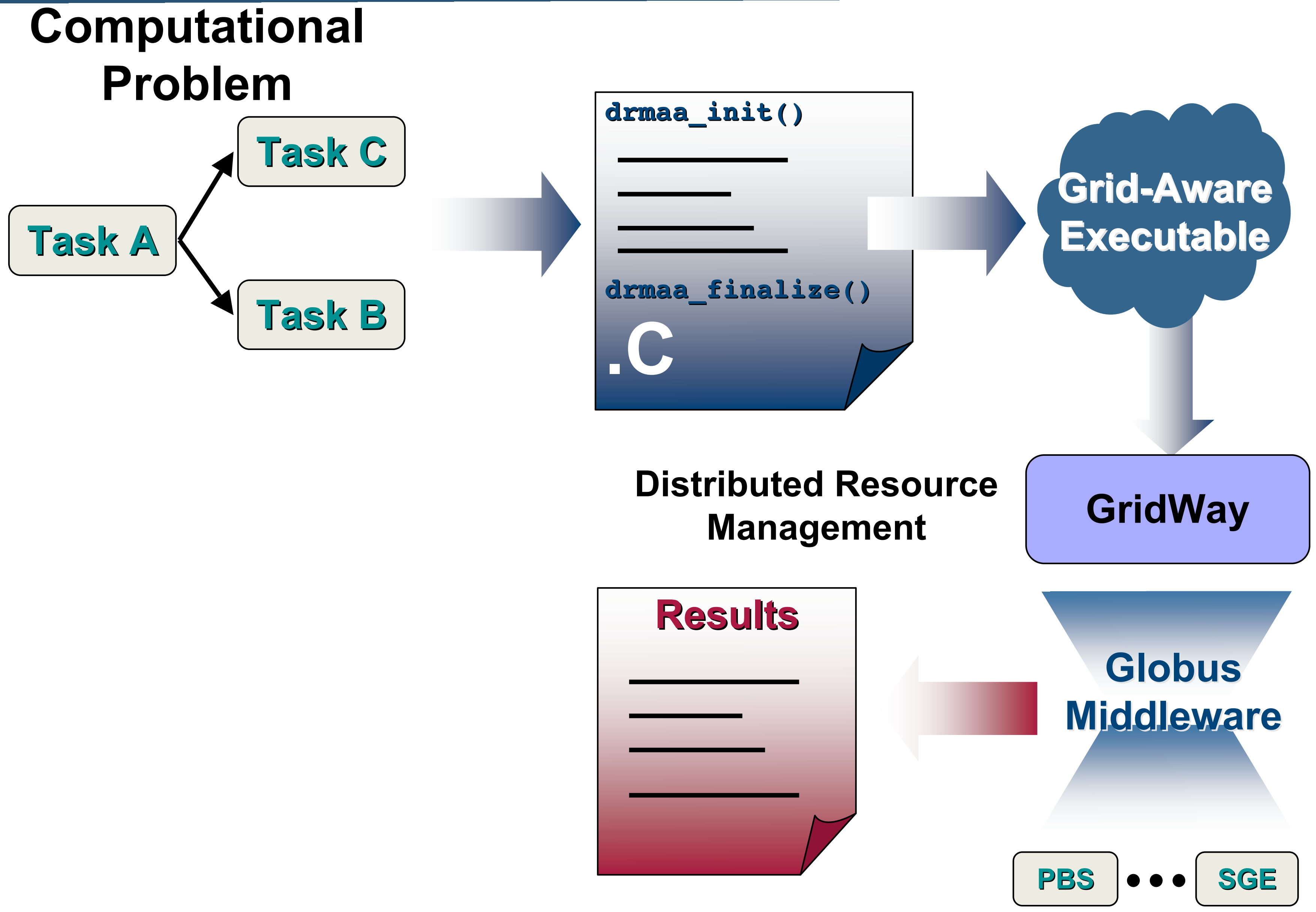
- The DRMAA specification constitutes a **homogenous interface** to different **DRMS** to handle job submission, monitoring and control, and retrieval of finished job status. Moreover, DRMAA has been developed by DRMAA-WG within the Global Grid Forum (GGF).

- 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



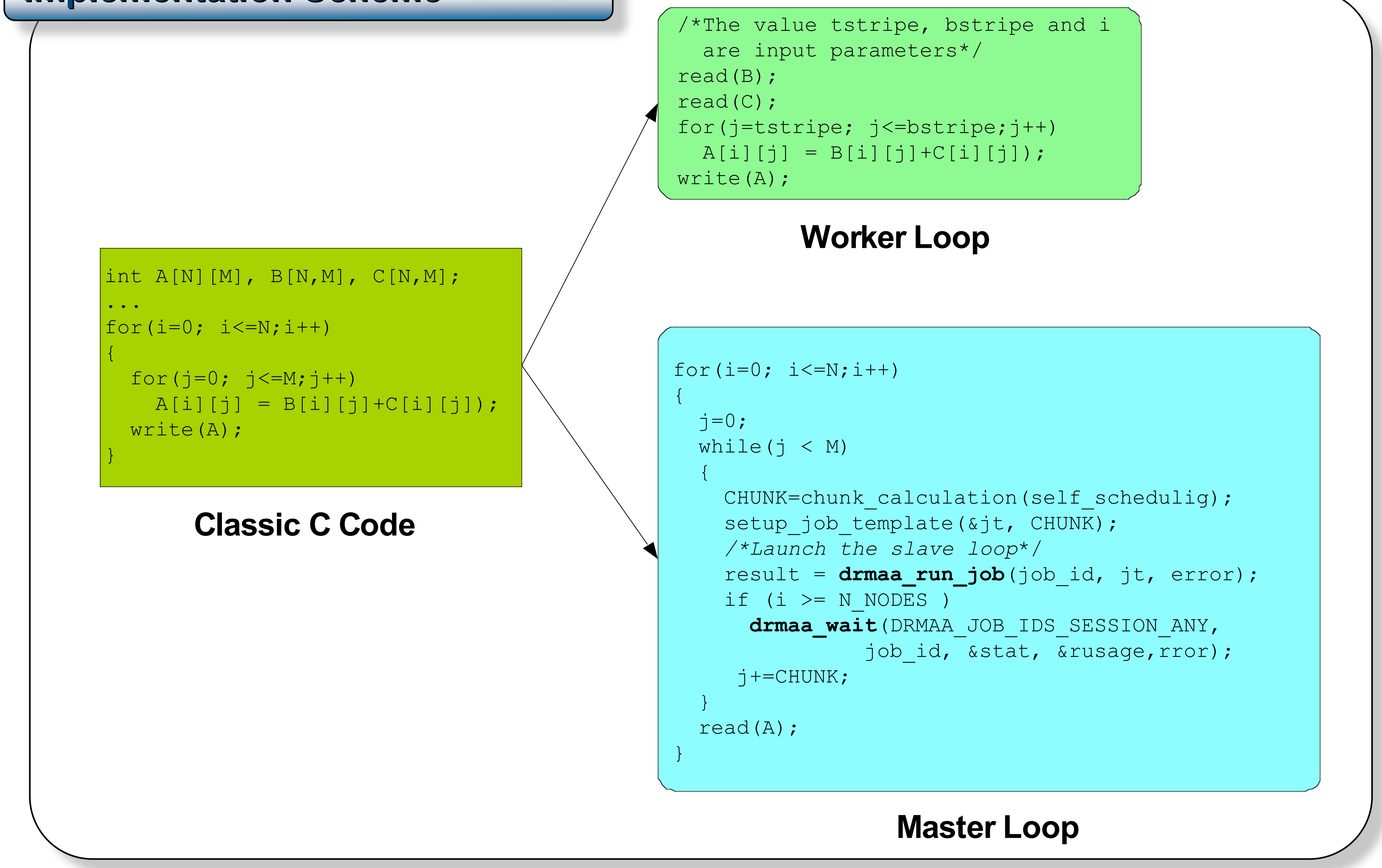
Loosely-Coupled Loop Scheduler

Advantages and Disadvantages

- **Main characteristics** of a loosely-coupled approach:
  - **Reliability**: When a resource fails the execution of the whole application continues.
  - **Dynamic Adaptation**: The worker loops can migrate to more suitable resources. New resources can be used to execute the remainder worker loops.
  - **Transparency**: The worker loop execution, fault tolerance and migration are transparent from the developer point of view.
  - **Deployment**: Resources exploitation GT4.0 pre-WS, GT4.0 WS and EGEE. It allows the drivers creation to other infrastructures.
- Main disadvantage ➔ The need for storing partial results in secondary storage.

A Example: Addition of 2-D Matrix

Implementation Scheme



```
int A[N][M], B[N,M], C[N,M];
...
for(i=0; i<=N;i++)
{
  for(j=0; j<=M;j++)
    A[i][j] = B[i][j]+C[i][j]);
  write(A);
}
```

Classic C Code

```
/*The value tstripe, bstripe and i
  are input parameters*/
read(B);
read(C);
for(j=tstripe; j<=bstripe;j++)
  A[i][j] = B[i][j]+C[i][j]);
write(A);
```

Worker Loop

```
for(i=0; i<=N;i++)
{
  j=0;
  while(j < M)
  {
    CHUNK=chunk_calculation(self_schedulig);
    setup_job_template(&jt, CHUNK);
    /*Launch the slave loop*/
    result = drmaa_run_job(job_id, jt, error);
    if (i >= N_NODES )
      drmaa_wait(DRMAA_JOB_IDS_SESSION_ANY,
                 job_id, &stat, &rusage,rror);
     j+=CHUNK;
  }
  read(A);
}
```

Master Loop

Experiences (1/3)

TestBed Description

Host	Model	Hz	OS	Memory	Nodes	GRAM
hydrus	Intel Pentium 4	3.2 Ghz	Linux 2.6	512MB	4	PBS
ursa	Intel Pentium 4	3.2 Ghz	Linux 2.6	512MB	1	fork
draco	Intel Pentium 4	3.2 Ghz	Linux 2.6	512MB	1	fork
cygnus	Intel Pentium 4	2.5 Ghz	Linux 2.6	512MB	1	fork

Objectives

- We evaluate the functionality and efficiency of the loosely-coupled loop scheduling in a computational Grid.
- We consider the simple self-scheduling schemes to distribute the Mandelbrot set application on a slightly heterogeneous testbed based on the Globus Toolkit.

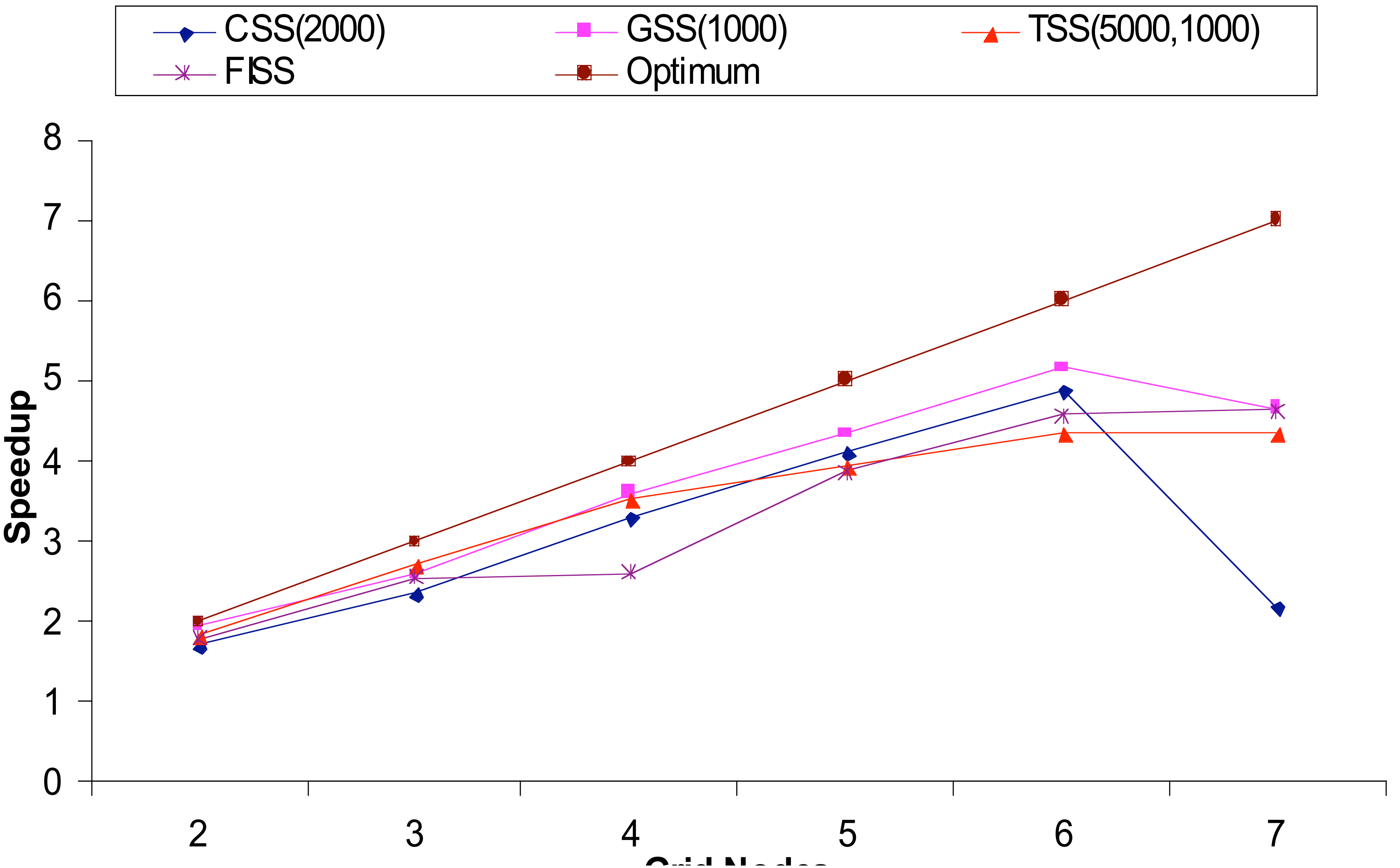
Experiences (2/3)

Experiment Description

- We consider an application that solves the Mandelbrot set problem for a windows size **60000x50000** pixels with 6 bits per pixel (2.1 Gigabytes).
- Domain, [-1.7, 0.8] x [-1.0, 1.0]
- Size of each stripe: **60000xchunk**
- Example with 5 nodes:

Scheme	Chunk size
CSS(2000)	2000 2000 2000 2000 2000 2000 2000 …
GSS(1000)	10000 8000 6400 5120 4096 3277 2622 …
TSS(5000, 1000)	5000 4750 4500 4250 4000 3750 3500 …
FISS	2000 2000 2000 2000 2000 3334 3334 …

Experiences (3/3)



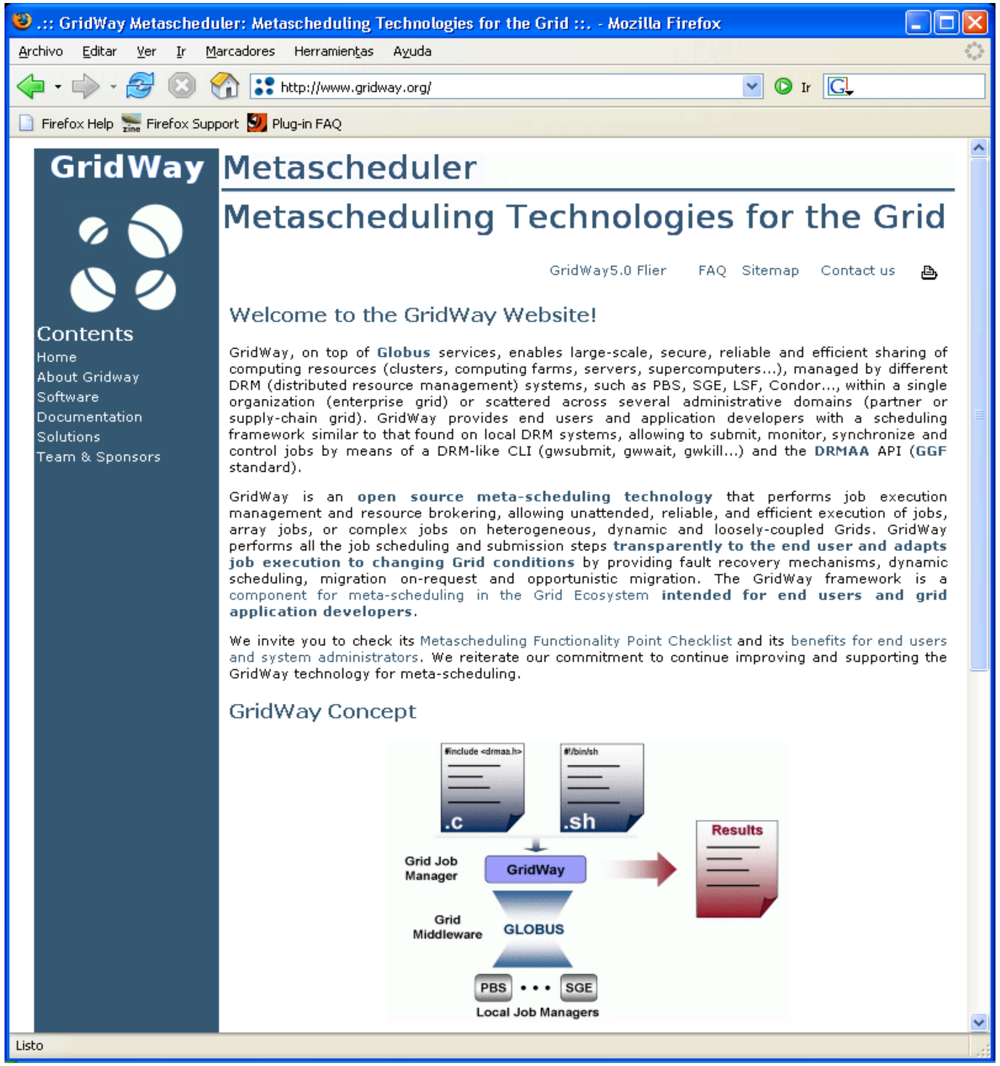
Conclusions

- We have presented the implementation of a set of self-scheduling algorithms using **Grid** technology (**DRMAA API** and **Grid*W*ay** framework).
- We compare this approach with MPICH-G2 applications.
- Main advantages: Reliability, Dynamic Adaptation, Transparency and Deployment.
- We have been demonstrated the functionality and efficiency of this approach with the calculation of Mandelbrot set problem.
- We demonstrated how Grid characteristics can degrade the execution time of the dynamic scheduling algorithms.

GridWay Information

Information and download in http://www.GridWay.org

Apache Licence



.:: GridWay Metascheduler: Metascheduling Technologies for the Grid ::. - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Ayuda
http://www.gridway.org/
Ir
Firefox Help
Firefox Support
Plug-in FAQ
Listo

GridWay Metascheduler

Metascheduling Technologies for the Grid

GridWay5.0 Flier FAQ Sitemap Contact us

Contents

- Home
- About Gridway
- Software
- Documentation
- Solutions
- Team & Sponsors

Welcome to the GridWay Website!

GridWay, on top of **Globus** services, enables large-scale, secure, reliable and efficient sharing of computing resources (clusters, computing farms, servers, supercomputers...), managed by different DRM (distributed resource management) systems, such as PBS, SGE, LSF, Condor..., within a single organization (enterprise grid) or scattered across several administrative domains (partner or supply-chain grid). GridWay provides end users and application developers with a scheduling framework similar to that found on local DRM systems, allowing to submit, monitor, synchronize and control jobs by means of a DRM-like CLI (gwsubmit, gwwait, gwkill...) and the **DRMAA** API (**GGF** standard).

GridWay is an **open source meta-scheduling technology** that performs job execution management and resource brokering, allowing unattended, reliable, and efficient execution of jobs, array jobs, or complex jobs on heterogeneous, dynamic and loosely-coupled Grids. GridWay performs all the job scheduling and submission steps **transparently to the end user and adapts job execution to changing Grid conditions** by providing fault recovery mechanisms, dynamic scheduling, migration on-request and opportunistic migration. The GridWay framework is a component for meta-scheduling in the Grid Ecosystem **intended for end users and grid application developers**.

We invite you to check its Metascheduling Functionality Point Checklist and its benefits for end users and system administrators. We reiterate our commitment to continue improving and supporting the GridWay technology for meta-scheduling.

GridWay Concept

Information about GridWay

Additional Information about GridWay

Grid Ecosystem in **Globus** site

Tutorial in **IBM** site

Solaris Instalation in **Sun Microsystems** site

"DRMAA" and "*Grid Scheduling Architecture*" WGs in **GGF**

Loosely-Coupled Loop Scheduling in Computational Grids

José Herrera, Eduardo Huedo,
Rubén S. Montero and Ignacio M. Llorente

CSIC

INTA

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

GridWay

Distributed Systems
Architecture Group
Universidad Complutense de Madrid