Available online at www.sciencedirect.com
SCIENCE DIRECT
Parallel Computing 32 (2006) 267–279
PARALLEL COMPUTING
www.elsevier.com/locate/parco

Benchmarking of high throughput computing applications on Grids ☆

R.S. Montero [a,*], E. Huedo [b], I.M. Llorente [a,b]

[a] Departamento de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain
[b] Laboratorio de Computación Avanzada, Simulación y Aplicaciones Telemáticas, Centro de Astrobiología (CSIC-INTA), 28850 Torrejón de Ardoz, Spain

Received 15 September 2004; received in revised form 15 June 2005; accepted 16 December 2005
Available online 24 January 2006

Abstract

Grids constitute a promising platform to execute loosely coupled, high-throughput parameter sweep applications, which arise naturally in many scientific and engineering fields like bio-informatics, computational fluid dynamics, particle physics, etc. In spite of the simple computational structure of these applications, its efficient execution and scheduling are challenging because of the dynamic and heterogeneous nature of Grids. In this work, we propose a benchmarking methodology to analyze the performance of computational Grids in the execution of high throughput computing applications, that combines: (i) a representative benchmark included in the NAS Grid Benchmark suite; (ii) a performance model that provides a way to parametrize and compare different Grids; and (iii) a set of application-level performance metrics to analyze and predict the performance of this kind of applications. The benchmarking methodology will be applied to the performance analysis of a Globus-based research testbed that spans heterogeneous resources in five institutions.

Keywords: Grid computing; Benchmarking; Performance modelling; Globus toolkit

1. Introduction

Grid computing has emerged as a promising computing platform that can support the execution of next generation scientific applications, and will arguably open up avenues in many research disciplines. However, the performance that these platforms can potentially deliver to applications remain poorly understood. In this way, to attain an efficient usage of current Grid infrastructures it is necessary to define a standard methodology for its evaluation and benchmarking.

☆ This research was supported by Ministerio de Educación y Ciencia, through the research grant TIC 2003-01321 and 2002-12422-E, and by Instituto Nacional de Técnica Aeroespacial "Esteban Terradas" (INTA) – Centro de Astrobiología.
* Corresponding author. Tel.: +34 91 394 75 38; fax: +34 91 394 75 27.
E-mail addresses: rubensm@dacya.ucm.es (R.S. Montero), huedoce@inta.es (E. Huedo), llorente@dacya.ucm.es (I.M. Llorente).

0167-8191/$ - see front matter
doi:10.1016/j.parco.2005.12.001

Benchmarking is a widely accepted method to evaluate the performance of computer architectures. Traditionally, benchmarking of computing platforms has been successfully performed through low level probes that measure the performance of specific aspects of the system when performing basic operations, e.g. LAPACK [1], as well as representative applications of the typical workload, e.g. SPEC [2] or NAS Parallel Benchmarks [3]. In this sense, benchmarking has been proved helpful for investigating the performance properties of a given system, predicting its performance in the execution of a category of applications, and for comparing different systems.

The critical difference between a Grid and traditional HPC systems is that in the later a single software layer provides a centralized and single vision of all the resources. On the other hand, the capabilities of a Computational Grid, made up of heterogeneous components with dynamic performance, are usually provided by three layers: Grid fabric (e.g. worker nodes, operating systems or local resource managers) interconnected by a heterogeneous network, basic Grid services (e.g. job submission, data management or information services) and high level tools and Grid services (e.g. resource brokers, meta-schedulers or portals).

Grid benchmarks can be also grouped in the two aforementioned categories: low level probes that provide information of specific aspects of system's performance; and benchmarks that are representative of a class of applications. In this first category, the Network Weather Service [4] provides accurate forecast of dynamically changing performance characteristics from a distributed set of computing resources. Also Chun et al. [5] have proposed a set of benchmark probes for Grid assessment. These probes exercise basic Grid operations with the goal of measuring the performance and the performance variability of basic Grid operations, as well as the failure rates of these operations. Finally, the GridBench tool [6] developed in the CrossGrid project, is a benchmark suite for characterizing individual Grid nodes and collections of Grid resources. GridBench includes micro-benchmarks and application kernels to measure computational power, inter-process communication bandwidth, and I/O performance.

On the other hand, the NAS Grid benchmarks (NGB) have been recently proposed [7] based on the well-known NAS Parallel Benchmarks. This benchmark suite has been previously considered by Peng et al. [8] to evaluate the performance of a cluster Grid with Sun Grid Engine and Globus, and by its designers, Van der Wijngaart and Frumkin [9], to measure and improve the performance of NASA's Information Power Grid.

In this work, we propose a benchmarking methodology and a set of Grid-specific metrics to analyze the performance of computational Grid infrastructures in the execution of High Throughput Computing (HTC) applications. We will use the embarrassingly distributed benchmark of the NGB suite to represent the execution profile of these applications. The benchmarking process proposed here provides a way to investigate performance properties of Grid environments, to predict the performance of this category of applications, and to compare different platforms by inserting performance models in the benchmarking process.

The rest of this paper is organized as follows: In Section 2, we present the evaluation metrics used in this work, and the main characteristics of the NGB suite. The Grid environment used in this research is then introduced in Section 3, along with the Grid*W*ay framework used to execute the benchmarks, and the implementation of the NGB. In Section 4, we discuss how NGB can be a valuable tool to evaluate the performance of a Grid infrastructure. Finally, Section 5 presents a discussion of our results.

2. Grid benchmarking

In general, we can identify three aspects that should be assessed when evaluating a Grid environment, namely: functionality, reliability and performance. The use of a Grid requires a high level of expertise and a good understanding of all its components and their interaction. In this way, an user must manually perform to some extent, all the submission stages involved in the execution of an application (e.g. resource selection, resource preparation or job monitoring). Therefore, Grid benchmarks should verify a basic functionality of the environment. On the other hand, Grids are difficult to efficiently harness due to their heterogeneous nature and the unpredictable changing conditions they present. A suitable methodology for Grid benchmarking should help to identify and quantify failure situations, the capability to recover from them, and so to determine the reliability of a Grid.

In this work we concentrate on the development of a benchmarking methodology to assess the performance offered by a Grid environment in the execution of HTC applications. In this section we will first characterize

the execution of this kind of applications on a Grid, and then we propose a set of application-level performance metrics to analyze and predict their performance.

2.1. Workload characterization

Let us consider a High Throughput Computing (HTC) application that comprises the execution of a set of independent tasks, each of which performs the same calculation over a subset of parameter values. Note that given the heterogeneous nature of a Grid, the execution time of each task can differ greatly. So the following analysis is valid for general HTC applications, where each task may require distinct instruction streams.

In the execution of this kind of applications, a Grid can be considered, from the computational point of view, as an array of *heterogeneous* processors. Therefore, the number of tasks completed as a function of time is given by

$$n(t) = \sum_{i \in G} N_i \left\lfloor \frac{t}{T_i} \right\rfloor, \tag{1}$$

where N_i is the number of processors in the Grid (G) that can compute a task in T_i seconds, including file transfer and middleware overheads. The characterization of a Grid by the above equation is not obvious, since it is a discontinuous function that can potentially involve a high number of terms.

The best characterization of the Grid can be obtained if we take the line that represents the average behavior of the system, see Fig. 1. This approach has been previously proposed by Hockney and Jesshope [10] to characterize the performance of homogeneous array architectures on vector computations. So, a first-order description of a Grid can be made by

$$n(t) = mt + b. \tag{2}$$

Using the r_∞ and $n_{1/2}$ parameters defined by Hockney and Jesshope, this formula can be rewritten as:

$$n(t) = r_\infty t - n_{1/2} \quad \text{with } m = r_\infty \text{ and } b = -n_{1/2}. \tag{3}$$

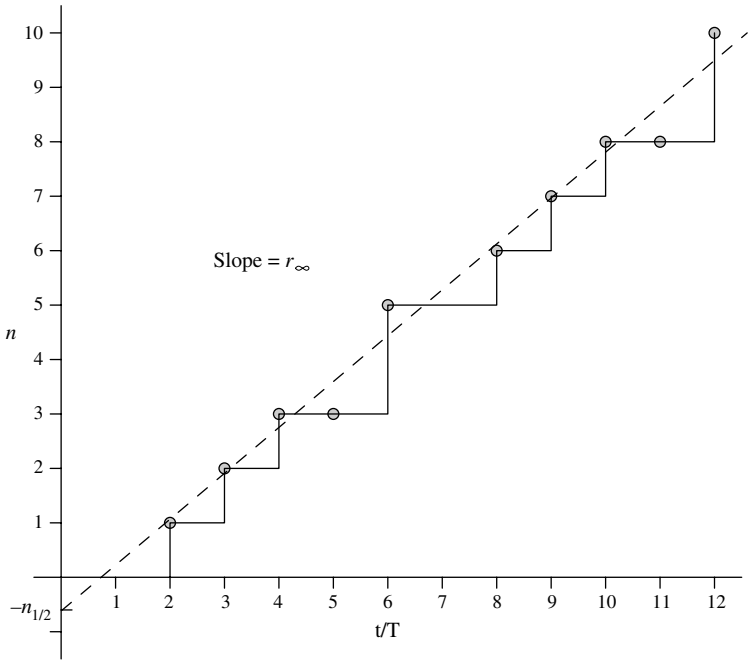

Fig. 1. The number of tasks n as a function of time t on an heterogeneous array of two processors with execution times $2T$ and $3T$ (full line); and linear approximation of the array (broken line).

These parameters are called [10]:

- Asymptotic performance (r_∞): the maximum rate of performance in tasks executed per second. In the case of an homogeneous array of N processors with an execution time per task T, we have $r_\infty = N/T$.
- Half-performance length ($n_{1/2}$): the number of task required to obtain the half of the asymptotic performance. This parameter is also a measure of the amount of parallelism in the system as seen by the application. In the homogeneous case, for an embarrassingly distributed application we obtain $n_{1/2} = N/2$.

The approximation (3) can be interpreted as an idealized representation of a Grid, equivalent to an homogeneous array of $2n_{1/2}$ processors with an execution time per task $2n_{1/2}/r_\infty$.

We have considered so far the maximum performance (r_∞) of a Grid in the execution of a set of independent tasks. The linear relation (3) can be used to define the performance of the system (tasks completed per second) on actual applications with a finite number of tasks [10]:

$$r(n) = n(t)/t = \frac{r_\infty}{1 + n_{1/2}/n}. \tag{4}$$

The half-performance length ($n_{1/2}$), on the other hand, provides a quantitative measure of the heterogeneity in a Grid. This result can be understood as follows, faster processors contribute in a higher degree to the performance obtained by the system. Therefore the apparent number of processors ($2n_{1/2}$), from the application's point of view, will be in general lower than the total processors in the Grid (N). We can define the degree of heterogeneity (v) as

$$v = \frac{2n_{1/2}}{N}. \tag{5}$$

This parameter varies form $v = 1$ in the homogeneous case, to $v \approx 0$ when the actual number of processors in the Grid is much greater than the apparent number of processors (highly heterogeneous).

The $n_{1/2}$ is an useful characterization parameter for Grid infrastructures in the execution of HTC applications. For example, let us consider two different Grids with a similar asymptotic performance. In this case, by analogy with the homogeneous array, a lower $n_{1/2}$ parameter reflects a better performance (in terms of wall time) per Grid resource, since the same performance (in terms of throughput) is delivered by a smaller "number of processors".

In this work, we will use the *Embarrassingly Distributed* (ED) benchmark of the *NAS Grid Benchmarks* [7,11] (NGB) suite to represent the execution profile of the applications described above. The NGBs have been presented as data flow graphs, encapsulating an instance of a *NAS Parallel Benchmarks* (NPB) [12] code in each graph node, which *communicates* with other nodes by sending/receiving initialization data. Like NPB, NGB specifies several different classes or problem sizes, in terms of number of tasks, mesh size and number of iterations (see [7] for a detailed description).

The ED benchmark represents the important class of *Parameter Sweep Applications* (PSA), which arise naturally in many scientific and engineering fields like Bio-informatics, Computational Fluid Dynamics (CFD), Particle Physics, etc. Grid infrastructures constitute a promising platform to execute these loosely-coupled high-throughput PSAs. However, in spite of its relatively simple structure, the efficient execution of PSAs involves challenging issues. In general, the efficient execution of this kind of applications will combine the following elements: adaptive scheduling and execution; and re-use of common files between tasks to reduce the file transfer overhead [13].

2.2. Performance metrics

A Grid is a system not subject to a centralized control, and based on standard, open and general-purpose interfaces and protocols, while providing some level of quality of service (QoS), in terms of security, throughput, response time or the coordinated use of different resource types [14]. The functionality of a Grid is usually provided by three layers: Grid fabric, basic Grid services and high level tools. In this sense, we will hereafter refer to a *Grid* (or *testbed*) as the set of hardware and software elements that conform it, including all the

elements in the previous layers. Therefore, we intend to measure the performance of the whole infrastructure rather than each component's performance, individually.

In the previous section we have characterized the performance of a Grid in the execution of HTC applications by the r_∞ and $n_{1/2}$ parameters. These parameters can be determined for a Grid in two ways:

- *Intrusive* benchmarking. The system parameters are obtained by linear fitting to the experimental results obtained in the execution of large-scale HTC applications. In order to empirically determine r_∞ and $n_{1/2}$ the benchmarking process should be intrusive to exercise all the resources in the testbed ($n \gg N$). Therefore, this methodology is not intended to be applied over long periods of time, but at carefully controlled and scheduled occasions.
- *Non-intrusive* benchmarking. In general, it may not be feasible to run such an intrusive high throughput benchmark for large Grid environments. In this situation, the r_∞ and $n_{1/2}$ parameters can be computed using Eq. (1) and raw performance data (average wall time per task, T_i) of each resource, which can be obtained in a systematic non-intrusive way.

The wall time of each task can be split into

$$T_i = T_i^{\text{xfr}} + T_i^{\text{exe}} + T^{\text{sch}}, \tag{6}$$

where T_i^{xfr} and T_i^{exe} are the average file transfer and execution times on host i, including middleware overheads and local queue times; and T^{sch} is the scheduling time which represents the resource selection overhead. In the following experiments we assume $T^{\text{sch}} = 60$ s (see discussion in Section 3.2). Combining Eqs. (1) and (6), we get

$$n(t) = \sum_{i \in G} N_i \left\lfloor \frac{t}{T_i^{\text{xfr}} + T_i^{\text{exe}} + T^{\text{sch}}} \right\rfloor. \tag{7}$$

This equation can be use to obtain the *non-intrusive* r_∞ and $n_{1/2}$ parameters by fitting the best straight line. We can also estimate the influence of the resource selection overhead by comparing the non-intrusive r_∞ and $n_{1/2}$, with those obtained by making ($T^{\text{sch}} = 0$) in Eq. (7).

Besides comparing different Grid environments by their performance, and the r_∞ and $n_{1/2}$ parameters, it is of crucial interest to analyze the potential gain in performance that a site could obtain by *joining* the Grid. In other words, a suitable way to estimate the difference in performance obtained by the application running on the resources available to the user, and that obtained by porting it to the Grid. To this end we propose the following application-level performance metrics:

- *Grid speed-up* (S_{site}). A site or institution is defined as the set of hardware and software resources within the same administration domain. The speed-up that an user can expect when she executes a given class of applications on the Grid can be defined, by analogy to parallel computers, as

$$S_{\text{site}} = \frac{T_{\text{site}}}{T_{\text{Grid}}}, \tag{8}$$

  where T_{Grid} is the application turnaround time using all the Grid, and T_{site} is the *optimum* execution time using only the resources available in a given *site*. For HTC applications T_{Grid} can be easily calculated with Eq. (3).
- *Critical problem size* (n_{site}). In the case of HTC applications, it is defined as the number of jobs from which it is more efficient to use the Grid rather than only the site resources. This parameter should be calculated for each site, and gives an estimation of the suitability of the Grid to execute a range of applications. It is interesting to note that in the case of an strictly homogeneous environment n_{site} is the number of nodes in the site.

3. The Grid environment

In this section we describe the Grid environment whose performance will subsequently be evaluated. This description includes both, the hardware and network configuration, and the Grid scheduler used to execute the benchmarks. Also, some details about the implementation of the ED benchmark are discussed.

3.1. Grid fabric and basic middleware

This work has been possible thanks to the collaboration of several research centers and universities that temporarily shared some of their resources in order to set up a heterogeneous and geographically distributed testbed. The testbed is build up from five sites, all of them connected by the Spanish *National Research and Education Network* (NREN), RedIRIS. The geographical distribution and interconnection network of the sites are shown on Fig. 2. This organization results in a highly heterogeneous testbed, since it presents several architectures, processor speeds, DRMS and network links.

NGB does not attempt to measure the performance of the underlying Grid hardware, but the functionality, reliability and performance of the Grid environment. However, a clear understanding of the hardware configuration of the Grid resources will aid the analysis of the subsequent experiments. Table 1 shows the characteristics of the machines in the research testbed, based on the Globus Toolkit 2.X [15].

3.2. High level tools: GridWay

The Globus Toolkit has become a de facto standard in Grid computing [15]. Globus services allow secure and transparent access to resources across multiple administrative domains, and serve as building blocks to

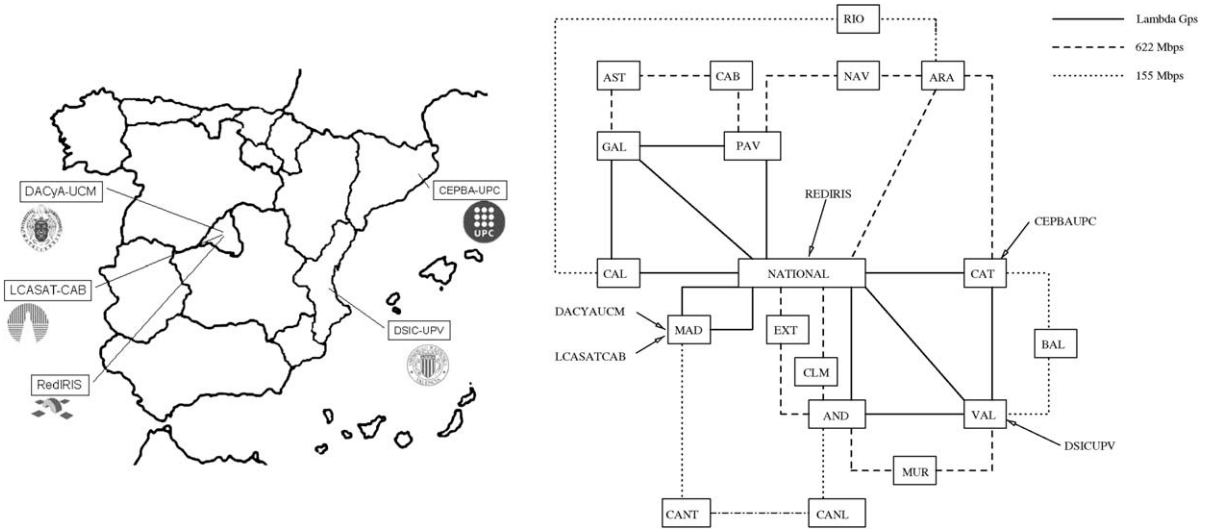

Fig. 2. Geographical distribution of the sites in Spain (left-hand graph) and topology of RedIRIS, the Spanish National Research and Education Network (NREN) (right-hand graph).

Table 1
Characteristics of the machines in the research testbed

Name	Site	Nodes	CPU	Speed	Memory per node	DRMS
hydrus	DACYA-UCM	1	Intel P4	2.5 GHz	512 MB	Fork
cygnus	DACYA-UCM	1	Intel P4	2.5 GHz	512 MB	Fork
cepheus	DACYA-UCM	1	Intel PIII	600 MHz	256 MB	Fork
aquila	DACYA-UCM	1	Intel PIII	700 MHz	128 MB	Fork
babieca	LCASAT-CAB	5	Alpha EV67	450 MHz	256 MB	PBS
platon	RedIRIS	1	2 × Intel PIII	1.4 GHz	1 GB	Fork
heraclito	RedIRIS	1	Intel Celeron	700 MHz	256 MB	Fork
ramses	DSIC-UPV	3	2 × Intel PIII	900 MHz	512 MB	PBS
khafre	CEPBA-UPC	1	4 × Intel PIII	700 MHz	2 GB	Fork

In this table only the computational resources devoted to the testbed are shown.

implement the stages of Grid scheduling [16]. However, the user is responsible for manually performing all the submission steps in order to achieve any functionality.

To overcome these limitations, we have developed a light-weight meta-scheduler that allows an easier and more efficient execution of jobs on a dynamic Grid environment in a "*submit & forget*" fashion. The core of the Grid*Way* framework [17] is a personal *submission agent* that performs all scheduling stages and watches over the correct and efficient execution of jobs on Globus-based Grids.

The brokering process of the Grid*Way* framework starts discovering available compute resources by accessing the Grid Index Information Service (GIIS) and, those resources that do not meet the user-provided requirements are filtered out. At this step, an authorization test is also performed on each discovered host to guarantee user access to the remote resource. Then, the dynamic attributes of each host are gathered from its local Grid Resource Information Service (GRIS). This information is used by an user-provided rank expression to assign a rank to each candidate resource. Finally, the resultant prioritized list of candidate resources is used to dispatch the job.

The resource selection overhead is determined by the cost of retrieving the static and dynamic resource information, the detection of the resource availability, and the scheduling process itself. In the present case, the cost of scheduling jobs, i.e. rank calculation, can be neglected compared to the cost of accessing the Grid Information Services (Globus MDS), which can be extremely high [18].

In order to reduce the information retrieval overhead, the GIIS information is locally cached at the client host and updated every 5 min, this update frequency determines how often the testbed is searched for new resources. The GRIS contents are also cached locally but updated every 30 s, since the CPU availability information is generated every 30 s by the GRIS provider.

The availability of a resource is measured in terms of the CPU load in the case of a fork job manager, or the number of free slots in the case of a DRMS like PBS. Therefore, GridWay waits to submit the next job until the CPU load published in the GRIS decreases to a given threshold or the number of free slots is greater than zero. We have experimentally determined that the mean scheduling time per task (T^{sch}) in the testbed is 60 s.

3.3. Benchmark implementation

The Distributed Resource Management Application API (DRMAA) Working Group [19], within the Global Grid Forum (GGF) [20], has developed an API specification that allows a high-level interaction with Distributed Resource Management Systems (DRMS). The DRMAA standard constitutes a homogeneous interface to different DRMS to handle job submission, monitoring and control, and retrieval of finished job status.

Although DRMAA could interface with DRMS at different levels, for example at the intranet level with SGE or Condor, in the present context we will only consider its application at Grid level. In this way, the DRMS (Grid*Way* in our case) will interact with the local job managers (e.g. Condor, PBS or SGE) through

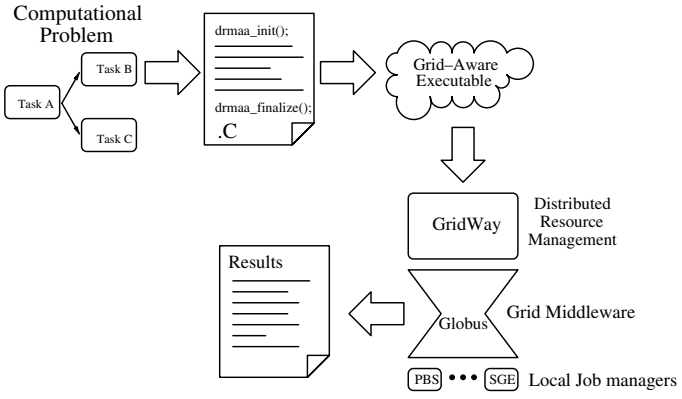

Fig. 3. Development and execution cycle using the DRMAA interface.

```
// Initialization
jt= SP;

drmaa_init(contact, err);

// Submit all jobs simultaneously and wait for them
drmaa_run_bulk_jobs(job_ids, jt, 0, num_tasks, 1, err);
drmaa_synchronize(job_ids, timeout, 1, err);

drmaa_exit(err);
```

Fig. 4. Implementation of the ED benchmark using DRMAA.

the Grid middleware (Globus). This development and execution scheme with DRMAA, Grid*W*ay and Globus is depicted in Fig. 3.

DRMAA represents a suitable and portable API to express distributed communicating jobs, like the NGB. In this sense, the use of standard interfaces allows the comparison between different Grid implementations, since neither NGB nor DRMAA are tied to any specific Grid infrastructure, middleware or tool.

The ED benchmark, as described in Section 2.1, comprises the execution of several independent tasks. Each one consists in the execution of the SP flow solver [12] with a different initialization parameter for the flow field. In the present work we have used the FORTRAN serial version of the SP flow solver code. These kind of HTC applications can be directly expressed with the DRMAA interface as *bulk* jobs [21]. The DRMAA code for this benchmark is shown in Fig. 4.

4. Results

In this section, we analyze the performance offered by the testbed presented in Section 3 using the Embarrassingly Distributed (ED) NAS Grid benchmark. In order to stress the computing capabilities of the testbed we will consider the ED family class D (72 tasks width). However, we preserve the problem size of the class A to retain the computational requirements of the benchmark. It is interesting to note that growing the problem size as stated in the NGB specification would limit the number of available machines in the testbed and would not reflect the computational profile of interest in this work.

Class A tasks are executed on those resources with at least 256 MB of main memory to prevent memory swapping. This is implemented by imposing such requirement in the resource discovery stage of the resource selection process. In the following experiments, cepheus is used as client and so it stores the executable and input files, and receives the output files.

4.1. Performance study

We begin the analysis presenting the intrusive and non-intrusive measurements made on the testbed of the parameters r_∞ and $n_{1/2}$. The ED benchmark class D allows us to evaluate the behavior of the testbed in saturation, i.e. subject to a workload that exceeds its processing capacity. The benchmarking measurements have been performed six times during April 2004 at different times of the day, to capture in the subsequent analysis the variability on the bandwidth, and response times of the local resource managers.

Fig. 5 shows the experimental performance obtained in two executions of the ED benchmark, along with that predicted by Eq. (4). The r_∞ and $n_{1/2}$ have been calculated by linear fitting to: (i) experimental results obtained in the execution of the benchmark; (ii) Eq. (7) using the average file transfer and execution times

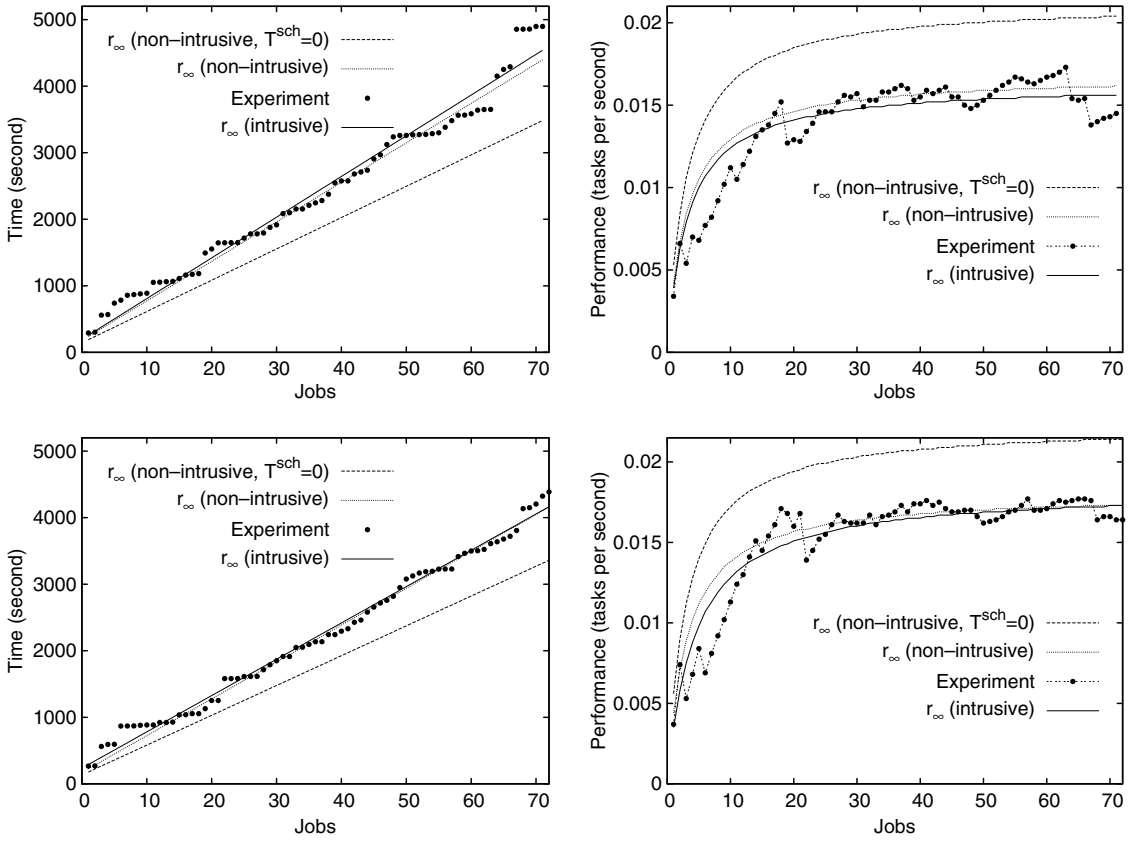

Fig. 5. Measurements of r_∞ and $n_{1/2}$ on the testbed based on experimental data, and raw resource performance (left-hand charts). Experimental performance of the ED benchmark on the testbed, along with that predicted by Eq. (4) (right-hand charts).

of each host; and also (iii) Eq. (7) with $T^{\text{sch}} = 0$. As can be observed from these plots the first-order description made by Eq. (4) provides a good characterization of the Grid.

The resource selection process reduces the asymptotic performance of the Grid, because of a delay between tasks consecutively submitted to the same host. This delay is mainly due to the Globus MDS update frequency and the Grid*Way* resource broker. However, it does not affect to the $n_{1/2}$ parameter (see also Table 2), since the brokering overhead increases the execution time (T_i) by the same amount in all the resources.

Table 2 shows the values of the r_∞ and $n_{1/2}$ coefficients, for the sake of completeness we also include the turnaround time (T_{Grid}) of each execution of the benchmark. Based on the *intrusive* results, the testbed is characterized by an average asymptotic performance of 0.016 tasks per second. In order to achieve the half of this asymptotic performance it is necessary to execute, on average, three tasks. And so, the apparent number of

Table 2
Turnaround time T_{Grid} (min), r_∞ (tasks per seconds), $n_{1/2}$ (tasks), and degree of heterogeneity coefficient v for each experiment

Intrusive		Non-intrusive		$T^{\text{sch}} = 0$		v	T_{Grid}
r_∞	$n_{1/2}$	r_∞	$n_{1/2}$	r_∞	$n_{1/2}$		
0.0146	2.21	0.0159	2.99	0.0200	2.95	0.25	85.2
0.0161	3.54	0.0164	3.03	0.0212	3.06	0.39	86
0.0138	3.23	0.0141	3.02	0.0192	3.09	0.36	104.8
0.0162	3.63	0.0167	3.03	0.0192	3.09	0.4	83.1
0.0163	3.21	0.0168	3.02	0.0215	3.01	0.36	81.6
0.0183	4.3	0.0181	3.08	0.0223	2.98	0.48	73.2

resources to the application is approximately 6, with an execution time of 375 s per task. This could be expected since this time is roughly equal to the average performance of the six hosts (see wall time in Table 3, note that this time must be scaled for multi-processors).

Fig. 6 summarizes the Grid performance during the six executions of the ED.D benchmark, along with the performance model calculated using the previous parameters:

$$r(n) = \frac{0.016}{1 + 3/n}. \tag{9}$$

In general, because of the dynamic availability and performance of Grid resources, the r_∞ and $n_{1/2}$ will not be constant for a Grid (see Table 2).

Let us now consider a testbed obtained by removing the resources from LCASAT and RedIRIS. The measurement of $r_\infty = 0.0125$ and $n_{1/2} = 2.1$ are shown in Fig. 7. The subtraction of this seven processors reduces the asymptotic performance of this new testbed in 20%. Note also that the apparent number of processors to the application in this case is 4, as could be expected from the previous discussion.

In order to analyze the difference in performance that a user could expect, let us consider the Grid speed-up (S_{site}), as defined in Eq. (8). In this case we will estimate the optimum execution, time (T_{site}), by calculating the optimum application *makespan* [13,22]. Fig. 8 shows the achieved speed-up by each site in the execution of the ED.D benchmark.

A similar study can be performed by means of the critical problem size (n_{site}) and the performance model of the global Grid. In this way, we can estimate, for a given site, the number of jobs from which it is more advantageous to use the entire Grid rather than its own resources. Fig. 9 shows the Grid performance model along with the peak performance of each organization (T_{site}). The critical problem size is determined by the intersection of the performance line of each organization with the testbed model.

Table 3
Execution and transfer time (s) statistics over all executions

Host	Execution time		Transfer time		Wall time	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
hydrus	263.28	36	27.25	14.08	290.52	39.81
babieca	705.55	166.29	427.25	294.75	1119.38	349.95
ramses	1043.19	173.84	28.34	47.43	1071.54	178.02
cygnus	249.81	10.02	27.37	14.52	277.18	17.05
khafre	1602.97	210.4	68.68	3.65	1671.65	210.68
platon	834.12	22.74	26.54	3.68	860.66	23.7

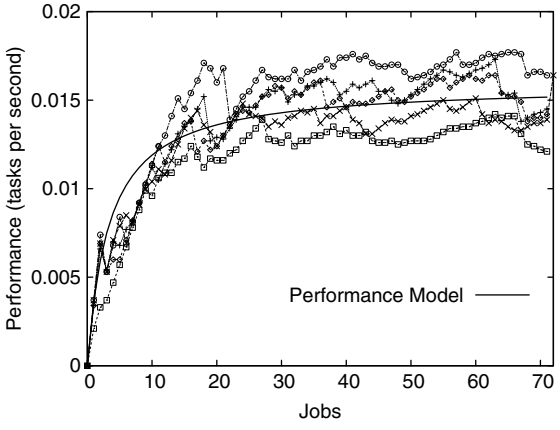

Fig. 6. Throughput (jobs per second) for six different executions of the ED benchmark, and testbed performance model.

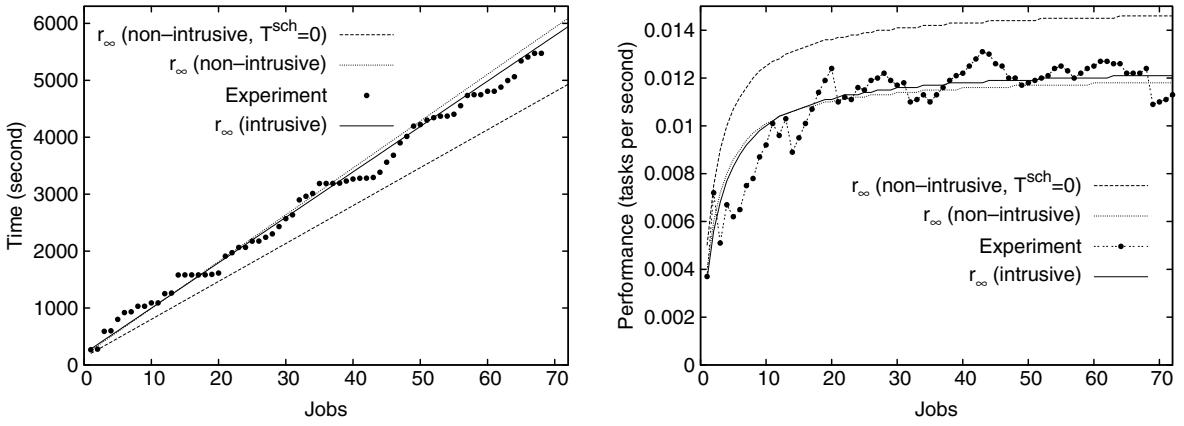

Fig. 7. Measurements of r_∞ and $n_{1/2}$ on the testbed based on experimental data, and raw resource performance (left-hand chart). Experimental performance of the ED benchmark on the testbed without LCASAT and RedIRIS site, along with that predicted by Eq. (4) (right-hand chart).

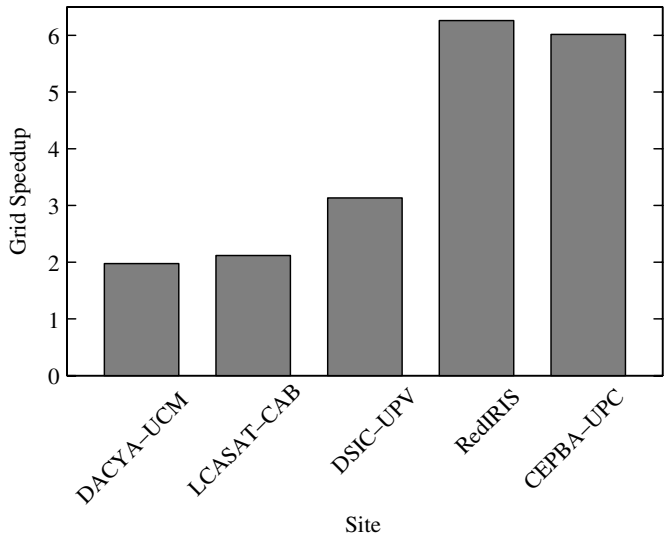

Fig. 8. Grid speed-up (S_{site}) for each site.

As shown in Fig. 9, for some sites (RedIRIS, UPC) the modest computational resources contributed to the Grid makes the use of the testbed advantageous from the first job. As a consequence these sites present a higher Grid speed-up (see Fig. 8). Nevertheless, for other organizations, the number of jobs from which an application can benefit from using the Grid ranges between 2 and 4 jobs, and so obtain a lower Grid speed-up.

4.2. Diagnostic study

The above metrics provide high level information of interest for the user. However, and due to the distributed nature of the Grid, it is very important to quantify the overheads induced by all its components, and to analyze their influence in the global performance.

Apart from allowing the performance characterization of the testbed, the ED benchmark enables the study of the Grid components' behavior. This study is of special interest for diagnostic and tuning purposes, interesting for application and middleware developers, and system administrators and architects [5].

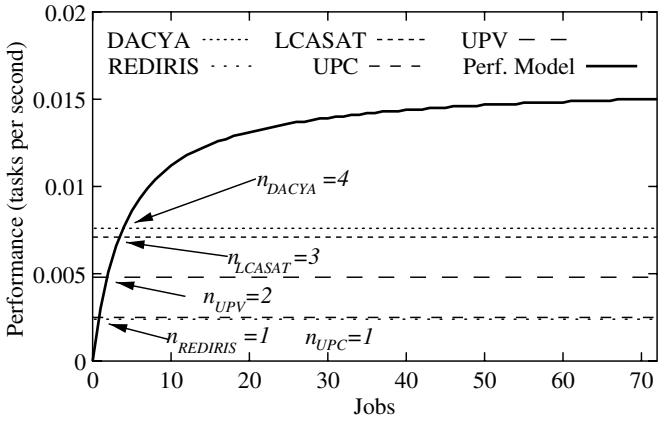

Fig. 9. Grid critical problem size (n_{site}) for each site.

As can be observed from Table 3, resources with fork local job managers present a lower standard deviation (hydrus, cygnus and platon). This fact is mainly due to a lack of additional scheduling in these job managers, and the absence of any workload when the benchmarking measurements were performed. However, clusters and SMP systems in the testbed are characterized by a greater standard deviation (babieca, ramses and khafre) given by the dynamism of the PBS resource manager and by the shared access to system resources (memory and I/O), respectively.

On the other hand, due to the network topology, those hosts directly connected to RedIRIS through a 1 Gbps non-dedicated link presents a low variability and average transfer time of 27 s, with the exception of khafre. This is because of the 30 s polling period used by the GRAM (Globus Resource Allocation Manager) *job manager* in the Globus version installed on this resource (GT2.2), whereas the *job manager* in the version installed on the rest of resources (GT2.4) uses a polling period of 10 s. Lastly, babieca presents a very high mean and standard deviation due to a problem with the *job manager*, which remains unsolved.

5. Conclusions

In this paper we have presented a methodology to analyze the performance of computational Grids in the execution of high throughput computing applications. The benchmarking process is based on the execution of the embarrassingly distributed benchmark of the NGB suite, with an appropriate scaling, to stress the computational capabilities of the testbed. We have also used a performance model for the Grid in saturation, whose suitability has been empirically proved in a research testbed. This performance model enables the comparison of different platforms in terms of their asymptotic performance (r_∞) and half-performance length ($n_{1/2}$) parameters.

Moreover, we have proposed a non-intrusive way to compute these parameters, based on the experimental average performance of each resource. The average resource performance could be obtained with specific light-weight non-intrusive probes which provide continual information on the health of the Grid environment. In this way, the r_∞ and $n_{1/2}$ parameters also provides an overview of the dynamic capacity of the Grid, which could eventually be used to generate global meta-scheduler strategies.

The execution of ED benchmark together with the site speedup and critical problem size, has been used to analyze the potential gain in performance that a Grid site could obtain by joining the Grid. And so, the speedup that can be expected when executing HTC applications on the Grid.

The benchmarking process used in this work also serve as an useful diagnostic tool. As could be expected PBS job managers and SMP systems present a greater variability on the execution times. However, in our testbed, this is not the case for the file transfer times because of the quality of the interconnection links between sites. This analysis also helped to identify incorrectly configured GRAM modules, a jobmanager problem in one of the resources, and the influence of the GRIS update frequency.

Acknowledgments

We would like to thank all the research centers that generously contribute resources to the experimental testbed. They are the European Center for Parallelism of Barcelona (CEPBA) in the Technical University of Catalonia (UPC), the Department of Computer Architecture and Automatics (DACYA) in the Complutense University of Madrid (UCM), the Department of Information Systems and Computation (DSIC) in the Polytechnic University of Valencia (UPV), the Laboratory of Advanced Computing, Simulation and Telematic Applications (LCASAT) in the Center for Astrobiology (CAB), and the Spanish National Research and Education Network (RedIRIS). All of them are part of the Spanish Thematic Network on Grid Middleware.

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, Lapack Users' Guide, third ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
[2] T.S.P.E. Corporation, http://www.spec.org/.
[3] D.H. Bailey, E. Barszcz, J.T. Barton, The NAS parallel benchmarks, International Journal of Supercomputer Applications 5 (3) (1991) 63–73.
[4] R. Wolski, N. Spring, J. Hayes, The network weather service: a distributed resource performance forecasting service for metacomputing, Future Generation Computing Systems 15 (5) (1999) 757–768.
[5] G. Chun, H. Dail, H. Casanova, A. Snavely, Benchmark probes for grid assessment, in: Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS04), 2004.
[6] G. Tsouloupas, M.D. Dikaiakos, Gridbench: a tool for benchmarking grids, in: 4th International Workshop on Grid Computing (GRID 2003), 17 November 2003, Phoenix, AZ, USA, Proceedings, IEEE Computer Society (2003) 60–67.
[7] R.F. Van der Wijngaart, M.A. Frumkin, NAS Grid Benchmarks Version 1.0, Technical Report NAS-02-005, NASA Advanced Supercomputing (NAS) Division, NASA Ames Research Center, Moffett Field, CA, 2002.
[8] L. Peng, S. See, J. Song, A. Stoelwinder, H.K. Neo, Benchmark performance on a cluster grid with NGB, in: Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), 2004, p. 275a.
[9] R.F. Van der Wijngaart, M.A. Frumkin, Evaluating the information power grid using the NAS grid benchmarks, in: Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), 2004, p. 275b.
[10] R.W. Hockney, C.R. Jesshope, in: Parallel Computers 2: Architecture Programming and Algorithms, Adam Hilger Ltd., Bristol, 1988, pp. 81–116.
[11] M.A. Frumkin, R.F. Van der Wijngaart, NAS Grid Benchmarks: a tool for grid space exploration, Journal of Cluster Computing 5 (3) (2002) 247–255.
[12] D.H. Bailey, E. Barszcz, J.T. Barton, The NAS parallel benchmarks, Journal of Supercomputer Applications 5 (3) (1991) 63–73.
[13] E. Huedo, R.S. Montero, I.M. Llorente, Experiences on adaptive grid scheduling of parameter sweep applications, in: Proceedings of the 12th Euromicro Conference Parallel, Distributed and Network-based Processing (PDP2004), IEEE CS Press, 2004, pp. 28–33.
[14] I. Foster, What Is the Grid? A Three Point Checklist, GRIDtoday 1(6). Available from: <http://www.gridtoday.com>.
[15] I. Foster, C. Kesselman, Globus: a metacomputing infrastructure toolkit, Journal of Supercomputer Applications 11 (2) (1997) 115–128.
[16] J.M. Schopf, Ten actions when superscheduling, Technical Report GFD-I.4, Scheduling Working Group—The Global Grid Forum, 2001.
[17] E. Huedo, R.S. Montero, I.M. Llorente, A Framework for adaptive execution on grids, Software—Practice and Experience (SPE) 34 (7) (2004) 631–651.
[18] X. Zhang, J. Freschl, J. Schopf, A performance study of monitoring and information services for distributed systems, in: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03), Washington, USA, 2003, pp. 270–280.
[19] Distributed Resource Management Application API Working Group—The Global Grid Forum, http://www.drmaa.org, 2004.
[20] The Global Grid Forum, http://www.gridforum.org, 2004.
[21] H. Rajic, R. Brobst, W. Chan, F. Ferstl, J. Gardiner, J.P. Robarts, A. Haas, B. Nitzberg, J. Tollefsrud, Distributed resource management application API Specification 1.0, Technical Report, DRMAA Working Group—The Global Grid Forum, 2003.
[22] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, second ed., Prentice-Hall, New Jersey, NJ, 2002.