ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Bari

INFN/TC-02/18
11 Luglio 2002

SCHEDULING SYSTEM IN A GRID ENVIRONMENT

Marcello Castellano[1,2], Giacomo Piscitelli[1,2], Nicola Simeone[1,2]
Domenico Di Bari[2,3], Daniela Cozza[3]
[1)]Department of Electrical and Electronic Engineering, Polytechnic of Bari, Via Orabona 4, 70126 Bari, Italy
[2)]INFN-Sezione di Bari, Via Orabona 4, 70126 Bari, Italy
[3)]Dipartimento Interateneo di Fisica dell'Univ. di Bari, Via Orabona 4, 70126 Bari, Italy

Abstract

In this paper, extensive results of simulation experiments are presented, evaluating the scalability degree of batch and on-line mode scheduling techniques in a data-intensive grid environment. On-line techniques reveal a slight decrease in performance when scalability increases. Conversely batch techniques performance suffer for the local nature of the state estimation technique. Experimental results are reported for the following scheduling algorithms: Minimum Completion Time, Minimum Execution Time, Switching Algorithm, Opportunistic Load Balance and batch scheduling algorithm: MinMin, MaxMin and Sufferage.

A grid multi-thread toolset simulator has been used which is based on a java class library. A simulated scenario in agreement with a number of forthcoming scientific data-intensive applications foreseen by the European Data Grid Project has been taken into account. A statistical model based workload-description is considered and the slowdown metric is applied to evaluate the grid performance.

PACS.: 07.05
Keywords: Computational Grids, DataGrid, Scheduling, Distributed System

Published by ***SIS–Pubblicazioni***
Laboratori Nazionali di Frascati

1 INTRODUCTION

The progress in network technology is enabling high-performance distributed computing, in which computational and data resources in wide area network are logically used to solve large-scale problems. The real and specific problem underlying the Grid concept is the availability and sharing of resources in dynamic, multi-institutional virtual organizations[1]. The computational Grid category denotes systems that have higher aggregate computational capacity available for single applications than the capacity of any constituent machine in the system. The data Grid category is for systems that provide an infrastructure for synthesizing new information from data repositories such as digital libraries or data warehouses that are distributed in a wide area network. Moreover data grids exhibit specialized storage management and data access services[2].

A Grid based system needs for heterogeneous resource management and wide-area scalability. Many organizations based on network-distributed computing systems require a flexible overall system configuration that enables site autonomy, multiple scheduling polices and fault tolerance. A grid might extend from few resources to several ones. The problem arises of potential performance degradation as grids size increases. As a consequence, experience with two decades of parallel and distributed applications indicates that scheduling is fundamental for performance. The overall aim is to efficiently and effectively schedule the applications that need to utilize the available resources in the Metacomputing environment.

Designing high-performance scheduling systems for grids is particularly challenging: both the software and hardware resources of the underlying system may exhibit heterogeneous performance characteristics, resources may be shared by other users, and networks, computers and data may exist in distinct administrative domains[3].

Data-intensive, high-performance computing applications require the efficient management and transfer of terabytes or petabytes of information in wide-area, distributed computing environments. Examples of data-intensive applications include experimental analyses and simulations in several scientific disciplines, such as high-energy, climate modeling, earthquake engineering and astronomy. The data Grid initiatives, European DataGrid Project[4] and Globus[5], are working on developing large-scale data organization, catalog, management, and access technologies.

In this paper a study is presented on the scalability of a scheduling system in a data-intensive Grid environment by a toolset grid simulator. In section 2 the reference data grid system model is described. The design of the scheduling system is reported in section 3 with online and offline scheduling techniques. Results and discussions are in the last section.

2 DATAGRID SYSTEM MODEL

In this work it is assumed that the Grid available to the user has the following topology: it is a set of computing resources that are accessible to the user via high speed network links and are hierarchically organized. This is not only a logical topology, but attempt to take into account the future physical network topology of the Grid, according to the eu-DataGrid specifications[6]. Each site contains a number of hosts, where a host can be any computing platform, from a single-processor workstation to multi parallel processor system, which is available for computation. Each site contains a certain number of mass storage elements too. From now on, these are referred as Computing Elements (CE) and Storage Elements (SE). The distribution of the computational power and data is modeled upon the MONARC project guidelines: every site has a computational power equivalent to those who are below it and all the data owned by a site are replicated on the site below it[7]. The GRID provides a Scheduling Service, an Information Service which takes into account both the status of the available resources and the dynamic information about the system load. Moreover a Replica Management Service named Replica Catalogue is also considered, which provides both the correspondences between logical file names and physical file names and the location of replicated files.

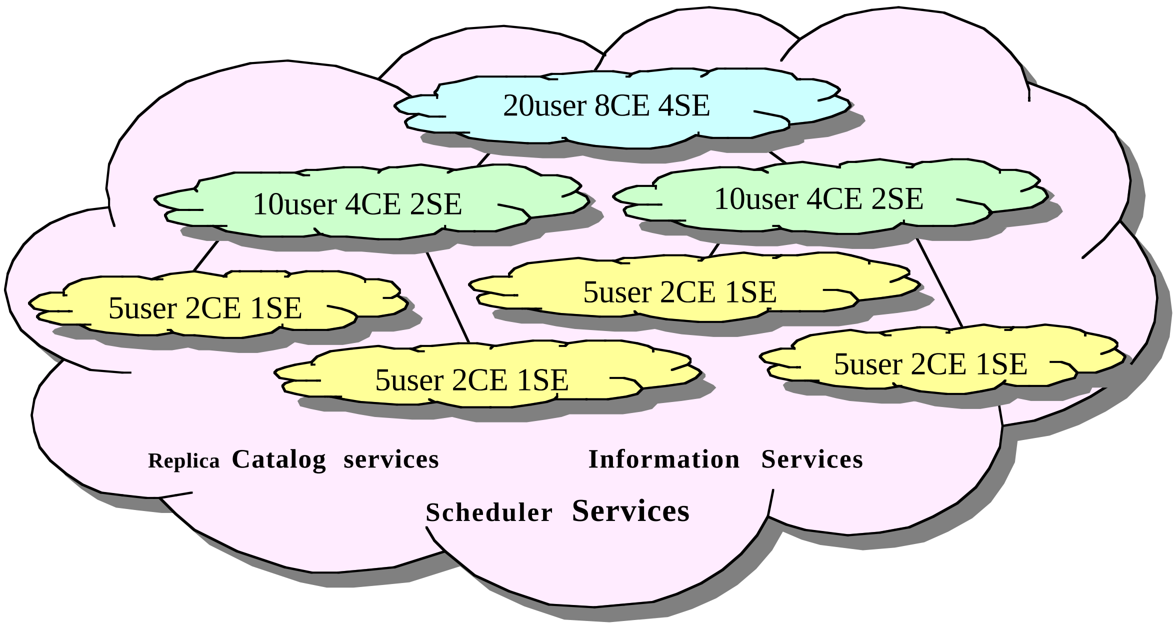

FIG. 1: The Reference System model.

3 DESIGNIN A GRID SCHEDULER

In a grid environment, the scheduling is performed in two phases: Resource Discovery and Mapping. The Resource Discovery comprises all the operations needed to collect information about available resources; the Mapping comprises all the operations required to map the requested jobs on the resource and to order their execution over time. In order to design the grid scheduler some scheduling policies, an application model and a performance model have been defined[3].

3.1 Scheduling Policies

In this work, two types of scheduling are considered: on-line mode scheduling and batch mode scheduling. The on-line algorithms are: the Minimum Completion Time, the Minimum Execution Time, the Opportunistic Load Balance and the Switching Algorithm [8] [9]. The batch algorithms are: the MinMin, the MaxMin and the Sufferage[10] [11]. A backfilling technique is considered too[12]. In the on-line mode scheduling, each task is scheduled without considering the previous and following tasks. When the arrival rate is low, the CEs may be ready to execute a task as soon as it arrives at the scheduler. Therefore, it may be beneficial to use the scheduler in the on-line mode so that a task need not wait until the next scheduling event to begin its execution. In the batch mode, the scheduler considers more than one task per time. This allows making better decisions, because the scheduler disposes of the resource requirement information for all the task. When the task arrival rate is high, there will be a sufficient number of tasks to keep the machines busy in between the mapping events, and while a mapping is being computed.

3.1.1 On-line Mode Techniques

The minimum completion time (MCT) heuristic assigns each task to the machine that results in that task’s earliest completion time. This causes some tasks to be assigned to machines that do not guarantee the requested minimum execution time. As a task arrives, all the Computing Elements in the Grid are examined to determine the Computing Element that assures the earliest completion time for the task.

The minimum execution time (MET) heuristic assigns each task to the Computing Element that performs that task’s computation in the least amount of execution time (this heuristic is also known as limited best assignment (LBA) and user directed assignment (UDA). This policy, in contrast to MCT, does not consider machine ready times and it can cause a severe imbalance in load across the machines. The advantages of this policy consist in assigning each task to the machine that guarantees the least amount of execution time.

The switching algorithm (SA) is motivated by the following considerations. The MET can potentially create load imbalance across machines by assigning many more tasks to some machines than to others, whereas the MCT tries to balance the load by assigning tasks for earliest completion time. If the tasks arrive in a random mix, it is possible to use the MET (at the expense of load balance) until a given threshold and then use the MCT to smooth the load across the machines. The SA heuristic uses the MCT and MET in a cyclic fashion depending on the load distribution across the machines. The purpose is to have a heuristic with the desirable properties of both the MCT and the MET. Let the maximum (latest) Queuing Time over all machines in the suite be QTmax, and the minimum (earliest) Queuing Time be QTmin. Then, the load balance index (LBI) across the machines is given by QTmin/QTmax. The parameter LBI can have any value in the interval [0,1], if LBI is 1 then the load is evenly balanced across the machines. If LBI is 0, then at least one machine has not yet been assigned a task. Two threshold values, LBIlow and LBIhigh, for the ratio LBI are chosen. Initially, the value of LBI is set to 0. The SA heuristic begins mapping tasks using the MCT until the value of load balance index increases to at least LBIhigh. Then, the SA begins using the MET to perform task mapping. This typically causes the load balance index to decrease. When it decreases to LBIlow, the SA switches back to using the MCT for mapping the tasks and the cycle continues.

The opportunistic load balancing (OLB) assigns a task to the machine that becomes ready next, without considering the execution time of the task onto that CEs. If multiple machines become ready at the same time, then one machine is arbitrarily chosen.

Backfill is a scheduling optimization which allows a scheduler to make better use of available resources by running jobs out of order. The jobs are executed according to a priority until a job which cannot be executed (there aren't sufficient resources), arrives; at this point the jobs go in a waiting queue and the new jobs are mapped only if they not delay the starting time of the waiting jobs [9)].

3.1.2 Batch Mode Techniques

Let's pile a certain number or jobs and, after the Resource Discovery, let's build a matrix, called MapTable, containing a column for each task and a row for each CE available. Let's build a vector containing the QT of each CE too, let's call it ceTable. Every cell of the MapTable contains the Estimated Execution Time on that CE.

For each job the CE that gives the earliest Completion Time (CT=ET+QT) is determined by scanning the related column of the MapTable. The job with the minimum CT, among all job piled, is executed. Then the ceTable is updated, considering that the CE chosen has to execute that job and the corresponding column of the MapTable, deleted. The cycle is repeated until all the jobs are mapped.

The MaxMin is quite similar to MinMin. The MapTable is built in the same way but the maximum CT, instead of the minimum, among all the minima is chosen.

The rationale behind Sufferage is that a job should be assigned to the CE that would ”suffer" the most if not assigned to that CE. For each task, its Sufferage value is defined as the difference between its best MCT and its second-best MCT. Tasks with high Sufferage values take precedence[10)].

3.2 Application Model

The requested jobs are independent each other, and every job is characterized by a Computational Weight per Megabyte and by an input file. The workload is generated by the use of a statistical model, instead of the use of a real trace based approach. Every User generates a request with a limited exponential probability distribution inter arrival time. The computational weight of the requested job is obtained by querying a table. This table could be updated by the user himself or by the schedulers. In our modeled system it is a static table and the computational weight has a limited exponential probability distribution. The bounds of the probability distributions are a decreased version of the requirements of the HEP applications.

3.3 Performance Model

The performance model describes the behavior of the job on the underlying system. In particular, the resource discovery provides information about the job, see TAB. 1a, and the suitable computing elements as shown in TAB. 1b.

Job's Computational weight per Mb	[SpecInt95*s/Mb]
Input File Size	[Mb]

TAB. 1a: Information about the Job after the Resource Discovery

Average CE Computational Power	[SpecInt95]
CE's Running Jobs	[ ]
Average Traversal Time	[ms]

TAB. 1b: Information about the CE after the Resource Discovery

In order to evaluate the CE execution time, the Job Computational Weight and the CE Average Computational Power have been determined using the parameters defined in TAB. 1a and TAB. 1b (see TAB. 2).

Job's Computational Weight	JobID.ExecWeight * InputLFN.FileSize	[SpecInt95*s]
CE's Computational Power	CE.AverageSI / (CE.TotJobs+1)	[SpecInt95]
Execution Time	JobID.CW / CE.SI	[s]

TAB. 2: Performance Parameters estimation

Moreover it is supposed that the chosen Computing Element is always 'close' to the Storage Element.

4 EXPERIMENTAL RESULTS AND DISCUSSION

A multi-thread toolset simulator [13) 14) 15)] has been used to produce numerical results about the study on the scalability of the data-intensive grid scheduling system presented above. Two sets of simulations have been performed, the first one using the less scalable approach (centralized) as shown in fig. 2a and the last one with the more scalable approach (distributed) as shown in FIG. 2b.

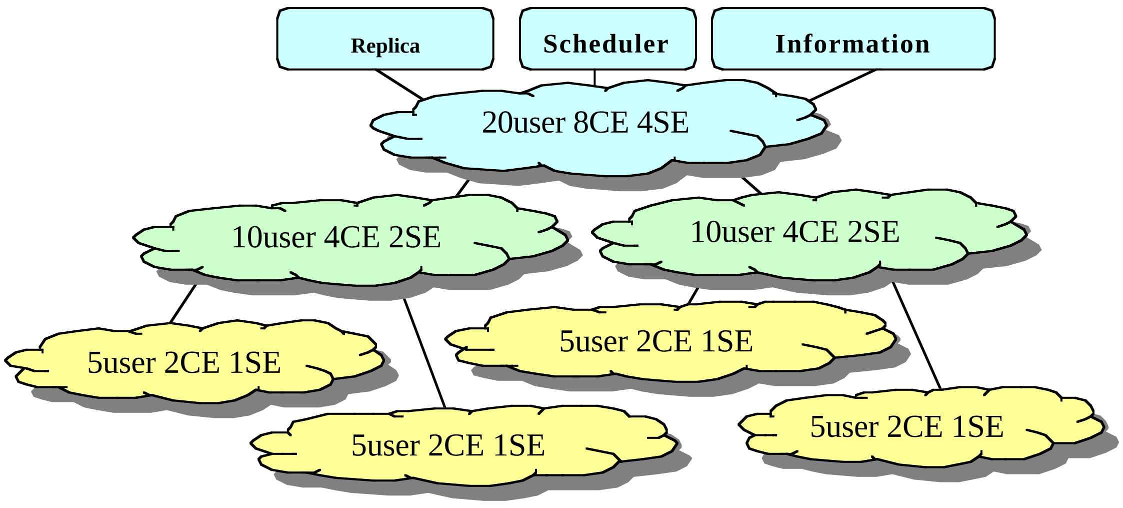

FIG. 2a: Centralized scheduling.

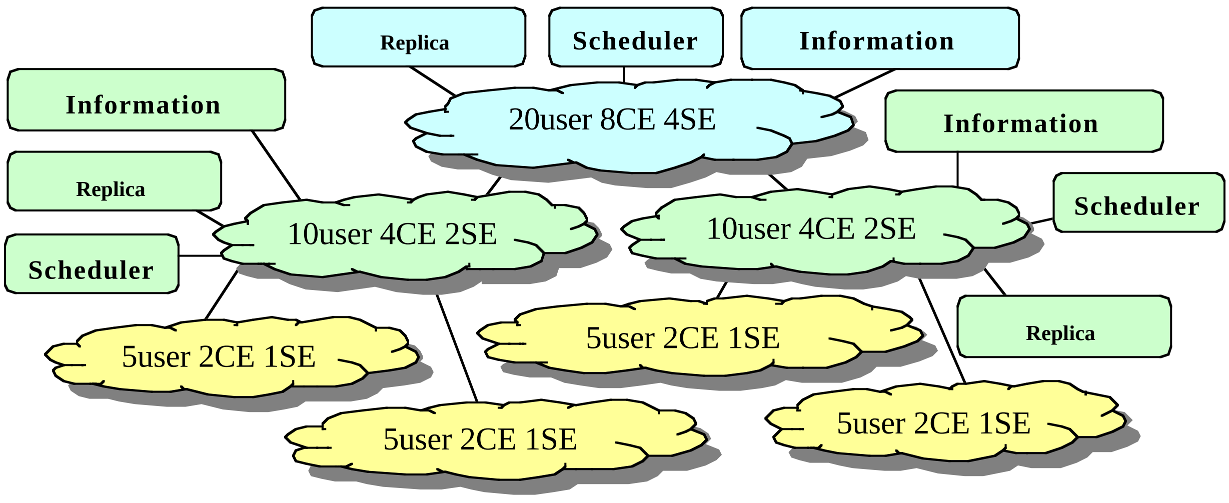

FIG. 2b: Distributed Scheduling.

The response time is the most suitable metric for open on_line systems. However this metric is dependent on the long/short type jobs that can characterize the system load. To avoid this condition, the slowdown metric, defined as runtime on a loaded system divided by runtime on a dedicated system has been adopted [16)].

Comparing the centralized vs distributed approach results shown in FIG.3, it is possible to reveal that the on-line techniques slightly decrease in performance when scalability increases, while the batch ones critically decrease due to the local nature of the state estimation technique. TAB. 3 shows the results in a quantitative fashion.

The Backfill is the worst technique, because it is the only approach that needs a higher and variable number of query per job to the Information Service. All the other techniques cause the same traffic towards the Information System and in this case the performances depend only on the used algorithms.

MCT	13%
MET	1%
SA	1%
OLB	0%
Backfill	67%
MinMin	104%
MaxMin	170%
Sufferage	85%

TAB. 3: Performance degradation when a distributed approach instead of a centralized one is used.

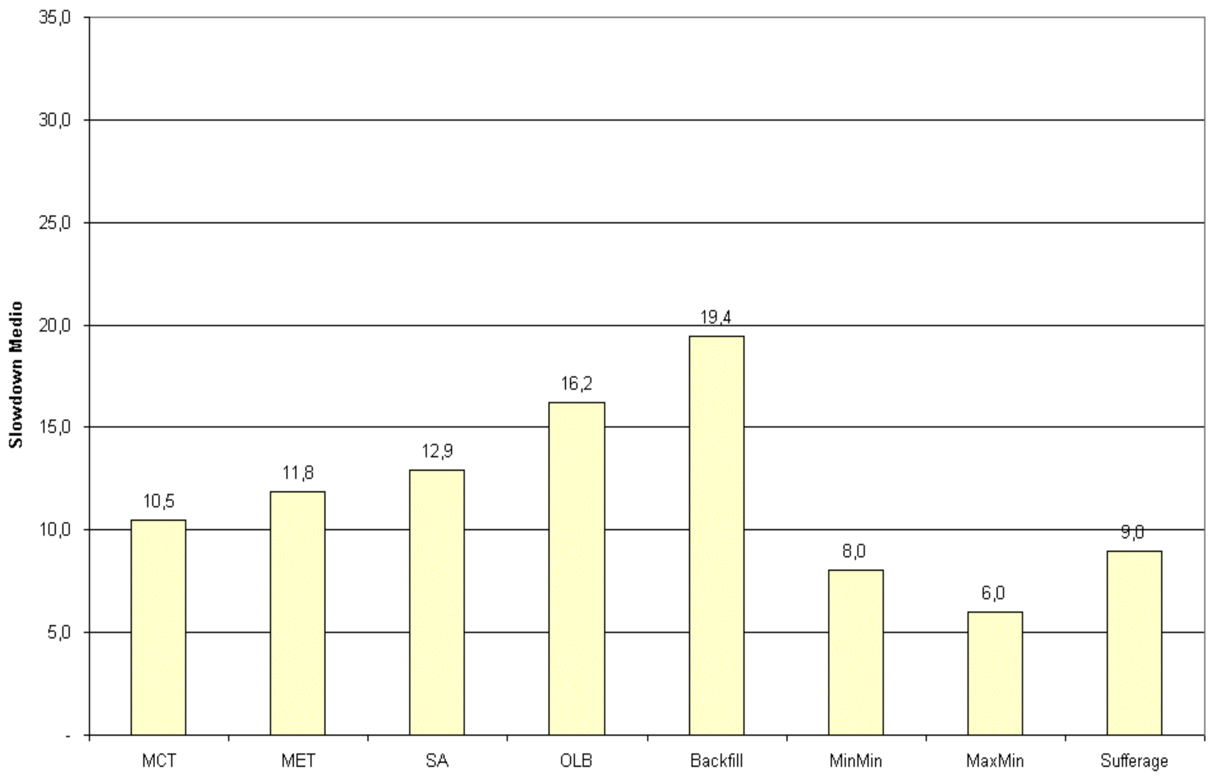

FIG. 3a: - Comparisons of the on-line and batch mode scheduling using the centralized approach.

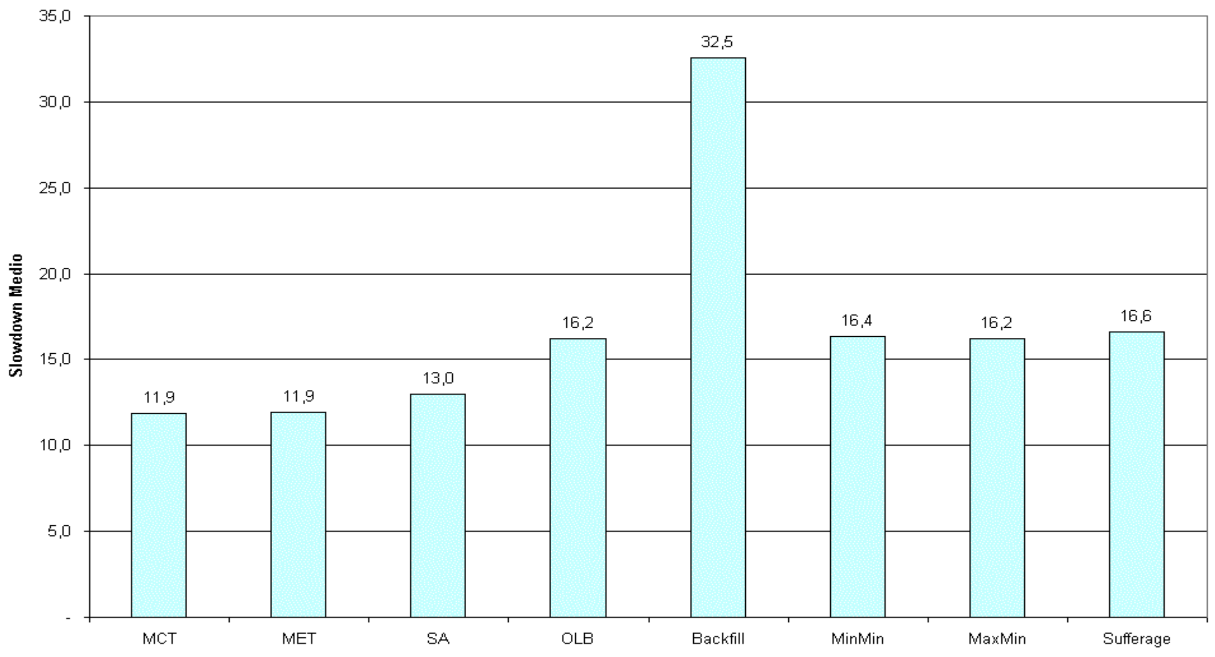

FIG. 3b: Comparisons of the on-line and batch mode scheduling using the distributed approach.

5 CONCLUSION

In this paper, extensive results of simulation experiments are presented, evaluating the scalability degree of batch and on-line mode scheduling techniques in an online open system data-intensive grid environment. The measures reveal for on-line techniques a slight decrease in performance when scalability increases. Conversely batch techniques performance suffer for the local nature of the state estimation technique. The comparison between on-line and batch mode techniques reveals that this last don't bear the presence of other schedulers. Hence, the online techniques are more suitable for solutions based on distributed scheduling systems rather than batch techniques.

6 REFERENCES

(1) I.Foster, C.Kesselman and S.Tuecke. The Anatomy of the Grid Enabling Scalable Virtual Organizations, Intl J. Supercomputer Applications, 2001.

(2) K.Krauter,R.Buyya and M.Maheswaran. A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing, Software Practice and Experience,1-7, 2001.

(3) F. Berman. High-performance schedulers. In I. Foster and C. Kesselman, editors, The Grid: Blueprint for a New Computing Infrastructure, pages 279-310. Morgan Kaufmann, San Fransisco, CA, 1999.

(4) www.eu-datagrid.org .

(5) www.globus.org

(6) http://server11.infn.it/workload-grid/

(7) M. Campanella, L. Perini. The analysis model and the optimization of geographical distribution of computing resources: a strong connection, MONARC note n. 1/98, 1998.

(8) R. Armstrong, D. Hensgen, and T. Kidd. The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions, *Proc. of the* 7th IEEE Heterogeneous Computing Workshop (HCW '98)," pp. 79–87, 1998.

(9) M.Maheswaran, S.Ali, H. J. Siegel, D. Hensgen and R. F. Freund. Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems, in Journal of Parallel and Distributed Computing-Special Issue on Software Support for Distributed Computing, v. 59,n.2,1999.

(10) H.Casanova, D.Zagorodnov and F.Berman. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments, Proceedings of the 9th Heterogeneous Computing Workshop, 2000, pp349-363, Cancun Mexico.

(11) T.D.Braun, et al. A Comparison Study of Static Mapping Heuristics for a Class of Meta-tasks on Heterogeneous Computing Systems, proceeding of the Eighth Heterogeneous Computing Workshop, 12 April, 1999 San Juan, Puerto Rico.

(12) D.Jackson, Q.Snell and M. Clement. Core Algorithms of the Maui Scheduler, Proceedings of the 7th Workshop on Job Scheduling Strategies for Parallel Processing, Cambridge, 2001.

(13) M.Castellano, G.Piscitelli, N.DiBari and E.Nappi. A description of a toolset for simulate a data grid , to be submitted to INFN as Technical Report, 2002.

(14) http://server11.infn.it/workload-grid/meetings/milano2.html

(15) http://server11.infn.it/archive-workload-eu-datagrid/att-0871/01-MinutesPrague.pdf

(16) D. Feitelson L. Rudolf, Metrics and Benchmarking for parallel Job Scheduling, Proceedings of the 4th Workshop on Job Scheduling Strategies for Parallel Processing, Orlando, 1998.