# Immediate mode scheduling in grid systems

**Article** *in* International Journal of Web and Grid Services · January 2007

DOI: 10.1504/IJWGS.2007.014075 · Source: DBLP

**4 authors**, including:

Fatos Xhafa
Universitat Politècnica de Catalunya
**798** PUBLICATIONS **9,742** CITATIONS
SEE PROFILE

Leonard Barolli
Fukuoka Institute of Technology
**1,393** PUBLICATIONS **10,655** CITATIONS
SEE PROFILE

Arjan Durresi
Indiana University-Purdue University Indianapolis
**385** PUBLICATIONS **5,250** CITATIONS
SEE PROFILE

# Immediate mode scheduling in grid systems

Fatos Xhafa

Department of Languages and Informatics Systems
Polytechnic University of Catalonia
Campus Nord, Ed. Omega, C/Jordi Girona 1–3
08034 Barcelona, Spain
Fax: +34–93–413–7833
E-mail: fatos@lsi.upc.edu

Javier Carretero

Department of Computer Architecture
Polytechnic University of Catalonia
Campus Nord, Ed. C6, C/Jordi Girona 1–3
08034 Barcelona, Spain
Fax: +34–93–413–7833
E-mail: jcarrete@ac.upc.edu

Leonard Barolli*

Department of Information and Communication Engineering
Faculty of Information Engineering
Fukuoka Institute of Technology (FIT)
3–30–1 Wajiro-higashi, Higashi-ku, Fukuoka 811–0295, Japan
Fax: +81–92–606–4970
E-mail: barolli@fit.ac.jp
*Corresponding author

Arjan Durresi

Department of Computer Science
Louisiana State University
298 Coates Hall, Baton Rouge, LA 70803, USA
Fax: +1 (225) 578–1465
E-mail: durresi@csc.lsu.edu

**Abstract:** Computational Grids (CGs) are nowadays successfully responding to increasing needs for high computation power. A key issue in CGs is the scheduling, which demands for efficient methods. In this work, we consider the scheduling problem in immediate mode, in which jobs are allocated as soon as they arrive in the system. This type of scheduling arises in many grid-based applications, especially, in real-time applications. We have implemented five immediate scheduling methods and have measured their performance with respect to four parameters: makespan, flowtime, resource utilisation and matching proximity by using a simulation benchmark for heterogeneous distributed systems. The computational results showed the performance of the immediate scheduling methods and allowed us to evaluate the advantages of these methods if we knew in advance certain grid characteristics (consistency of computing, heterogeneity of jobs and resources). The usefulness of the presented methods in web and grid scheduling services is also discussed.

**Keywords:** immediate mode; scheduling; Computational Grids; CGs; ETC simulation model; resource allocation; adaptive scheduling; grid services; real time applications.

**Reference** to this paper should be made as follows: Xhafa, F., Carretero, J., Barolli, L. and Durresi, A. (2007) 'Immediate mode scheduling in grid systems', *Int. J. Web and Grid Services*, Vol. 3, No. 2, pp.219–236.

**Biographical notes:** Fatos Xhafa received his PhD in Computer Science from the Polytechnic University of Catalonia (Barcelona, Spain) in 1998. He joined the Department of Languages and Informatics Systems of the Polytechnic University of Catalonia as an Assistant Professor in 1996 and is currently Associate Professor and member of the ALBCOM Research Group of this department. His current research interests include parallel algorithms, approximation and meta-heuristics, distributed programming, Grid and P2P computing.

Javier Carretero is a PhD student at the Department of Computer Architecture, Polytechnic University of Catalonia (Barcelona, Spain). His research interests include computer architecture, compilers, and operating systems, computer design and performance evaluation, and the development of applications aware of the underlying architecture. He is also interested in the fields of operational research, algorithmics, simulation and their applications to information and telecommunication technologies.

Leonard Barolli is a Professor in Department of Information and Communication Engineering, Fukuoka Institute of Technology (FIT), Japan. He received BE and PhD Degrees from Tirana University and Yamagata University in 1989 and 1997, respectively. He has published more than 200 papers in refereed journals and international conferences proceedings. He has served as a Guest Editor for many journals. He was PC Chair of IEEE AINA-2004 and ICPADS-2005 and General Co-Chair of IEEE AINA-2006. Presently, he is Workshops Chair of iiWAS-2006 and Workshops Co-Chair of IEEE AINA-2007 and ARES-2007. His research interests include high-speed networks, grid computing, P2P, *ad hoc* and sensor networks. He is member of IEEE, IPSJ and SOFT.

Arjan Durresi is an Assistant Professor of Computer Science at Louisiana State University. He received BE, MS and PhD from Polytechnic University of Tirana, Albania and a Superior Specialization Diploma in telecommunication from La Sapienza University in Rome, Italy and Italian Telecommunication Institute. He has 20 years of experience in industry and academic research, in the areas of network architectures, heterogeneous wireless networks, security, grid computing, *etc*. He has published more than forty journal and seventy conference papers. He also has over thirty contributions to standardisation organisations such as IETF, ATM Forum, ITU, ANSI and TIA. He is on the editorial boards of *Ad Hoc Networks Journal* (Elsevier) and a Senior Member of the IEEE.

## 1 Introduction

Computational Grids (CGs) emerged in late 1990s (Foster and Kesselman, 1998; Foster *et al*., 2001) to respond to the high demand for computational resources. By coupling together computers, workstations and LANs in a single virtual computational unit, CGs are able to provide to users direct access to computers, software, data, and other resources. CGs are primarily intended for large scale applications (Casanova and Dongarra, 1998; Wright, 2001; Linderoth and Wright, 2003; Frattolillo, 2005; Sun *et al*., 2005; Nefedova *et al*., 2006).

In many grid-based applications, an efficient scheduling of jobs or applications to grid resources is a key factor to ensure high performance applications. The characteristics of the CGs such as its high degree of heterogeneity of resources, its dynamic nature and the large-scale, make the scheduling problem very complex and challenging. Job scheduling in CGs can be viewed as a whole family of problems that try to capture different needs of grid applications in efficiently allocating jobs to grid resources. Many ideas and approaches from the scheduling in distributed systems can also be applied for scheduling in grid systems (the reader is referred to Karatza (2004) for a recent survey and to Hamscher *et al*. (2000) for different scheduling strategies for grid computing). Thus, different scheduling modes and, accordingly, different models and techniques known for conventional distributed systems could be used for matching scheduling needs of grid applications. In some applications such as Monte-Carlo simulations (Casanova *et al*., 2000), jobs could be periodic or they don't have deadline restrictions. In this case, batch mode scheduling would be appropriate. However, in many other applications such as multimedia applications (Zhang *et al*., 2005; Zaia *et al*., 2006) or interactive and real-time applications (Kranzlmuller *et al*., 2004; Zhang, 2002) could require immediate mode scheduling in which jobs must be scheduled as soon as they arrive in the system or adaptive scheduling (Casanova *et al*., 1999; Dail, 2002; Dail *et al*., 2003; Nikolopoulos and Polychronopoulos, 2003) in which resource allocation is done adaptively.

Scheduling in grids not only has to deal with new characteristics of the CGs such as dynamics and large-scale but also must consider other important parameters. Thus, unlike conventional distributed systems in which usually the throughput of the system is optimised, in CGs parameters such as flowtime, resource utilisation, matching proximity and load balancing must be also considered.

In the immediate mode scheduling, a job is scheduled as soon as it enters in the system without waiting for the next time interval when the scheduler will get activated or the job arrival rate is small having thus available resources to execute jobs immediately. In contrast to more sophisticated schedulers based on metaheuristics (Abraham *et al*., 2000; Buyya *et al*., 2000; Braun *et al*., 2001; Di Martino and Mililotti, 2004; Carretero and Xhafa, 2006; Xhafa, 2006) or economic models (Buyya, 2002; Buyya *et al*., 2002) that could need larger execution times to provide the allocation (*e.g*., a genetic-based scheduler would require more time for the population of individuals to converge), immediate scheduling could better respond to the dynamic nature of the grid systems and the presence of time restrictions on job executions on the grid. Immediate mode methods have been proposed in different works in the scheduling literature (Maheswaran *et al*., 1999; Abraham *et al*., 2000; Braun *et al*., 2001; Wu and Shu, 2001). Our main focus here is to adapt and implement these methods for the case of grid systems as well as to conduct a complete comparative study on their performance. To the best of our knowledge, this is the first time these methods are empirically studied altogether. Furthermore, in previous work the performance of these methods is measured using only the makespan of the system while we study their performance with regard to four parameters: *makespan*, *flowtime*, *resource utilisation* and *matching proximity.*

We consider the following five immediate mode methods: Opportunistic Load Balancing (OLB), Minimum Completion Time (MCT), Minimum Execution Time (MET), Switching Algorithm (SA) and *k*-Percent Best (*k*PB). Once implemented, these methods have been tested using the simulation benchmark proposed by Braun *et al.* (2001), which is obtained from the Expected Time to Compute (ETC) model that simulates distributed heterogenous systems. The experimental study revealed the performance of these methods with regard to the four considered parameters. Furthermore, based on the computational results, we are able to evaluate the performance of these methods for different grid scenarios regarding certain grid characteristics such as consistency of computing, degree of heterogeneity of resources and jobs. Thus, we can identify the usefulness of the considered methods if we knew in advance these grid characteristics and choose the appropriate method accordingly.

We also discuss some issues related to the integration of immediate methods in web and grid scheduling services to improve performance and load balancing in distributed web servers (Harchol-Balter *et al*., 2003; Cardellini *et al*., 1999). Moreover, the recent grid standards and architectural schemes such as the Open Grid Services Architecture (OGSA),[1] developed by the Globus Alliance and based on standard XML-based web services technology, require the identification and integration of efficient dynamic schedulers as part of global Grid service delivery. We observe that immediate mode schedulers are appropriate for decentralised scheduling services, that is, scheduling services that will integrate local schedulers, which will be in charge for scheduling user jobs to idle computers or servers. Immediate mode-based schedulers are especially efficient for LANs due to their small or moderate size and for sites with high throughput.

The rest of the paper is organised as follows. We give in Section 2 a description of the job scheduling in computational grids. The immediate mode methods considered in this work are given in Section 3. We give in Section 4 some computational results, which are evaluated in Section 5. We discuss in Section 6 the usefulness of the immediate mode scheduling methods in web and grid service scheduling and end up in Section 7 with some conclusions.

## 2 Problem description

Job scheduling in grids consists in efficiently allocating jobs to resources in a global, heterogenous and dynamic environment. The efficiency means that we are interested to allocate jobs as fast as possible and optimising the following (conflicting) criteria: *makespan, flowtime, resource utilisation* and matching proximity.

Jobs have the following characteristics: are originated from different users/applications, have to be completed in unique resource, are independent and could have their requirements over resources. On the other hand, resources could dynamically be added/dropped from the grid, can process one job at a time and have their own computing characteristics.

In order to formalise the instance definition of the problem, a benchmark simulation model is used. The main reason behind this choice is the difficulty of using at present real distributed grid systems to evaluate different grid scenarios, *e.g.*, different degree of heterogeneity of resources, which would require modifying the configuration of grid nodes. Instead, we use the ETC matrix model by Braun *et al.* (2001). In this model an instance of the problem can be formalised as follows:

- A *number* of independent (user/application) *jobs* to be scheduled
- A *number* of heterogeneous *machines* candidates to participate in the planning
- The *workload* of each job
- The *computing capacity* of each machine (in *mips*)
- Ready time $ready_m$ – when machine *m* will have finished the previously assigned jobs. This parameter measures the previous workload of a machine.
- The ETC matrix ($nb\_jobs \times nb\_machines$). $ETC[i]$ $[j]$ is the expected execution time of job *i* in machine *j*.

As an example of an ETC, we could consider the case when components $ETC[i]$ $[j]$ are computed by dividing the workload of a job by the computing capacity of a machine.

## 2.1 Optimisation criteria

Several parameters can be measured for a given schedule *S*. We consider the following:

- *Makespan* (finishing time of latest job) defined as $\min_S \max\{Fj : j \in Jobs\}$
- *Flowtime* (sum of finishing times of jobs), that is, $\min_S \sum_{j \in Jobs} F_j$
- *Resource utilisation*, in fact, we consider the *average resource utilisation*
- *Matching proximity.*

The resource utilisation parameter is defined using the *completion time* of a machine, which indicates the time in which the machine will finalise the processing of the previous assigned jobs as well as of those already planned for the machine. Formally, for a machine *m*, we define:

$$completion[m] = ready[m] + \sum_{j \in schedule^{-1}(m)} ETC[j][m]. \tag{1}$$

Having the values of the completion time for the machines, we can define the *local_makespan*, which is the makespan by considering only the machines involved in the current schedule:

$$local\_makespan = \max\{completion[i] \mid i \in Machines'\}. \tag{2}$$

Then, we define:

$$local\_avg\_utilisation = \frac{\sum_{\{i \in Machines\}} completion[i]}{local\_makespan \cdot nb\_machines}. \tag{3}$$

The *matching proximity* parameter is also very useful for measuring the performance of the presented methods. Matching proximity indicates the degree of proximity of a given schedule with regard to the schedule produced by MET method (see later). A large value of matching proximity means that a large number of jobs is assigned to the machine that executes them faster. Formally, this parameter is defined as follows:

$$matching\_proximity = \frac{\sum_{i \in Jobs} ETC[i][schedule[i]]}{\sum_{i \in Jobs} ETC[i][MET[i]]}. \qquad (4)$$

It should be noted that these parameters are among the most important parameters of a grid system. Makespan measures the productivity (throughput) of the grid system, the flowtime measures the QoS of the grid system and resource utilisation indicates the quality of a schedule with respect to the utilisation of resources involved in the schedule aiming to reduce the idle time of resources. Resource utilisation is especially interesting when resource owners' benefit is concerned.

## 3 Immediate mode methods

In this work five immediate mode methods, namely, OLB, MCT, MET, SA and *k*PB are implemented and empirically evaluated.

### 3.1 Opportunistic load balancing

This method assigns a job to the earliest idle machine without taking into account the execution time of the job in the machine. If two or more machines are available at the same time, one of them is arbitrarily chosen. Usually this method is used in *scavenging grids*. One advantage of this method is that it tries to keep the machines as loaded as possible; however, the method is not aware of the execution times of jobs in machines, which is, certainly, a disadvantage regarding the makespan, flowtime and matching proximity parameters.

### 3.2 Minimum completion time

This method assigns a job to the machine yielding the earliest completion time (the ready times of the machines are used). When a job arrives in the system, all available resources are examined to determine the resource that yields the smallest completion time for the job. Note that a job could be assigned to a machine that does not have the smallest execution time for that job. This method is also known as Fast Greedy, originally proposed for SmartNet system (Freund *et al.*, 1998).

### 3.3 Minimum execution time

This method assigns a job to the machine having the smallest execution time for that job. Unlike MCT method, MET does not take into account the ready times of machines. Clearly, in grid systems having resources of different computing capacity, this method could produce an unbalance by assigning jobs to fastest resources.

However, the advantage is that jobs are allocated to resources that best fit them with regard to the execution time. This method is also known in the literature as Limited Best Assignment (LBA) and User Directed Assignment (UDA).

### *3.4 Switching algorithm*

This method tries to overcome some limitations of MET and MCT methods by combining their best features. More precisely, MET is not good for load balancing while MCT does not take into account execution times of jobs into machines. Essentially, the idea is to use MET till a threshold is reached and then use MCT to achieve a good load balancing. Thus, SA method combines MET and MCT cyclically based on the workload of resources.

In order to implement the method, let $r_{\max}$ be the maximum ready time and $r_{\min}$ the minimum ready time; the load balancing factor is then $r_{\min}/r_{\max}$, which takes values in [0, 1]. Note that for $r = 1.0$ we have a perfect load balancing and if $r = 0.0$ then there exists at least one idle machine. Further, we use two threshold values $r_l$ (low) and $r_h$ (high) for $r$, $0 < r_l < r_h < 1$. Initially, $r = 0.0$ so that SA starts allocating jobs according to MCT until $r$ becomes greater than $r_h$; after that, MET is activated so that $r$ becomes smaller than $r_l$ and a new cycle starts again until all jobs are allocated.

### *3.5 k-Percent best*

For a given job, this method considers a subset of candidate resources from which the resource to allocate the job is chosen. For a given job, the candidate set consists of $nb\_machines \cdot k/100$ best resources (with respect to execution times), where $k$, $nbmachines/100 \leq k \leq 100$. The machine where to allocate the job is the one from the candidate set yielding the earliest completion time. Note that for $k = 100$, *k*PB behaves as MCT and for $k = 100/nb\_machines$ it behaves as MET. It should be noted that this method could perform poorly if the subset of resources is not within $k\%$ best resources for none of jobs implying thus a large idle time.

Notice also that both *k*PB and SA combine the best features of MCT and MET, however, only *k*PB tries to simultaneously optimise the objectives of MCT and MET while assigning a job to a machine.

## 4 Computational results

In this section we present computational results obtained with the implementations of the immediate mode methods using a benchmark of instances by Braun *et al.* (2001) for distributed heterogenous systems. The simulation model of Braun *et al.* allows us to establish a fair comparison of the presented methods. Moreover, different grid scenarios can be considered in this model by combining different characteristics of the grid systems such as computing consistency, heterogeneity of resources and jobs.

The instances of this benchmark are classified into 12 different types of ETC matrices, each of them consisting of 100 instances, according to three parameters: job heterogeneity, machine heterogeneity and consistency. Instances are labelled as *u_x_yyzz.k* where:

- *u* means uniform distribution (used in generating the matrix).
- *x* means the type of consistency (*c*-consistent, *i*-inconsistent and *s* means semi-consistent). An ETC matrix is considered consistent when, if a machine $m_i$ executes job *j* faster than machine $m_j$, then $m_i$ executes all the jobs faster than $m_j$. Inconsistency means that a machine is faster for some jobs and slower for some others. An ETC matrix is considered semi-consistent if it contains a consistent submatrix.
- *yy* indicates the heterogeneity of the jobs (*hi* means high and *lo* means low).
- *zz* indicates the heterogeneity of the resources (*hi* means high and *lo* means low).

Note that all instances consist of 512 jobs and 16 machines. For each method we compute the makespan, flowtime, resource utilisation and matching proximity. We present computational results for a set of 12 instances of the Braun *et al.* benchmark consisting of three groups having consistent, semi-consistent and inconsistent ETC matrices, respectively. Moreover, for each group, instances have different types of heterogeneity of jobs and heterogeneity of resources.

## *4.1 Results for makespan value*

We give in Table 1 computation results for makespan parameter and in Figures 1, 2, 3, 4 and 5 their graphical representation. In Figures 1, 2 and 3 the makespan value is given according to the computing consistency of considered instances and in Figures 4 and 5 according to the heterogeneity of resources and jobs.

**Table 1** Makespan values obtained with immediate mode methods (in arbitrary time units)

| *Instance* | *OLB* | *MCT* | *MET* | *SA* ($r_l = 0.6, r_h = 0.9$) | *kPB* (*k* = 20%) |
|---|---|---|---|---|---|
| u_c_hihi.0 | 14 376 662.175 | 11 422 624.494 | 47 472 299.429 | 12 613 221.101 | 12 496 863.706 |
| u_c_hilo.0 | 221 051.823 | 185 887.404 | 1 185 092.968 | 194 549.794 | 201 153.956 |
| u_c_lohi.0 | 477 357.019 | 378 303.624 | 1 453 098.003 | 426 271.390 | 400 291.050 |
| u_c_lolo.0 | 7306.595 | 6360.054 | 39 582.297 | 8167.052 | 6846.273 |
| u_i_hihi.0 | 26 102 017.618 | 4 413 582.982 | 4 508 506.791 | 4 692 192.006 | 4 508 655.928 |
| u_i_hilo.0 | 272 785.200 | 94 855.913 | 96 610.481 | 102 980.982 | 93 005.897 |
| u_i_lohi.0 | 833 605.654 | 143 816.093 | 185 694.594 | 143 905.246 | 143 816.093 |
| u_i_lolo.0 | 89 380.269 | 3137.350 | 3399.284 | 3485.290 | 3122.956 |
| u_s_hihi.0 | 19 464 875.910 | 6 693 923.896 | 25 162 058.136 | 7 127 729.951 | 6 514 162.148 |
| u_s_hilo.0 | 250 362.113 | 126 587.591 | 605 363.772 | 149 050.289 | 123 543.792 |
| u_s_lohi.0 | 603 231.467 | 186 151.286 | 674 689.535 | 194 318.366 | 187 955.955 |
| u_s_lolo.0 | 8938.389 | 4436.117 | 21 042.413 | 5836.962 | 4405.247 |

**Figure 1** Graphical representation of makespan values for consistent ETC matrices ('u_c_hihi' instance (left) and the rest of 'u_c_xxyy' instances (right))

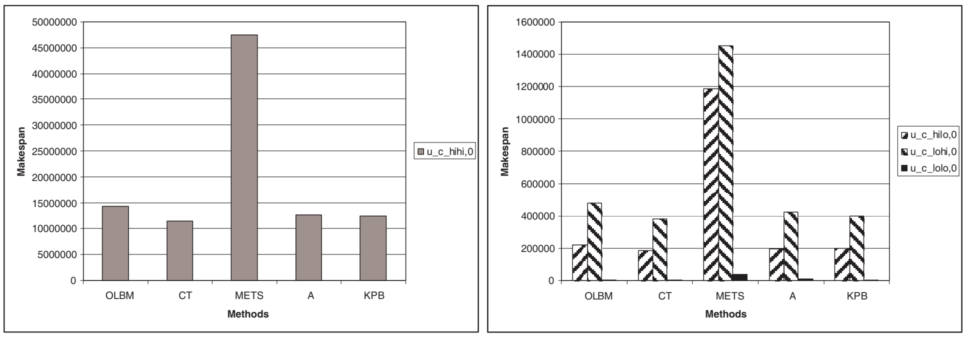


**Figure 2** Graphical representation of makespan values for inconsistent ETC matrices ('u_i_hihi' instance (left) and the rest of 'u_i_xxyy' instances (right))

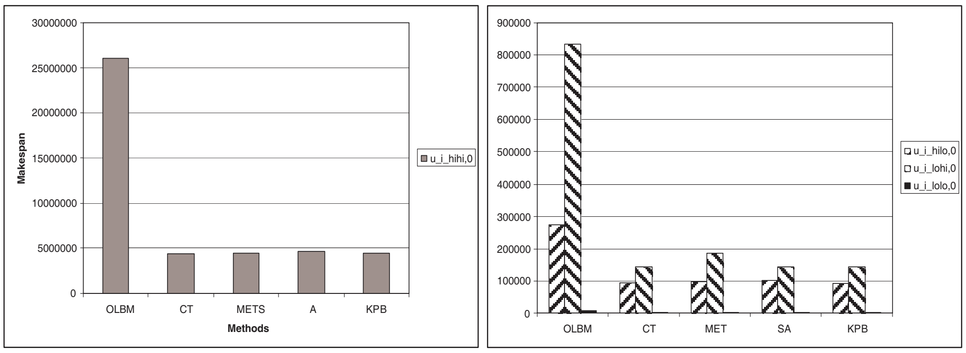


**Figure 3** Graphical representation of makespan values for semi-consistent ETC matrices ('u_s_hihi' instance (left) and the rest of 'u_s_xxyy' instances (right))

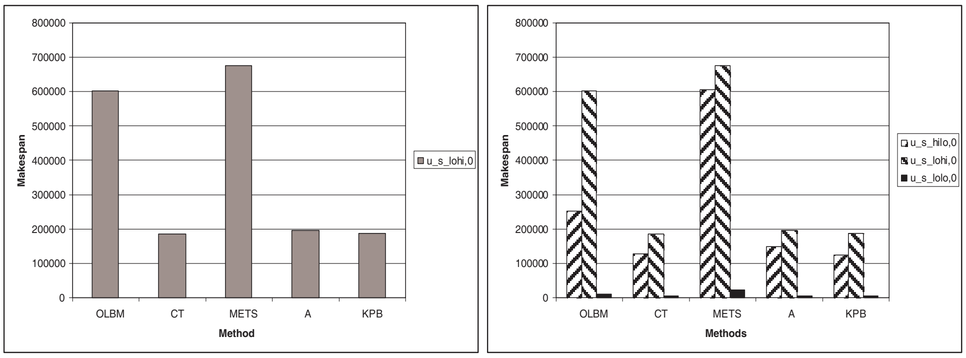
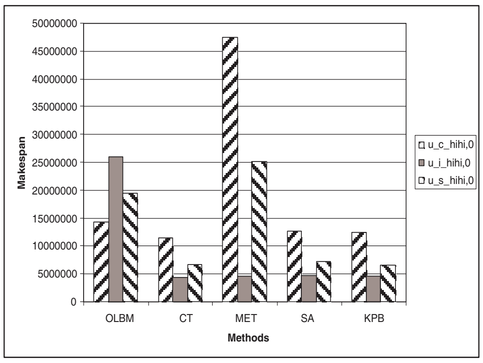


**Figure 4** Graphical representation of makespan values for 'hihi' instances (left) and 'lolo' instances (right)



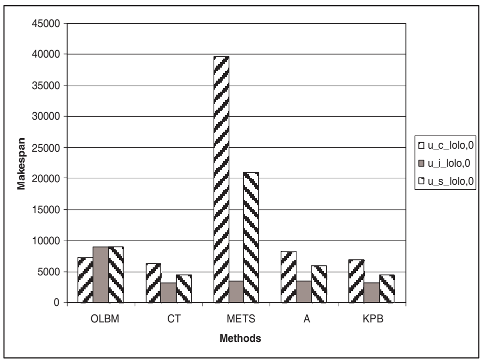
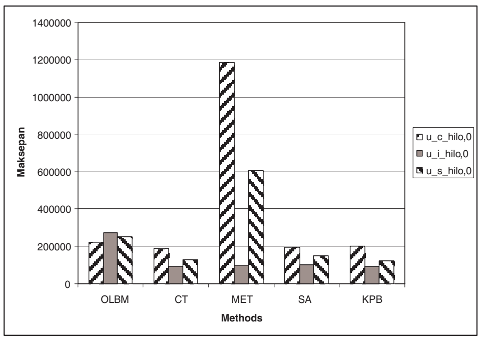
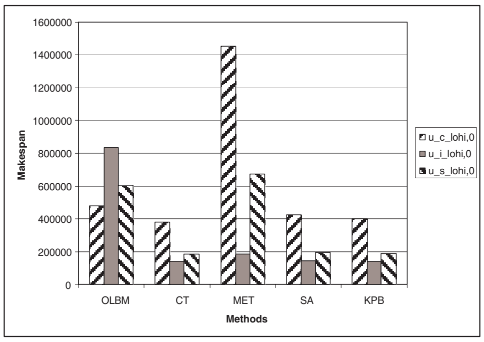


**Figure 5** Graphical representation of makespan values for 'hilo' instances (left) and 'lohi' instances (right)





## *4.2 Results for flowtime value*

We give in Tables 2 and 3 computation results for flowtime parameter and their graphical representation in Figures 6, 7, 8, 9 and 10. Again, in Figures 6, 7 and 8 the flowtime value is given according to the computing consistency of considered instances and in Figures 9 and 10 according to the heterogeneity of resources and jobs.

**Table 2** Flowtime values obtained with immediate mode methods (in arbitrary time units)

| *Instance* | *OLB* | *MCT* | *MET* |
|---|---|---|---|
| u_c_hihi.0 | 1 995 375 910.590 | 1 815 882 056.432 | 4 844 329 725.992 |
| u_c_hilo.0 | 35 915 700.026 | 35 394 474.558 | 195 784 905.697 |
| u_c_lohi.0 | 66 414 109.547 | 65 625 471.648 | 156 597 212.675 |
| u_c_lolo.0 | 1 189 657.523 | 1 193 304.135 | 6 399 549.478 |
| u_i_hihi.0 | 3 578 545 420.915 | 591 703 697.305 | 369 819 381.995 |
| u_i_hilo.0 | 40 168 914.127 | 16 309 961.615 | 12 778 823.342 |
| u_i_lohi.0 | 112 209 075.410 | 18 954 137.704 | 12 686 655.759 |
| u_i_lolo.0 | 1 380 271.657 | 545 326.248 | 443 305.108 |
| u_s_hihi.0 | 2 711 172 017.693 | 964 163 417.887 | 1 559 122 316.924 |
| u_s_hilo.0 | 36 996 922.147 | 22 357 619.927 | 57 134 246.084 |
| u_s_lohi.0 | 83 583 290.767 | 27 064 495.376 | 43 054 529.545 |
| u_s_lolo.0 | 1 395 727.226 | 797 285.039 | 1 973 399.444 |

**Table 3** Flowtime values obtained with immediate mode methods (in arbitrary time units) (continuation of results in Table 2)

| *Instance* | *SA ($r_l$ = 0.6, $r_h$ = 0.9)* | *kPB (k = 20%)* |
|---|---|---|
| u_c_hihi.0 | 1 846 422 579.779 | 2 002 538 268.654 |
| u_c_hilo.0 | 34 901 832.621 | 37 220 460.128 |
| u_c_lohi.0 | 63 250 093.468 | 66 422 481.844 |
| u_c_lolo.0 | 1 181 075.902 | 1 255 064.070 |
| u_i_hihi.0 | 557 457 024.518 | 596 857 288.709 |
| u_i_hilo.0 | 15 027 765.500 | 16 093 172.399 |
| u_i_lohi.0 | 18 900 796.619 | 18 552 145.175 |
| u_i_lolo.0 | 498 180.120 | 548 765.218 |
| u_s_hihi.0 | 949 405 775.550 | 907 398 445.237 |
| u_s_hilo.0 | 21 369 485.252 | 22 056 448.264 |
| u_s_lohi.0 | 25 883 799.205 | 27 266 722.451 |
| u_s_lolo.0 | 791 072.307 | 780 609.157 |

**Figure 6** Graphical representation of flowtime values for consistent ETC matrices ('u_c_hihi' instance (left) and the rest of 'u_c_xxyy' instances (right))

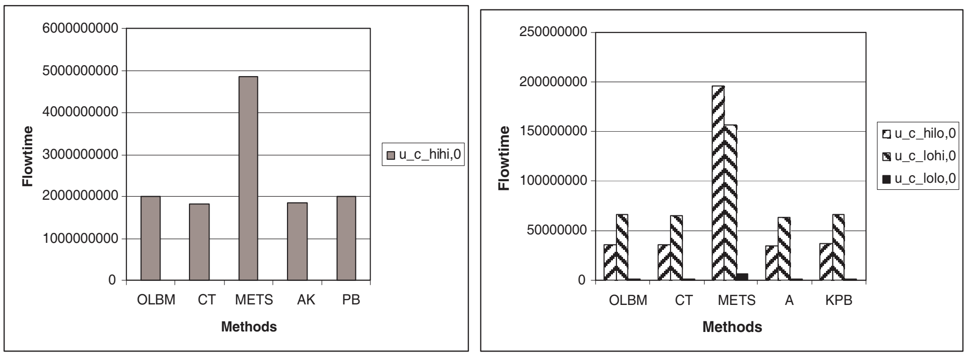


**Figure 7** Graphical representation of flowtime values for inconsistent ETC matrices ('u_i_hihi' instance (left) and the rest of 'u_i_xxyy' instances (right))

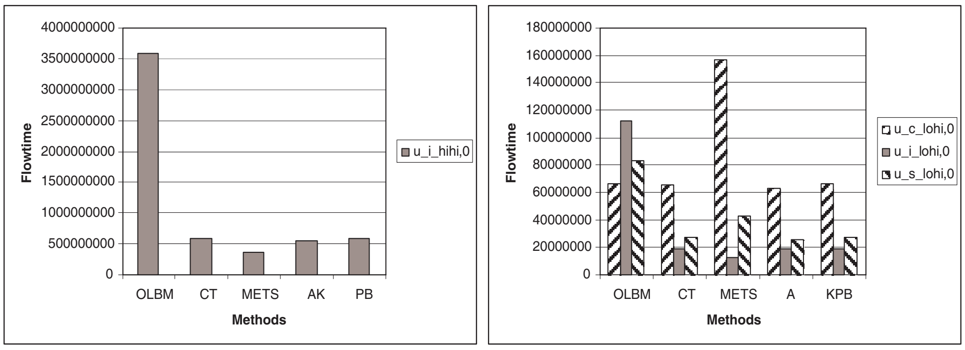


**Figure 8** Graphical representation of flowtime values for semi-consistent ETC matrices ('u_s_hihi' instance (left) and the rest of 'u_s_xxyy' instances (right))

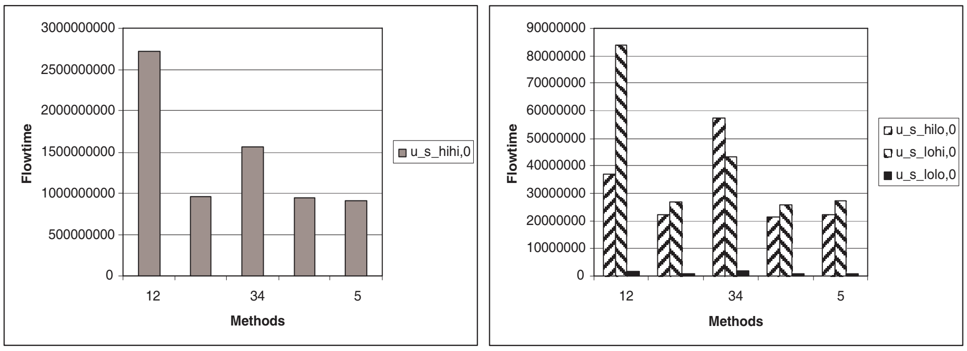


**Figure 9** Graphical representation of flowtime values for 'hihi' instances (left) and 'lolo' instances (right)

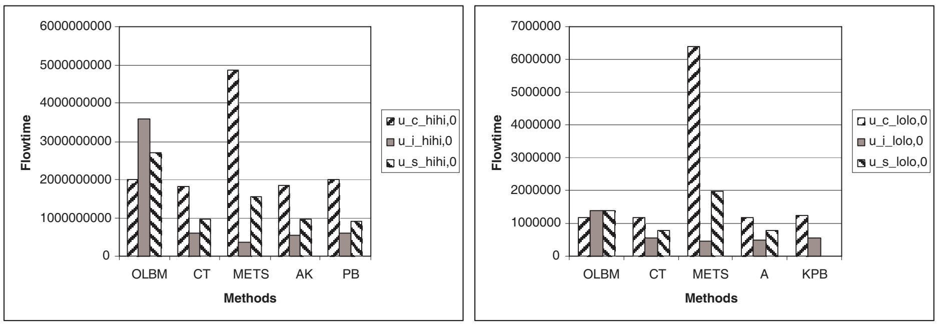
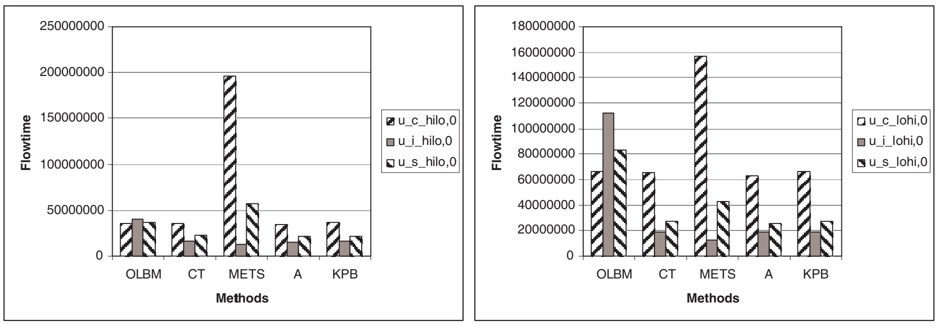


**Figure 10** Graphical representation of flowtime values for 'hilo' instances (left) and 'lohi' instances (right)



## *4.3 Results for resource utilisation and matching proximity*

We give in Tables 4 and 5 computational results for resource utilisation and matching proximity, respectively. Their graphical representation is shown in Figure 11.

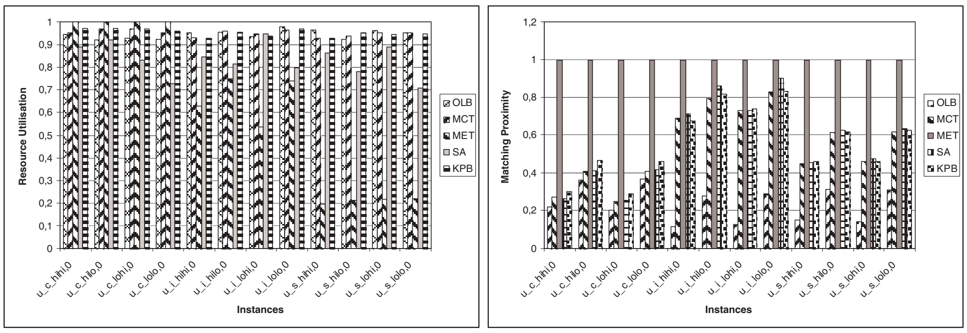
**Table 4** Average resource utilisation values obtained with immediate mode methods (see Equation 3)

| | | | | *SA* | *kPB* |
|---|---|---|---|---|---|
| *Instance* | *OLB* | *MCT* | *MET* | *($r_l$ = 0.6, $r_h$ = 0.9)* | *(k = 20%)* |
| u_c_hihi.0 | 0.946 | 0.953 | 1.000 | 0.890 | 0.972 |
| u_c_hilo.0 | 0.920 | 0.970 | 1.000 | 0.920 | 0.974 |
| u_c_lohi.0 | 0.928 | 0.969 | 1.000 | 0.832 | 0.969 |
| u_c_lolo.0 | 0.923 | 0.951 | 1.000 | 0.727 | 0.960 |
| u_i_hihi.0 | 0.951 | 0.932 | 0.628 | 0.846 | 0.929 |
| u_i_hilo.0 | 0.955 | 0.959 | 0.750 | 0.816 | 0.954 |
| u_i_lohi.0 | 0.934 | 0.949 | 0.536 | 0.948 | 0.937 |
| u_i_lolo.0 | 0.979 | 0.965 | 0.740 | 0.797 | 0.968 |
| u_s_hihi.0 | 0.967 | 0.928 | 0.197 | 0.863 | 0.928 |
| u_s_hilo.0 | 0.924 | 0.938 | 0.214 | 0.781 | 0.951 |
| u_s_lohi.0 | 0.961 | 0.953 | 0.216 | 0.891 | 0.946 |
| u_s_lolo.0 | 0.951 | 0.951 | 0.221 | 0.708 | 0.950 |

**Table 5** Matching proximity values obtained with immediate mode methods (see Equation 4)

| | | | | *SA* | *kPB* |
|---|---|---|---|---|---|
| *Instance* | *OLB* | *MCT* | *MET* | *($r_l$ = 0.6, $r_h$ = 0.9)* | *(k = 20%)* |
| u_c_hihi.0 | 0.217 | 0.272 | 1.000 | 0.264 | 0.300 |
| u_c_hilo.0 | 0.364 | 0.410 | 1.000 | 0.413 | 0.464 |
| u_c_lohi.0 | 0.204 | 0.247 | 1.000 | 0.255 | 0.287 |
| u_c_lolo.0 | 0.366 | 0.408 | 1.000 | 0.416 | 0.463 |
| u_i_hihi.0 | 0.114 | 0.688 | 1.000 | 0.713 | 0.675 |
| u_i_hilo.0 | 0.278 | 0.796 | 1.000 | 0.862 | 0.816 |
| u_i_lohi.0 | 0.127 | 0.729 | 1.000 | 0.730 | 0.739 |
| u_i_lolo.0 | 0.287 | 0.830 | 1.000 | 0.905 | 0.831 |
| u_s_hihi.0 | 0.148 | 0.448 | 1.000 | 0.453 | 0.461 |
| u_s_hilo.0 | 0.315 | 0.614 | 1.000 | 0.626 | 0.620 |
| u_s_lohi.0 | 0.141 | 0.463 | 1.000 | 0.474 | 0.462 |
| u_s_lolo.0 | 0.308 | 0.620 | 1.000 | 0.633 | 0.625 |

**Figure 11** Graphical representation of resource utilisation values (left) and matching proximity (right)



## 5 Performance evaluation

In this section we evaluate the computational results obtained from immediate mode methods regarding four parameters: makespan, flowtime, resource utilisation and matching proximity. As can be seen from the description of the considered methods, they use concrete strategies therefore the objective is to evaluate their performance, from which we could deduce which method to use for certain grid systems characteristics (configurations), especially if we knew such characteristics in advance.

From Tables 1 to 5 and Figures 1 to 11, we can easily draw the following conclusions:

- Makespan

  The methods that best perform regarding the makespan are MCT and *k*PB (see Table 1 and Figures 1 to 5). Moreover, a careful examination of the results show that MCT obtains better results for consistent matrices (see Figure 1) while *k*PB performs better for semi-consistent and inconsistent matrices (see Figures 2 and 3). SA also shows a good performance, except for inconsistent matrices. MET shows a good performance for inconsistent matrices and poor behaviour for consistent and semi-consistent matrices. On the other hand, for 'hihi' and 'lolo' instances, MCT, *k*PB and SA perform good while MET performs quite poorly. As can be seen in Figure 4, the combinations 'i_hihi' and 'i_lolo', that is, instances of inconsistent ETC matrices with either high heterogeneity of resources and jobs or low heterogeneity of resources and jobs are the best for MET method.

- Flowtime

  The performance of these methods regarding this parameter are not as good as for makespan (see Tables 2 and 3 and Figures 6 to 10); rather small reductions of flowtime are obtained. For consistent matrices SA performs better followed by MCT. For semi-consistent matrices the best performance is shown by SA and *k*PB while for inconsistent matrices MET outperforms all the other methods. OLB shows a poor performance. Regarding the combinations 'hi' and 'lo', MCT, SA and *k*PB perform best. Regarding MET performs best for the combination 'i_hihi' and 'i_lolo' and very poorly for the rest of combinations.

- Average resource utilisation

  The MET method obtains the worse resource utilisation overall, except for consistent ETC matrices for which, as expected, resource utilisation is 1.0. *k*PB performs better for consistent matrices, however, MCT and OLB achieve a better load balancing for inconsistent matrices. Overall, OLB is the method that performs best, in particular, for semi-consistent matrices. Nonetheless, MCT and *k*PB behave quite similarly. Observe also that though OLB achieves very good resource utilisation, its makespan values are not good. Thus, we could say that for very heterogenous systems, resource utilisation is not a good indicator for makespan. Finally, SA performs rather poorly (see Table 4 and Figure 11 (left)).

- Matching proximity

  Except for MET (which shows perfect proximity matching), the best matching proximity is obtained by *k*PB and SA. In particular, *k*PB performs better for consistent matrices and SA performs better for semi and inconsistent matrices. MCT performs poorly and OLB shows the worst matching proximity values (see Table 5 and Figure 11 (right)).

## 6 Immediate scheduling in web and grid services

Grid systems are expected to provide a large variety of complex services (Venugopal *et al.*, 2004) whose interactions require scheduling and resource allocation policies. At present, coordinated scheduling of services in grid systems is not yet available despite the existence of several middleware (*e.g.*, Globus Toolkit)[2] or brokerage examples (*e.g.*, Nimrod/G Resource Broker[3] Buyya *et al.* (2000)).

One approach to scheduling services is that of decentralised schedulers or brokers that integrate local schedulers (aka distributed scheduling components), which are in charge for scheduling user jobs to idle computers in a LAN (for instance, Nimrod/G Resource Broker system incorporates such a component). In this context, immediate mode schedulers presented here are appropriate for local schedulers and could be thus integrated in a more complex scheduler system as well as with higher-level scheduling services. Indeed, as shown by the study of this work, immediate mode methods explore the characteristics of the grid system and thus if we knew in advance some of these characteristics, for instance the degree of heterogeneity of jobs and resources or the degree of consistency we could adaptively choose the appropriate immediate method. For instance, this is possible for LANs since we could know in advance the characteristics of the LAN at the institution or enterprise. Grid scheduling architectures then should support cooperation between different scheduling instances. We could think also of job submission service responsible for managing jobs: it receives a job request from users/applications, requests scheduling to a grid scheduling service and requests job execution to local scheduler; this last could be immediate mode-based scheduler since this sort of schedulers are especially efficient for LANs – due to their small or moderate size – and for sites with high throughput.

Immediate mode schedulers could be also useful in scheduling of workflow applications in grids (Yu *et al.*, 2005) since in such applications the workflow scheduler must be able to adapt and update the schedule based on resource dynamics.

Finally, the presented immediate mode scheduling are useful to improve the efficiency of web services. In particular, methods such as OLB (Harchol-Balter *et al.* (2003) uses the Shortest Remaining Processing Time First method) are interesting since load balancing is indispensable for a web service system to assure even distribution of incoming requests on the web servers.

All in all, given that for a large family of grid applications requests for resources could be immediate, the presented immediate mode methods are potentially useful in scheduling jobs originated by such applications.

## 7 Conclusion and future work

In this work we have presented a family of methods for dynamic scheduling of independent jobs in CGs, known as immediate mode scheduling methods. The main characteristic of these methods is that they schedule a job as soon as it arrives in the system. The following five methods: OLB, MCT, MET, SA and *k*PB have been presented and empirically studied by using the benchmark by Braun *et al.* (2001). The computational results show that none of the methods performs best; rather, their performance depends on the grid scenarios defined by the heterogeneity of the jobs (high, low), heterogeneity of the resources (high, low) and consistency (consistent, inconsistent and semi-consistent) of computing. It should be noted that consistency allow to simulate real grid environments in which restrictions job-resource could exist.

By disposing implementations of such a variety of methods, we could choose and apply the appropriate method if we knew in advance some of the grid characteristics or if we are interested in a concrete parameter (makespan, flowtime, resource utilisation or matching proximity). Also, we observe that these methods are very interesting for web and grid scheduling services to achieve efficiency, load balancing and QoS.

We are currently testing a discrete-event grid simulator based on HyperSim package Phatanapherom and Kachitvichyanukul (2003) that we will use to study the performance of the presented methods and address the issue of experimenting in a dynamic grid environment.

## Acknowledgement

This research is partially supported by Projects ASCE TIN2005-09198-C02-02, FP6-2004-ISO-FETPI (AEOLUS) and MEC TIN2005-25859-E.

## References

Abraham, A., Buyya, R. and Nath, B. (2000) 'Nature's heuristics for scheduling jobs on computational grids', *The 8th IEEE Int. Conf. on Advanced Computing and Communications (ADCOM 2000)*, India.

Braun, T.D., Siegel, H.J., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D. and Yao, B. (2001) 'A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems', *J. of Parallel and Distr. Comp.*, Vol. 61, No. 6, pp.810–837.

Buyya, R. (2002) 'Economic-based distributed resource management and scheduling for grid computing', PhD thesis, Monash University, Melbourne, Australia.

Buyya, R., Abramson, D. and Giddy, J. (2000) 'Nimrod/G: an architecture for a resource management and scheduling system in a global computational grid', *The 4th Int. Conf. on High Performance Computing*, Asia-Pacific Region, China.

Buyya, R., Abramson, D., Giddy, J. and Stockinger, H. (2002) 'Economic models for resource management and scheduling in grid computing', *Concurrency and Computation: Practice and Experience*, Vol. 14, Nos. 13–15, pp.1507–1542.

Cardellini, V., Colajanni, M. and Yu, P.S. (1999) 'Dynamic load balancing on web-server systems', *Internet Computing*, IEEE, Vol. 3, No. 3, pp.28–39.

Carretero, J. and Xhafa, F. (2006) 'Using genetic algorithms for scheduling jobs in large scale grid applications', *Journal of Technological and Economic Development – A Research Journal of Vilnius Gediminas Technical University*, Vol. 12, No. 1, pp.11–17.

Casanova, H. and Dongarra, J. (1998) 'Netsolve: network enabled solvers', *IEEE Computational Science and Engineering*, Vol. 5, No. 3, pp.57–67.

Casanova, H., Kim, M., Plank, J.S. and Dongarra, J. (1999) 'Adaptive scheduling for task farming with grid middleware', *International Journal of High Performance Computing*, Sage Science Press, Vol. 13, No. 3, pp.231–240.

Casanova, H., Legrand, A., Zagorodnov, D. and Berman, F. (2000) 'Heuristics for scheduling parameter sweep applications in grid environments', *Heterogeneous Computing Workshop*, pp.349–363.

Dail, H.J. (2002) 'A modular framework for adaptive scheduling in grid application development environments', Master thesis, University of California, San Diego.

Dail, H.J., Berman, F. and Casanova, H. (2003) 'A decoupled scheduling approach for grid application development environments', *Journal of Parallel and Distributed Computing*, Vol. 63, No. 5, pp.505–524.

Di Martino, V. and Mililotti, M. (2004) 'Sub optimal scheduling in a grid using genetic algorithms', *Parallel Computing*, Vol. 30, pp.553–565.

Foster, I. and Kesselman, C. (1998) *The Grid – Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers.

Foster, I., Kesselman, C. and Tuecke, S. (2001) 'The anatomy of the grid', *International Journal of Supercomputer Applications*, Vol. 15, No. 3.

Frattolillo, F. (2005) 'Running large-scale applications on cluster grids', *International Journal High Performance Computing and Applications*, USA: Sage Publications, Inc., Vol. 19, No. 2, pp.157–172.

Freund, R., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D., Keith, E., *et al.* (1998) 'Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet', *Seventh Heterogeneous Computing Workshop*, pp.184–199.

Hamscher, V., Schwiegelshohn, U., Streit, A. and Yahyapour, R. (2000) 'Evaluation of job-scheduling strategies for grid computing', *Proceedings of the First IEEE/ACM International Workshop on Grid Computing*, pp.191–202.

Harchol-Balter, M., Schroeder, B., Bansal, N. and Agrawal, M. (2003) 'Size-based scheduling to improve web performance', *ACM Transactions on Computing Systems*, ACM Press, Vol. 21, No. 2, pp.207–233.

Karatza, H.D. (2004) 'Scheduling in distributed systems', *Performance Tools and Applications to Networked Systems: MASCOTS 2003 Revised Tutorial Lectures*, Lecture Notes in Computer Science, Springer, Vol. 965, pp.336–356.

Kranzlmuller, D., Rosmanith, H., Heinzlreiter, P. and Polak, M. (2004) 'Interactive virtual reality on the grid', *Proceedings of the Eighth IEEE International Symposium on Distributed Simulation and Real-time Applications (DS-RT'04)*, IEEE Computer Society, pp.152–158.

Linderoth, L. and Wright, S.J. (2003) 'Decomposition algorithms for stochastic programming on a computational grid', *Computational Optimization and Applications*, Vol. 24, pp.207–250.

Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D. and Freund, R.F. (1999) 'Dynamic mapping of a class of independent tasks onto heterogeneous computing systems', *J. of Parallel and Distr. Comp.*, Vol. 59, No. 2, pp.107–131.

Nefedova, V., Jacob, R., Foster, I., Liu, Zh., Liu, Y., Deelman, E., Mehta, G., Su, M. and Vahi, K. (2006) 'Automating climate science: large ensemble simulations on the TeraGrid with the GriPhyN virtual data system', *Proceedings of e-Science Conference 2006*, Amsterdam, 4–6 December.

Nikolopoulos, D.S. and Polychronopoulos, C.D. (2003) 'Adaptive scheduling under memory constraints on non-dedicated computationalfarms', *Future Generation Computer Systems*, Vol. 19, No. 4, pp.505–519.

Phatanapherom, S. and Kachitvichyanukul, V. (2003) 'Fast simulation model for grid scheduling using hypersim', *Proceedings of the 2003 Winter Simulation Conference*, New Orleans, USA, December.

Sun, Y., Wipat, A., Pocock, M.R., Lee, P.A., Watson, P., Flanagan, K. and Worthington, J.T. (2005) 'A grid-based system for microbial genome comparison and analysis', *Proceedings of 5th International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, IEEE Computer Society, pp.977–984.

Venugopal, S., Buyya, R. and Winton, L.J. (2004) 'A grid service broker for scheduling distributed data-oriented applications on global grids', *Middleware for Grid Computing: Proceedings of the 2nd Workshop on Middleware for Grid Computing*, ACM, Toronto, Ontario, Canada, 18–22 October, pp.75–80.

Wright, S.J. (2001) 'Solving optimization problems on computational grids', *Optima*, Vol. 65.

Wu, M. and Shu, W. (2001) 'A high-performance mapping algorithm for heterogeneous computing systems', *Proceedings of the 15th International Parallel & Distributed Processing Symposium*, p.74.

Xhafa, F. (2006) 'An experimental study on GA replacement operators for scheduling on grids', *The 2nd International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2006)*, Ljubljana, Slovenia, 9–10 October, pp.121–130.

Yu, J., Buyya, R. and Tham, C.K. (2005) 'QoS-based scheduling of workflow applications on service grids', *Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing*, IEEE, Melbourne, Australia.

Zaia, A., Bruneo, D. and Puliafito, A. (2006) 'Using the grid paradigm for multimedia applications: research articles', *Concurrency and Computation: Practice & Experience*, John Wiley and Sons Ltd., Vol. 18, No. 8, pp.899–910.

Zhang, L. (2002) 'Scheduling algorithm for real-time applications in grid environment systems', *IEEE International Conference on Man and Cybernetics*, Vol. 5.

Zhang, Sh., Lu, Zh. and Wu, J. (2005) 'mdGrid: a novel broadband multimedia content delivery service grid model', *Proceedings of the Ninth IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA 2005*, ACTA Press, pp.389–393.

## Notes

1 http://www.globus.org/ogsa/

2 http://www.globus.org/toolkit/

3 http://www.csse.monash.edu.au/~davida/nimrod/