Future Generation Computer Systems 28 (2012) 965–976

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Adaptive parallel job scheduling with resource admissible allocation on two-level hierarchical grids

Ariel Quezada-Pina [a], Andrei Tchernykh [a,*], José Luis González-García [e], Adán Hirales-Carbajal [a,b], Juan Manuel Ramírez-Alcaraz [c], Uwe Schwiegelshohn [d], Ramin Yahyapour [e], Vanessa Miranda-López [a]

[a] Computer Science Department, CICESE Research Center, Ensenada, BC 22860, Mexico
[b] Science Faculty, Autonomous University of Baja California, Ensenada, B.C., Mexico
[c] Colima University, C.P. 28040. Colima, Col., Mexico
[d] Robotics Research Institute, Technische Universität Dortmund, 44221 Dortmund, Germany
[e] GWDG – University of Göttingen, 37077 Göttingen, Germany

ARTICLE INFO

Article history:
Received 13 May 2011
Received in revised form
2 February 2012
Accepted 6 February 2012
Available online 18 February 2012

Keywords:
Grid
Online scheduling
Resource management
Job allocation
Admissible allocation
Hierarchical

ABSTRACT

We evaluate job scheduling algorithms that integrate both tasks of Grid scheduling: job allocation to Grid sites and local scheduling at the sites. We propose and analyze an adaptive job allocation scheme named admissible allocation. The main idea of this scheme is to set job allocation constraints, and dynamically adapt them to cope with different workloads and Grid properties. We present 3-approximation and 5-competitive algorithms named $MLB_a + PS$ and $MCT_a + PS$ for the case that all jobs fit to the smallest machine, while we derive an approximation factor of 9 and a competitive factor of 11 for the general case. To show practical applicability of our methods, we perform a comprehensive study of the practical performance of the proposed strategies and their derivatives using simulation. To this end, we use real workload traces and corresponding Grid configurations. We analyze nine scheduling strategies that require a different amount of information on three Grid scenarios. We demonstrate that our strategies perform well across ten metrics that reflect both user- and system-specific goals.

1. Introduction

Job scheduling is an important issue for achieving high performance on Grids. Various scheduling systems have already been proposed and implemented in different types of Grids. However, there are still many open issues in this field. One of the big challenges is the development of coordinated resource provisioning mechanisms that allow more efficient use of resources and higher user satisfaction.

In general, the problem of scheduling jobs on multiprocessors is well understood and has been subject to research for decades. Many research results exist for many different variants of this problem. Some of them provide theoretical insights while others give hints for the implementation of real systems.

However, scheduling in Grids is almost exclusively addressed by practitioners looking for suitable implementations. There are only very few theoretical results available on Grid scheduling which we will further discuss in Section 3.

In this paper, we consider a basic two layer Grid model. The highest layer consists of a Grid scheduler (resource broker or meta scheduler) that has a general view of job requests and allocates jobs to a suitable Grid site. The management of a specific resource is the task of a local resource management system that knows the present status of its machine and of the jobs that are allocated to it. Local scheduling is independently applied to each site. On each layer, different constraints and specifications are considered. Here, we assume rigid parallel jobs that have a given degree of parallelism and must be assigned exclusively to the specified number of processors or cores during their execution. We apply an admissible allocation policy that excludes certain machines from the set of machines available for allocation of a particular job [1]. In this paper, we conduct the theoretical analysis of two scheduling algorithms named Min-Lower-Bound with Admissible Job Allocation (MLB_a) and Min-Completion-Time with Admissible Job Allocation (MCT_a) under given constraints and makespan optimization criterion, that is, we consider minimization of largest completion time of any job in the system. Specifically, we show that the makespan obtained by our algorithms does not exceed the optimal makespan for any problem instances by more than

* Corresponding author. Tel.: +52 16461786994.
E-mail addresses: aquezada@cicese.mx (A. Quezada-Pina), chernykh@cicese.mx (A. Tchernykh), jose-luis.gonzalez-garcia@gwdg.de (J.L. González-García), ahirales@cicese.mx, ahirales@uabc.mx (A. Hirales-Carbajal), jmramir@ucol.mx (J.M. Ramírez-Alcaraz), uwe.schwiegelshohn@udo.edu (U. Schwiegelshohn), ramin.yahyapour@gwdg.de (R. Yahyapour).

0167-739X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.02.004

a certain factor. This factor is called a competitive factor in the on-line case, and an approximation factor in the off-line case, respectively.

An approximation factor of 9 for the off-line case and competitive factor of 11 for the on-line case are achieved, improving and extending known results [1]. We also derived an approximation factor of 3 and a competitive factor of 5 for cases when all jobs fit to the smallest machine as in the problem discussed by Bougeret et al. [2].

To show the practicability and competitiveness of our algorithms, we conduct a comprehensive study of their performance and derivatives using simulation. We take into account several issues that are critical for practical adoption of the scheduling algorithms: we use Grid workloads based on real production traces, consider three Grid scenarios based on heterogeneous HPC systems, and consider problems of scheduling jobs with unspecified execution time requirements as only job user run time estimates are available in real execution environment.

The algorithms target computationally intensive parallel applications and make scheduling decisions without precise performance prediction information, in particular, about the job run time. They are simple, proceed on a job-by-job basis, and allow an efficient implementation in real systems.

We continue this paper by formally presenting our Grid scheduling model in Section 2. We discuss related work in Section 3. We introduce hierarchical job scheduling algorithms and classify them in Section 4, where we also discuss job admissible allocation schemes. We derive their approximation and competitive factors for off-line and on-line cases, respectively, in Section 5. Experimental setups are presented in Section 6 while experimental results are analyzed in Section 7. Finally, we conclude with a summary and an outlook in Section 8.

2. Model

2.1. Basic definitions

We address an on-line scheduling problem: n parallel jobs $J_1, J_2, \ldots, J_n$ must be scheduled on m sites (parallel machines $N_1, N_2, \ldots, N_m$. Let m_i be the number of identical processors of machine N_i also called the size of machine N_i. Let $s_{f,l} = \sum_{i=f}^{l} m_i$ be the total number of processors belonging to machines from N_f to N_l. We assume, without loss of generality, that the parallel machines are arranged in non-descending order of their sizes $m_1 \leq m_2 \leq \cdots \leq m_m$.

Each job J_j is described by a tuple $(r_j, \text{size}_j, p_j, p'_j)$: its release date $r_j \geq 0$ that denotes time relative to the beginning of a schedule, its size $1 \leq \text{size}_j \leq m_m$ that is referred to as its processor requirements or degree of parallelism, execution time p_j, and user run time estimate p'_j.

In the on-line scenario, no job parameters are available before a job is submitted. The scheduling task is particularly difficult if the processing time of a job only becomes available at the completion time of this job. This is called a non-clairvoyant scenario. In some real systems, the user is required to provide an estimate of the processing time of his job to prevent faulty jobs from wasting compute resources. This estimate can also be used by the scheduler although the estimates may be rather imprecise. In this paper, we address both on-line scenarios.

Further, $w_j = p_j \cdot \text{size}_j$ is the work of job J_j, also called its area in the schedule or its resource consumption. The jobs are submitted over time and must be immediately and irrevocably allocated to a single machine. However, the processor allocation for a job can be delayed until the required number of processors is actually available. The job is executed in space sharing mode by exclusively allocating exactly size_j processors for an uninterrupted period of time p_j to it. As we do not allow pre-emption nor multi-site execution nor co-allocation of processors from different machines, a job J_j can only run on machine N_i if $\text{size}_j \leq m_i$ holds. Remember that the precise processing time and the precise work are not available to a scheduler in our scenarios.

We use $g_j = i$ to denote that job J_j is allocated to machine N_i, n_i to denote the number of jobs allocated to machine N_i, and $W_i = \sum_{g_k=i} w_k$ to denote the total work of jobs allocated to machine N_i.

The completion time of job J_j of instance I in a schedule S is denoted by $c_j(S, I)$. Formally, the makespan of a schedule S and instance I is $C_{\max}(S, I) = \max_{J_j} \{c_j(S, I)\}$. The optimal makespan of instance I is denoted by $C^*_{\max}(I)$. If it is possible without causing ambiguity we will omit the instance I and schedule S.

We denote our Grid machine model by GP_m. In the three field notation $(\alpha|\beta|\gamma)$ with machine environment (α), job characteristics (β), and objective function (γ), introduced by Graham et al. [3] for theoretical scheduling problems, our scheduling problem is characterized as $GP_m|r_j, \text{size}_j|C_{\max}$ for the $C_{\max}$ optimization criterion. Objective functions are based upon metrics discussed in Section 2.2. As in [1], we use the notation *MPS* (Multiple machine Parallel Scheduling) to refer to this problem, while the notation *PS* (Parallel Scheduling) describes the parallel job scheduling on a single parallel machine $P_m|r_j, \text{size}_j|C_{\max}$.

Different scheduling algorithms may be used by the Local Resource Management System (*LRMS*). In the experimental parts of this paper, we assume that the *LRMS* uses an on-line parallel scheduling algorithm: First-Come-First-Serve policy with *EASY* backfilling [4], where the scheduler may use later jobs to fill holes in the schedule, even if that delays the expected start time of other jobs, so long as the first job's expected start time is not delayed. In order to apply *EASY* backfilling, user estimated runtimes are used.

2.2. Metrics

Formally, the competitive factor of algorithm A is defined as $\rho_A = \max_I \frac{C_{\max}(S_A, I)}{C^*_{\max}(I)}$ taken over all possible problem instances. Again, we omit algorithm A, if it does not cause any ambiguity. The approximation factor is defined similarly for deterministic (off-line) scheduling problems. Note that in our deterministic scheduling, all jobs are available at the start of the schedule and the scheduler knows the parameters of all jobs. We assume that the resources involved are stable and dedicated to the Grid. We also consider well known performance metrics commonly used to express the objectives of different stakeholders of Grid scheduling (end-users, local resource providers, and Grid administrators), such as: mean slowdown $SD_b = \frac{1}{n} \sum_{j=1}^{n} \frac{c_j - r_j}{\max\{10, p_j\}}$, and mean waiting time $t_w = \frac{1}{n} \sum_{j=1}^{n} (c_j - p_j - r_j)$.

To eliminate the emphasis on very short jobs (i.e., with close to zero runtime) the slowdown is bounded by a commonly used threshold of 10 s.

In this paper, we use t_w over turnaround (response) time $TA = \frac{1}{n} \sum_{j=1}^{n} (c_j - r_j)$, and sum of job waiting times $SWT = \sum_{j=1}^{n} (c_j - p_j - r_j)$. The differences between these metrics are constants regardless of the scheduler being used: $\frac{1}{n} \sum_{j=1}^{n} p_j$, and $1/n$. We do not use resource-centric metrics like utilization $U = \sum_{j=1}^{n} \frac{p_j \cdot \text{size}_j}{C_{\max} \cdot s_{1,m}}$ and throughput $Th = \frac{n}{C_{\max}}$. There is a close relationship between them: as $C^*_{\max}$, $\sum_{j=1}^{n} \frac{p_j \cdot \text{size}_j}{s_{1,m}}$, and n being constants for a given experiment; x percent reduction in $C_{\max}$ corresponds to $\frac{x}{100\% - x}$ percent increase in the utilization and throughput [5].

Note that in experimental analysis, the lower bound of the optimal completion time $\hat{C}^*_{\max} = \max \left\{ \max_j (r_j + p_j), \frac{\sum_{j=1}^{n} w_j}{s_{1,m}} \right\}$ is used instead of the optimal makespan $C^*_{\max}$ to calculate the competitive factor [6].

3. Related work

Job scheduling is a crucial part of efficient Grid implementation. Diversified aspects of the problem are discussed in the literature to cope with the new challenges of multi domain distributed systems: centralized, hierarchical and distributed models [7,8]; static [9,10] and dynamic scheduling policies [11]; multi-objective optimization [12]; adaptive policies dealing with dynamic behavior of resources [13,14]; autonomic management [15]; QoS and system level agreement constraints [16]; economic models [17]; resource selection [18]; scheduling of data and computational intensive applications [19,20]; workflow scheduling [21,22]; data locality with migration, binding execution and data storage sites [23]; replication [24]; performance evaluation [25]; among other topics.

Major emphasis of these works is to make scheduling decisions using exact resource information, despite their practical weakness and unpredictable impact on the efficiency of the scheduling in real Grid environments.

Scheduling in Grids is studied to give hints for the implementation of real systems in which processing time estimates of jobs may be available. Hence, it is important that a Grid scheduler is able to integrate user time estimates with resource allocation strategies. This is one of the objectives of this paper.

Grids vary significantly in size and the workload is very dynamic. A defined quality of service for scheduling algorithms is an important performance characteristic. Therefore, theoretical worst-case analysis is a relevant approach as it provides such guarantees with performance bounds.

In [26] Schwiegelshohn et al. showed that the performance of Garey and Graham's list scheduling algorithm is significantly worse in Grids than in multiprocessors. There is no polynomial time algorithm that guarantees schedules with a competitive factor <2 for Grids. Therefore, the list scheduling bound of $2-1/m$ for off-line problems, see results of Garey and Graham [27], as well as for on-line ones, does not apply to Grids. Even more, list scheduling cannot guarantee a constant competitive factor in the off-line case [28].

Schwiegelshohn et al. [26] presented an on-line non-clairvoyant algorithm that guarantees a competitive factor of 5 for the Grid scenario, where all available jobs can be used for local scheduling, that is, migration between machines is allowed. Hence, this approximation algorithm guarantees to generate a schedule with completion time being within a constant ratio 5 of the optimal solution. The off-line non-clairvoyant version of this algorithm has an approximation factor of 3.

Tchernykh et al. [1] provided an algorithm for on-line clairvoyant scheduling of parallel rigid jobs in a Grid with the competitive factor $2e + 1$, based on the load balancing model of Bar-Noy and Freund [29].

More theoretical results are available for off-line problems both clairvoyant and non-clairvoyant. We just mention main results without details, because corresponding problems are not addressed in our paper. Pascual et al. [30] modeled an off-line clairvoyant system consisting of N clusters with exactly m identical processors each, and proposed an algorithm that guaranteed a worst-case approximation factor on the global makespan equal to 4. Bougeret et al. [2] considered the problem with k parallel machines of different sizes, and jobs with sizes that must be fitted to the smallest machine. The proposed scheduling algorithm achieves a $5/2$ approximation factor.

Concerning the hierarchical scheduling problem, Tchernykh et al. [28] considered a clairvoyant problem with jobs that can require more resources than available on the smallest machine. They presented off-line algorithms named $MLB_a + LSF$ and $MCT_a + LSF$ with approximation factors of 10. On the first layer, the schedule minimum lower bound and job minimum completion time strategies are used to allocate jobs to resources. On the second layer, the larger size first local scheduling algorithm is applied.

Tchernykh et al. [31] extended these results by introducing the admissible factor that parameterizes the availability of the Grid sites for the job allocation. The competitive factor of the adaptive on-line scheduling algorithm $MLB_a + PS$ with admissible job allocation that varies between 5 and infinity by changing the admissible factor was derived for specific workload characteristics. Tchernykh et al. [1] extended this result for a more general workload model. The competitive factor of the $MLB_a + PS$ algorithm varies between 17 and infinity with change of the admissible factor. Here, we are able to improve our result to a 9 approximation factor without release times, and achieve 11 competitive factor with release times. We also discuss different variation of the problem, and derive their approximation and competitive factors.

4. Algorithms

Algorithms, that have no knowledge about jobs other than the number of unfinished jobs in the system, their processor requirements and user runtime estimates, are considered.

4.1. Hierarchical scheduling

Scheduling algorithms for two layer Grid models can be split into a global allocation part and a local scheduling part. Hence, we regard MPS as a two stage scheduling strategy: $MPS = MPS\_Alloc + PS$. At the first stage, we map each job to a suitable machine using a given selection criterion. At the second stage, the PS algorithm is applied to each machine independently for jobs allocated during the previous stage. Note that our algorithms proceed on a job-by-job basis.

It is easy to see that the competitive factor of the MPS algorithm is lower bounded by the competitive factor of the PS algorithm considering a degenerated Grid that only contains a single machine. The best possible PS on-line non-clairvoyant algorithm has a competitive factor of $2 - 1/m$ with m denoting the number of processors in the parallel machine; see results of Naroska and Schwiegelshohn [32]. Hence, the lower bound of a competitive factor for any general two-layer on-line MPS is at least $2 - 1/m$.

4.1.1. Allocation strategies

Three different levels of available information for job allocation are distinguished in this paper. Each level differs in type and amount of information it requires to perform the job allocation.

Level A: Once a job has been submitted its processor requirements are known. There is no information on the required processing time of jobs, that is, we consider non-clairvoyant scheduling on this level. But our algorithm may use information of previously performed allocations. Algorithms *Random*, *MLp*, *MPL*, *LBal_S* allocate jobs to the admissible site randomly, to the site with least load per processor $(\min_{i=1,\ldots,m}(n_i/m_i))$, to the site with least job processor requirements per processor $(\min_{i=1,\ldots,m}\left\{\sum_{g_k=i}(\text{size}_k/m_i)\right\})$, and to the site with the least standard deviation of job processor requirements per processor $(\min_{q=1,\ldots,m}\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(PL_i^q-\overline{PL}\right)^2})$, with $PL^q_{i=1,\ldots,m} = \frac{1}{m_i}\sum_{g_k=i}\left(\text{size}_k + \text{size}_j^q\right)$ and size_j^q being the size of job J_j added to site q.

Level B: We have access to all information of Level A and to job runtime estimate p'_j. *MLB* allocates job to the site with least work per processor $\min_{i=1,\ldots,m}\left\{\sum_{g_k=i}\frac{\text{size}_k\cdot p'_k}{m_i}\right\}$ before allocating job J_j.

Level C: We have access to all information of Level B and to all local schedules as well. *MCT*, *MWT*, *MWWT_S*, *MST* allocate job J_j to the site with earliest completion time $\min_{i=1,\ldots,m}\{C^i_{\max}\}$ before

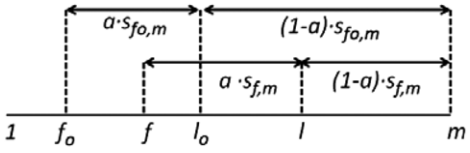


Fig. 1. An example of admissible processors for allocation of jobs with a factor a [1].

allocating job J_j, to the site with minimum average job waiting time $\min_{i=1,\ldots,m}\left\{\sum_{g_k=i}\frac{t_w^k}{n_i}\right\}$, to the site with minimum average job weighted waiting time $\min_{i=1,\ldots,m}\left\{\sum_{g_k=i}(t_w^k\cdot \text{weight}_k/n_i)\right\}$, and to the site with earliest start time for this job $\min_{i=1,\ldots,m}\{s_j^i\}$, respectively. More details on the allocation strategies can be found in [6].

4.1.2. Admissible allocation

The analysis of on-line machine allocation problem has rarely been addressed so far. Unfortunately, the allocation may result in inefficient machine utilization in the worst case. One of the structural reasons for such inefficiency in on-line job allocation is potential occupation of large machines by jobs with small processor requirements causing highly parallel jobs to wait for their execution if they are submitted later.

In [1] the admissible set of machines for a job J_j was defined to be the machines with indexes $\{f_j,\ldots,l_j\}$ (or simple $\{f,\ldots,l\}$, if it does not cause any ambiguity), where f_j is the smallest index i such that $m_i \geq \text{size}_j$, and l_j is the smallest machine index such that $s_{f_j,l_j} \geq a\cdot s_{f_j,m}$ (see Fig. 1), where s_{f_j,l_j} is be the total number of processors belonging to machines from N_{f_j} to N_{l_j}. Note, that $l_j \leq m$ always holds. An admissible factor $0 < a \leq 1$ parameterizes the admissibility ratio used for the job allocation. Note that at least one machine is admissible as a is strictly larger than 0, while $a = 1$ defines all available machines are admissible. If f is the smallest index such that $m_f \geq \text{size}_j$ then the job can be allocated to machines from index f to m. The admissible factor reduces available machines to the machines with indexes from f to l (see Fig. 1).

Let us assume that job J_j is allocated to a machine from the set $f,\ldots,l$ that contains $a\cdot s_{f,m}$ processors (see Fig. 1). Hence, $(1-a)\cdot s_{f,m}$ processors are excluded from $s_{f,m}$ processors eligible for allocation of job J_j. Note that machines are indexed in non-decreasing order of their sizes. Obviously, the total set of machines is represented by the integer set $1,\ldots,m$.

5. Analysis of the algorithms

In this section, we analyze two algorithms named $MLB_a + PS$ and $MCT_a + PS$, where not all machines are admissible for a given job although they are able to execute this job. We show that the performance of allocation strategies depends on the admissible factor, and can be dynamically adapted to cope with different workloads and changes in the configuration.

Theorem 1. *Off-line scheduling of rigid parallel jobs on Grids with identical processors using the algorithm MCT_a+PS with an admissible allocation range $0 < a \leq 1$ has the approximation factor*

$$\rho \leq \begin{cases} 1+\dfrac{2}{a^2} & \text{if } a \leq \dfrac{s_{f,l}}{s_{f_0,m}} \\ 1+\dfrac{2}{a(1-a)} & \text{if } a > \dfrac{s_{f,l}}{s_{f_0,m}} \end{cases}$$

with $1 \leq f_0 \leq f \leq l \leq m$ being parameters that depend on the machine configuration and workload.

Proof. Let us assume that the makespan of machine k is also the makespan $C_{\max}$ of the Grid, so that $C_{\max} = C_k$ holds. Further, let job J_d be the last job that was added to this machine.

Since J_d was added to machine k, MCT_a guarantees $C_k \leq C_i$ before adding J_d. Hence, based on the 2-competitiveness of the algorithm *PS*, we have $C_k \leq \min C_i = \min\{2\cdot\max\{p_{\max}, \frac{W_i}{m_i}\}\}$, and $C_k \leq \max\{\min\{2p_{\max}\}, \min\{2\frac{W_i}{m_i}\}\}$. Let machine e be the machine with the smallest ratio $\frac{W_i}{m_i}$: $\frac{W_e}{m_e} = \min_{f\leq i}\left\{\frac{W_i}{m_i}\right\}$. Hence, $C_k \leq \max\{2p_{\max}, 2\frac{W_e}{m_e}\}$. By adding J_d we obtain

$$C_{\max} \leq C_k + p_d \leq C_k + p_{\max} \quad \text{and}$$
$$C_{\max} \leq \max\left(2\frac{W_i}{m_i} + p_{\max}, 3p_{\max}\right). \tag{1}$$

Let us now consider properties of the optimal solution.

$$W_{f,l} = \sum_{i=f}^{l} W_i = \sum_{i=f}^{l}\frac{W_i}{m_i}\cdot m_i \geq \sum_{i=f}^{l}\frac{W_e}{m_e}\cdot m_i$$
$$= \frac{W_e}{m_e}\sum_{i=f}^{l}\cdot m_i = \frac{W_e}{m_e}\cdot s_{f,l}. \tag{2}$$

Let J_b be the job having the smallest size among all jobs that can be executed at machine N_f in our schedule, that is $l_b \geq f$. Hence jobs packed at $N_f,\ldots,N_l$ cannot be allocated to a machine with a smaller index than f_b. As J_b can be executed on machine N_f, we have $C^*_{\max} \geq \frac{W_{f,l}}{s_{f_0,m}}$. Substituting (2) in this formula, we have $C^*_{\max} \geq \frac{W_e}{m_e}\cdot\frac{s_{f,l}}{s_{f_0,m}}$.

With $s_{f,l} \geq a\cdot s_{f,m}$, we obtain $s_{f,l} \geq a^2\cdot s_{f_0,m}$ if $a \leq \frac{s_{f,m}}{s_{f_0,m}}$ holds. This yields $C^*_{\max} \geq \frac{W_e}{m_e}\cdot\frac{s_{f,l}}{s_{f_0,m}} \geq \frac{W_e}{m_e}\cdot a^2$.

Due to $C^*_{\max} \geq p_{\max}$, we have with (1) $\rho \leq \max\{\frac{2\cdot W_e}{m_e\cdot C^*_{\max}} + \frac{p_{\max}}{C^*_{\max}}, \frac{3p_{\max}}{C^*_{\max}}\} \leq \max\{1+\frac{2}{a^2}, 3\}$ and $\rho \leq 1+\frac{2}{a^2}$, since $a < 1$.

With $s_{f_0,m} \leq s_{f,m} + a\cdot s_{f_0,m}$ (see Fig. 1), and if $a > \frac{s_{f,m}}{s_{f_0,m}}$, we have $C^*_{\max} \geq \frac{W_e}{m_e}\cdot\frac{s_{f,l}}{s_{f_0,m}} \geq \frac{W_e}{m_e}\cdot\frac{a\cdot s_{f,m}}{\frac{s_{f,m}}{1-a}} = \frac{W_e}{m_e}\cdot a\cdot(1-a)$ and $\rho \leq \max\{\frac{2\cdot W_e}{m_e\cdot C^*_{\max}} + \frac{p_{\max}}{C^*_{\max}}, \frac{3p_{\max}}{C^*_{\max}}\} \leq \max\{1+\frac{2}{a\cdot(1-a)}, 3\}$ and $\rho \leq 1+\frac{2}{a\cdot(1-a)}$, since $a < 1$. □

Theorem 2. *Off-line scheduling of rigid parallel jobs on Grids with identical processors using the algorithm MLB_a+PS with an admissible allocation range $0 < a \leq 1$ has an approximation factor of*

$$\rho \leq \begin{cases} 1+\dfrac{2}{a^2} & \text{if } a \leq \dfrac{s_{f,l}}{s_{f_0,m}} \\ 1+\dfrac{2}{a(1-a)} & \text{if } a > \dfrac{s_{f,l}}{s_{f_0,m}} \end{cases}$$

with $1 \leq f_0 \leq f \leq l \leq m$ being parameters that depend on the machine configuration and workload.

Proof. Let us assume that the makespan of machine k is also the makespan of the Grid, so that $C_{\max} = C_k$ holds.

Further, let job J_d be the last job that was added to this machine.

Machines $f = f_d,\ldots,l = l_d$ constitute the set $M_{\text{admissible}}(d)$. Since J_d was added to machine k, MLB_a guarantees $\frac{W_k}{m_k} \leq \frac{W_i}{m_i}$ for all $i = f,\ldots,l$. Note that W_k does not include the work of job J_d.

Hence, based on the property of the 2-competitive *PS* we have

$$C_k \leq \max\left\{2\cdot\frac{W_k}{m_k}, 2\cdot p_{\max}\right\}. \tag{3}$$

We know that

$$W_{f,l} = \sum_{i=f}^{l} W_i = \sum_{i=f}^{l} \frac{W_i}{m_i} \cdot m_i \geq \sum_{i=f}^{l} \frac{W_k}{m_k} \cdot m_i$$
$$= \frac{W_k}{m_k} \sum_{i=f}^{l} \cdot m_i = \frac{W_k}{m_k} \cdot s_{f,l}. \quad (4)$$

Let J_b be the job having the smallest size among all jobs executed at machines $N_f, \ldots, N_l$. We use the notation $f_0 = f_b$. Hence jobs packed at $N_f, \ldots, N_l$ cannot be allocated to a machine with a smaller index than f_0. As J_b is executed on one of the machines $N_f, \ldots, N_l$, we have $l_b \geq f$ and $C^*_{\max} \geq \frac{W_{f,l}}{s_{f_0,m}}$.

Substituting (4), we have $C^*_{\max} \geq \frac{W_k}{m_k} \cdot \frac{s_{f,l}}{s_{f_0,m}}$.

As scheme MLB_a uses W_k without including the work of job J_b, we must consider job J_b in addition. In the worst case, C_k is increased by $p_d \leq C^*_{\max}$ resulting in $C_{\max} \leq C_k + C^*_{\max}$ and $\rho \leq 1 + \frac{C_k}{C^*_{\max}}$. Due to (3), we have $\rho \leq 1 + \frac{\max\{2\frac{W_k}{m_k}, 2p_{\max}\}}{C^*_{\max}}$.

With $s_{f,l} \geq a \cdot s_{f,m}$, we obtain $s_{f,l} \geq a^2 \cdot s_{f_0,m}$ if $a \leq \frac{s_{f,m}}{s_{f_0,m}}$. This yields $C^*_{\max} \geq \frac{W_k}{m_k} \cdot \frac{s_{f,l}}{s_{f_0,m}} \geq \frac{W_k}{m_k} \cdot a^2$ and $\rho \leq 1 + \frac{\max\{2\frac{W_k}{m_k}, 2p_{\max}\}}{C^*_{\max}}$. Since $\frac{2}{a^2} \geq 2$, we have $\rho \leq 1 + \frac{2}{a^2}$.

With $s_{f_0,m} \leq s_{f,m} + a \cdot s_{f_0,m}$ (see Fig. 1), and if $a > \frac{s_{f,m}}{s_{f_0,m}}$, we have $C^*_{\max} \geq \frac{W_k}{m_k} \cdot \frac{s_{f,l}}{s_{f_0,m}} \geq \frac{W_k}{m_k} \cdot \frac{a \cdot s_{f,m}}{\frac{s_{f,m}}{1-a}} = \frac{W_k}{m_k} \cdot a \cdot (1-a)$ and $\rho \leq 1 + \frac{\max\{2\frac{W_k}{m_k}, 2p_{\max}\}}{C^*_{\max}} \leq 1 + \frac{2}{a \cdot (1-a)}$. □

Note that both $MLB_a + PS$ and $MCT_a + PS$ approximation bounds produce the same result $\rho \leq 9$ for $a = 0.5$.

Theorem 3. *On-line scheduling of rigid parallel jobs on Grids with identical processors using the algorithms $MCT_a + PS$ and $MLB_a + PS$ with an admissible allocation range $0 < a \leq 1$ has the competitive factor*

$$\rho \leq \begin{cases} 3 + \frac{2}{a^2} & \text{if } a \leq \frac{s_{f,l}}{s_{f_0,m}} \\ 3 + \frac{2}{a(1-a)} & \text{if } a > \frac{s_{f,l}}{s_{f_0,m}} \end{cases}$$

with $1 \leq f_0 \leq f \leq l \leq m$ being parameters that depend on the machine configuration and workload.

Proof. Let assume that r is the release date of the last job in a schedule. After time r, an off-line algorithm can be applied. However, it must start with all processors being idle. Time span to reach it is limited to the maximum processing time of any job. Therefore, the schedule terminates after time $r + p_{\max} + C_{\max}$ at the latest. The theorem is valid as the inequalities $C^*_{\max} \geq r$ and $C^*_{\max} \geq p_{\max}$ hold, see Theorems 1 and 2. □

Remark. Figs. 2 and 3 show the bounds of the competitive factor of strategies $MCT_a + PS$ and $MLB_a + PS$ if the admissible values $a \leq \frac{s_{f,l}}{s_{f_0,m}}$ and $a > \frac{s_{f,l}}{s_{f_0,m}}$, respectively. One can see that if $a \leq \frac{s_{f,l}}{s_{f_0,m}}$ the worst case bounds change from ∞ to 5 (Fig. 2) as a function of a. If $a > \frac{s_{f,l}}{s_{f_0,m}}$ the worst case bounds change from ∞ (if a is close to 0) to ∞ (if a is close to 1) (see Fig. 3) with minimum value 11 if $a = 0.5$. Fig. 4 shows the resulting bound that is the maximum of worst case competitive factors showed in Figs. 2 and 3. The bound produces $\rho \leq 11$ for $a = 0.5$.

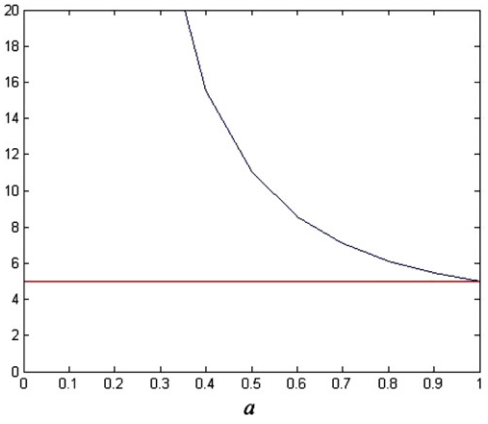


Fig. 2. The competitive factor of strategies $MCT_a + PS$ and $MLB_a + PS$ if $a \leq \frac{s_{f,l}}{s_{f_0,m}}$.

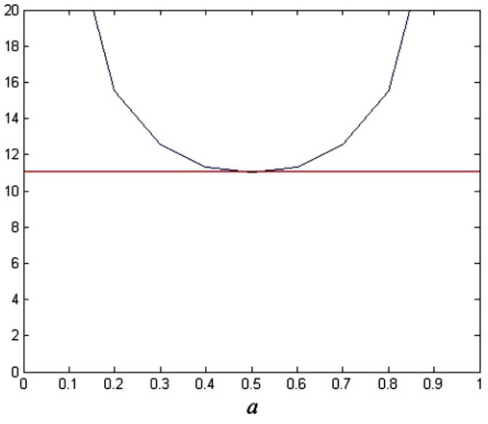


Fig. 3. The competitive factor of strategies $MCT_a + PS$ and $MLB_a + PS$ if $a > \frac{s_{f,l}}{s_{f_0,m}}$.

Theorem 4. *Scheduling of rigid parallel jobs on Grids with identical processors using the algorithms $MCT_a + PS$, $MLB_a + PS$ and jobs with sizes that are fitted to the smallest machine has the approximation factor of 3 and competitive factor of 5.*

Proof. The proof follows immediately considering constraints related to the size of the jobs. If all jobs have sizes of no more than the size of the smallest machine, $size_j \leq m_1$, then all jobs can be allocated to all machines from 1 to m. Hence, we get $f_0 = f = 1$. Considering $l = m$, we have $s_{f,l} = s_{f_0,m}$, $a = 1$. Due to Theorems 1–3, we get $\rho \leq 3$ for the off-line case and $\rho \leq 5$ for the on-line case. □

6. Experimental setup

Two fundamental issues have to be addressed for performance evaluation. On one hand, representative workload traces are needed to produce reliable results. On the other hand, a good testing environment should be set up to obtain reproducible and comparable results. In this section, we present common Grid trace-based simulation setups with an emphasis on the mentioned two issues.

6.1. Grid configuration

One of the goals of this work is to evaluate the viability of our algorithms in large HPC infrastructures. We use several traces collected from HPC and Grid architectures (introduced in the following subsection). Furthermore, we define a possible large multi-site HPC system composed by resources that are related to the traces.

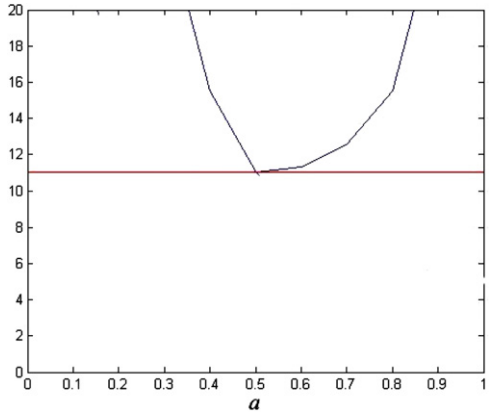


Fig. 4. The competitive factor of strategies $MCT_a + PS$ and $MLB_a + PS$.

We consider three Grid scenarios. In Grid1, we considered 7 existing sites (KTH, SDSC-SP2, HPC2N, CTC, LANL, SDSC-BLUE, SDSC-DS) with total 4442 processors. Corresponding real workload traces are used. These traces belong to machines with 100, 128, 240, 430, 1024, 1152 and 1368 processors, respectively.

In Grid2, we considered 9 sites (5 sites of Grid DAS, KTH, HP2CN, CTC and LANL) with a total of 2194 processors. The sizes of the machines are 64, 64, 64, 64, 100, 144, 240, 430 and 1024 processors. Corresponding traces are used. Further details regarding the traces features can be found in the Parallel Workloads Archive (PWA) [33] and the Grid Workloads Archive (GWA) [34].

In Grid3 scenario, we consider a much smaller Grid of 11 sites with a total of 136 processors of the following sizes 4, 4, 4, 4, 8, 8, 8, 16, 16, 32, 32. We use such a scenario to simulate a heavy load of jobs with high parallelism and low parallelism. It simulates test cases considered in Theorems 1–3, where not all jobs can be fitted to the smallest machine. In Grid3, we use Grid2 workload with jobs that required at most 32 processors. In this workload, 1 processor is required by 37% of jobs; 2, 4, 8, and 32 processors are required by 23%, 10%, 6% and 5% of jobs, respectively.

In Grid1 and Grid2 scenarios, most of jobs can be allocated to the smallest machine, as considered in Theorem 4. For instance, in the traces of Grid2, most of the jobs require at most 32 processors (97%) and can be allocated to the smallest machine of size 64.

6.2. Workload

The accuracy of the evaluation highly relies upon the applied workloads. For testing the job execution performance under a dedicated Grid environment, we use Grid workloads based on real production traces. Carefully reconstructed traces from real supercomputers provide a realistic job stream for simulation-based performance evaluation of Grid job scheduling algorithms. In this paper, traces from the archives PWA and GWA are used.

Unification of these sites into a Grid will require a merging of users and their jobs. It does not guarantee a representation of a real Grid; nevertheless, in absence of common Grid workload traces, it is a reasonable starting point to evaluate Grid scheduling strategies based on real traces. Time zone normalization, profiled time intervals normalization, and invalid jobs filtering are considered. More information about their properties and detailed characteristics of individual traces can be found in the archive of Dror Feitelson [33]. Note that they also provide user run time estimates for all jobs. Background workload (locally generated jobs) is an important issue in non-dedicated Grid environments, but is not considered in this paper.

The number of jobs in Grid1 workload is just half of the number of jobs in Grid2 workload. However, resource consumption is twice as much. It gives us two different scenarios for simulation.

In Grid1 workload, most of the jobs require at most 128 processors (95%). 1 processor is required by 18% of jobs; 8, 16, 32, 64 processors are required by 20%, 9%, 14% and 8% of jobs, respectively. The workload of Grid2 is less parallel. Most of the jobs require at most 32 processors (97%), 1 processor is required by 36% of jobs; 2, and 4 processors are required by 26% and 9% of jobs, respectively. Hence, we have observed predominance of low parallel jobs in both logs with maximal job sizes 1368, 1024, 32 in Grid1, Grid2 and Grid3, respectively.

7. Simulation results

This section presents and analyzes the simulation results of our job allocation strategies using the performance metrics as described in Section 2. Details of the Grid configurations and traces used in our experiments are described in Section 6.

The experimental evaluation of the two stage scheduling strategies $MPS = MPS\_Alloc + PS$ is performed in two steps. In the first step, we compare nine allocation strategies presented in Section 4.1.1. User run time estimates are used in both *MPS_Alloc* and *PS*. In the second part, we present a detailed study of the impact of the admissible factor incorporated into the job allocation policies on the overall Grid performance. We show that the presented algorithms are beneficial under certain conditions, and allow an efficient implementation in real systems.

To obtain valid statistical values, experiments of 6-months time interval are simulated for Grid1 and Grid2, and 30 experiments of 7 days for Grid3.

7.1. Evaluation method

Since the problem is multi-objective in its general formulation, several performance criteria are considered. A good scheduling algorithm should schedule jobs to achieve high Grid performance while satisfying various other demands. Often, resource providers and users have different, sometimes conflicting, performance goals: from minimizing response time to optimizing the resources utilization. Grid resource management may use multi-criteria decision support. General multi-criteria decision methodology based on the Pareto optimality can be applied for this purpose. However, it is very difficult to achieve fast solutions by using the Pareto dominance as needed for Grid resource management. The problem is very often simplified to a single objective problem or to the different methods of objective combining. There are various ways to model preferences, for instance, they can be given explicitly by stakeholders to specify the importance of every criterion or a relative importance between criteria. This can be done by a definition of criteria weights or criteria ranking by their importance.

In order to provide effective guidance in choosing the best strategy we performed a combined analysis of several metrics according to the methodology proposed by Tsafrir et al. [35] and applied for the Grid scheduling problem in [6]. They introduce an approach to multi-criteria analysis assuming equal importance of each metric. The goal is to find a robust and well performing strategy under all test cases, with the expectation that it will also perform well under other conditions, e.g., with different Grid configurations and workloads.

The analysis is conducted as follows. First, we evaluate the degradation (relative error) in performance of each strategy under each metric. This is done relative to the best performing strategy for the metric: $100 \cdot (\frac{\text{metric}}{\text{best\_metric}} - 1)$. Thus, for three main metrics (Section 2.2), each strategy is characterized by 3 numbers (9 for 3 Grids), reflecting its relative performance degradation under the test cases. Then, we average these 3 values (assuming equal importance of each metric), and rank the strategies for each Grid. The best strategy, with the lowest average performance degradation, has rank 1; the worst strategy has rank 9.

Table 1
Rounded percentages of the performance degradations of the strategies for 3 Grids and 3 metrics.

Metrics	Strategy								
		MCT	*MLB*	*LBal_S*	*MLp*	*MPL*	*Random*	*MST*	*MWWT_S$_a$*	*MWT*
ρ	Grid1	0	0	0	2	0	21	0	7	0
	Grid2	0	0	0	0	0	2	0	0	0
	Grid3	2	22	19	87	17	83	0	45	26
SD_b	Grid1	1284	408	18	71	0	41 924	91	12 576	204
	Grid2	4001	754	17	53	0	12 196	111	7 550	628
	Grid3	1	33	10	65	18	46	0	50	42
t_w	Grid1	1256	463	11	119	0	51 929	80	17 303	252
	Grid2	4833	1799	70	153	0	18 027	252	9 132	933
	Grid3	1	32	9	65	16	49	0	47	40

Table 2
Mean percentages of the performance degradation of the strategies and their ranking of Grid1, Grid2 and Grid3.

		MCT	*MLB*	*LBal_S*	*MLp*	*MPL*	*Random*	*MST*	*MWWT_S*	*MWT*
Mean	Grid1	846.7	290.3	9.7	64.0	0.0	31 291.3	57.0	9962.0	152.0
	Grid2	2944.7	851.0	29.0	68.7	0.0	10 075.0	121.0	5560.7	520.3
	Grid3	1.3	29.0	12.7	72.3	17.0	59.3	0.0	47.3	36.0
Mean over all test cases		1264.23	390.10	17.13	68.33	5.67	13 808.53	59.33	5190.00	236.10
Ranking	Grid1	7	6	2	4	1	9	3	8	5
	Grid2	7	6	2	3	1	9	4	8	5
	Grid3	2	5	3	9	4	8	1	7	6
Ranking over all test cases		7	6	2	4	1	9	3	8	5

Table 3
Grid3. Percentages of the performance improvements of the strategies with admissible allocation if $a = 0.5$ (3 metrics).

Metrics	Strategy								
	MCT$_a$	*MLB$_a$*	*LBal_S$_a$*	*MLp$_a$*	*MPL$_a$*	*Random$_a$*	*MST$_a$*	*MWWT_S$_a$*	*MWT$_a$*
ρ	1.25	2.51	0.37	24.79	1.54	22.68	0.78	2.54	3.06
SD_b	0.96	0.80	3.12	62.69	5.20	7.76	1.78	4.49	11.64
t_w	1.06	1.11	2.75	57.88	4.80	10.24	1.80	0.67	8.29
Average	1.09	1.47	2.08	48.45	3.85	13.56	1.45	2.57	7.66
Ranking	9	7	6	1	4	2	8	5	3

Then, we calculate the average performance degradation of Grid1, Grid2, and Grid3. We try to identify strategies which perform reliably well in different scenarios; that is, we try to find a compromise that considers all of our test cases. For example, the rank of the strategy in the average performance degradation could not be the same for any of the metrics individually or for any of the Grid scenarios individually.

7.2. Allocation strategies

The performance data for various combinations of allocation strategies with *EASY* local scheduling algorithm are presented in [6]. The authors show the mean degradations of fourteen allocation strategies in Grid1 and Grid2, considering 24 frequently used performance metrics.

To consider more test cases, in this paper, we evaluate these strategies under Grid3 scenario. Table A.1 in Appendix shows complete results. Table 1 shows the performance degradation of strategies over our three main metrics. Table 2 shows the mean performance degradation and ranking of Grid1, Grid2 and Grid3 scenarios.

We find that in large Grids the *MPL* and *LBal_S* allocation strategies that take only job sizes into account perform better than the others algorithms for waiting time, bounded slowdown, and competitive factor metrics. Additional information used in the allocation phase such as available on the level B and C does not help to construct better Grid schedules.

In small Grids with heavy load the allocation strategies *MST* and *MCT* that use local schedule information, also referred as resource state information, slightly surpass other approaches in terms of the same metrics if user runtime estimates p'_j are used. However, in all test cases the *MPL* and *LBal_S* show advantages.

It turns out that *MPL* and *LBal_S* are the best performing algorithms. Besides the performance aspect the use of *MPL* and *LBal_S* does not require additional management overhead such as requesting information about local schedules or constructing preliminary schedules by the Grid broker. We arrive at this conclusion based on a wide range of simulation parameters and metrics.

We find that in large grids small degradation in the competitive factor is probably because in the considered on-line scenarios and given workloads the last arrived jobs determined the makespans.

7.3. Admissible allocation strategies

The admissible factors used for all experiments are in the range from 0.1 to 1 with step 0.1.

Tables A.2–A.4 in the Appendix show complete admissible allocation results. Table 3 shows percentages of the performance improvements of the strategies considering our three main metrics. Improvements considering all metrics are presented in Table A.2 in Appendix.

The scheduling length is reduced in most strategies when applying the admissible job allocation. *MLp$_a$* and *Random$_a$* reduce the competitive factors more than 20%. The strategies *MST$_a$* and *MCT$_a$* with the smallest initial competitive factors have small improvements (0.78% and 1.25%). We can see that the performance improvement of bounded slowdown varies greatly from 0.8% up to 62.69%, and waiting time from 0.67% up to 57.88%.

Table 4
Grid3. The performance degradation ranking of the strategies without ($a = 1$) and with admissible job allocation ($a = 0.5$).

Rank	Strategy								
	MCT	*MLB*	*LBal_S*	*MLp*	*MPL*	*Random*	*MST*	*MWWT_S*	*MWT*
$a = 1$	2	5	3	8	4	9	1	7	6
$a = 0.5$	3	7	4	1	5	8	2	9	6

Table 5
Grid 3. Percentages of the performance degradation of strategies and their ranking if $a = 0.5$ and $a = 1$ (3 metrics).

Metrics	Strategy																	
	MCT_a	*MCT*	MLB_a	*MLB*	$LBal\_S_a$	*LBal_S*	MLp_a	*MLp*	MPL_a	*MPL*	$Random_a$	*Random*	MST_a	*MST*	$MWWT\_S_a$	*MWWT_S*	MWT_a	*MWT*
ρ	1	2	20	23	20	19	41	87	15	17	41	83	0	0	43	45	22	26
SD_b	62	63	114	115	73	79	0	168	81	91	118	136	59	62	142	143	112	130
t_w	44	45	87	90	52	57	0	137	58	66	93	115	41	44	110	111	85	102
Average	36	37	74	76	48	52	14	131	51	58	84	111	33	35	98	100	73	86
Ranking	4	5	11	12	6	8	1	18	7	9	13	17	2	3	15	16	10	14

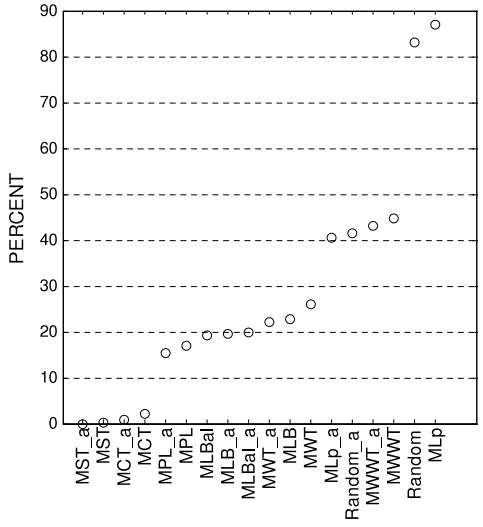


Fig. 5. Grid3. Mean competitive factor degradation ranking.

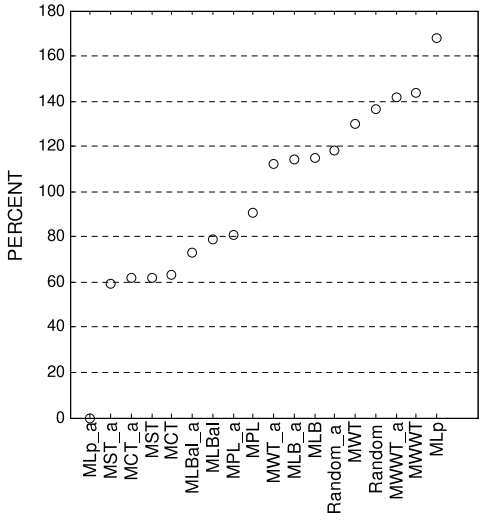


Fig. 6. Grid3. Mean bounded slowdown degradation ranking.

MLp and *Random* show better improvements with admissible job allocation constraints. These constraints incorporate job processor requirements into the decision maker helping to obtain better load balance. Other strategies that use job sizes in their original definition have considerable smaller benefit.

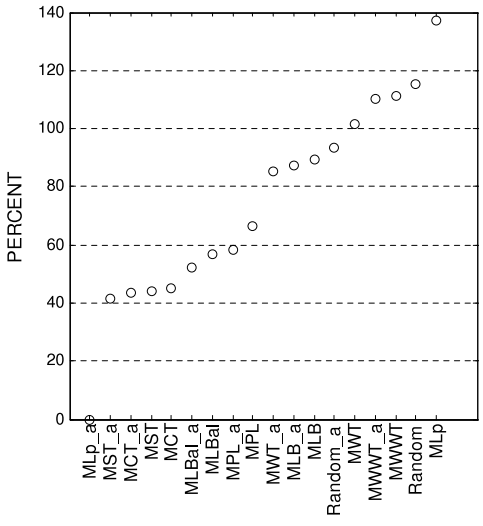


Fig. 7. Grid3. Mean waiting time degradation ranking.

Table 4 shows the ranking of the strategies with and without admissible allocation in Grid3. In this scenario, with $a = 1$ *MST*, *MCT*, *LBal_S*, and *MPL* outperform the other algorithms providing graceful performance degradation and able to cope with different demands. However, with admissible allocation of $a = 0.5$ *MLp* outperforms those showing the best results.

Table 5 shows percentages of the performance degradation of strategies if $a = 0.5$ and $a = 1$ considering three metrics average.

Figs. 5–8 shows the performance degradations rankings of 19 strategies with and without admissible scheme. Figs. 5–7 depict the ranking for each metric with percent of corresponding degradation. As expected, MST_a, *MST*, MCT_a, and *MCT* strategies deliver good performance and outperforms other algorithms in terms of approximation factor. MLp_a is the best performing algorithm for mean bounded slowdown and mean waiting time metrics. MST_a, MCT_a, *MST*, and *MCT* perform well for minimization of these two metrics. They show a similar behavior and differ by 60% in the performance degradation from the best strategy in terms of mean bounded slowdown and 45% in terms of mean waiting time.

Fig. 8 shows the mean degradation ranking considering all test cases.

We can see that the three best strategies are achieved with an admissible factor $a = 0.5$.

Table A.1
Percentages of the performance degradation of Grid 1, Grid 2 and Grid 3.

Metric	Strategy								
		MCT	*MLB*	*LBal_S*	*MLp*	*MPL*	*Random*	*MST*	*MWWT_S$_a$*	*MWT*
ρ	Grid1	0	0	0	2	0	21	0	7	0
	Grid2	0	0	0	0	0	2	0	0	0
	Grid3	2	22	19	87	17	83	0	45	26
SD	Grid1	1501	469	20	93	0	53 741	100	17 658	236
	Grid2	5734	1031	10	62	0	17 688	110	17 609	999
	Grid3	1	33	13	63	19	41	0	54	43
SD_b	Grid1	1284	408	18	71	0	41 924	91	12 576	204
	Grid2	4001	754	17	53	0	12 196	111	7 550	628
	Grid3	1	33	10	65	18	46	0	50	42
Th	Grid1	0	0	0	2	0	18	0	7	0
	Grid2	0	0	0	0	0	2	0	0	0
	Grid3	1	18	17	47	14	44	0	27	19
TA	Grid1	47	17	0	4	0	1 923	3	641	9
	Grid2	93	34	1	3	0	345	5	175	18
	Grid3	1	32	9	65	15	49	0	46	40
U	Grid1	0	0	0	2	0	18	0	7	0
	Grid2	0	0	0	0	0	2	0	0	0
	Grid3	2	18	16	46	15	46	0	30	20
t_w	Grid1	1256	463	11	119	0	51 929	80	17 303	252
	Grid2	4833	1799	70	153	0	18 027	252	9 132	933
	Grid3	1	32	9	65	16	49	0	47	40
WTA	Grid1	100	49	0	35	2	2 513	0	909	35
	Grid2	135	74	2	19	0	457	7	271	55
	Grid3	2	28	8	112	16	111	0	13	34
WTA_w	Grid1	23	5	0	6	0	815	1	292	8
	Grid2	20	7	1	4	0	91	2	65	11
	Grid3	2	24	10	102	16	110	0	11	30
WWT_s	Grid1	378	187	0	130	6	9 483	1	3 429	132
	Grid2	700	384	10	101	0	2 378	39	1 411	287
	Grid3	2	28	8	112	16	112	0	13	34

Table A.2
Grid 3. Percent of the performance metric improvement with corresponding admissibility factor.

Metric	Strategy								
	MCT$_a$	*MLB$_a$*	*LBal_S$_a$*	*MLp$_a$*	*MPL$_a$*	*Random$_a$*	*MST$_a$*	*MWWT_S$_a$*	*MWT$_a$*
ρ	1.25	2.51	0.37	24.79	1.54	22.68	0.78	2.54	3.06
	0.5	0.5	0.6	0.5	0.5	0.5	0.6	0.6	0.5
SD	1.06	0.37	4.08	65.75	5.68	5.44	1.78	6.18	12.80
	0.5	0.6	0.5	0.5	0.5	0.5	0.5	0.2	0.2
SD_b	0.96	0.80	3.12	62.69	5.20	7.76	1.78	4.49	11.64
	0.5	0.2	0.5	0.5	0.5	0.5	0.5	0.2	0.2
Th	0.51	3.14	0.00	34.68	0.0	25.62	1.57	3.35	1.19
	0.5	0.5	0.6	0.5	–	0.5	0.6	0.5	0.5
TA	1.06	1.11	2.75	57.83	4.79	10.22	1.80	0.67	8.28
	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.2	0.2
U	1.12	2.55	0.32	31.25	1.64	29.43	0.67	2.44	3.01
	0.5	0.5	0.6	0.5	0.5	0.5	0.6	0.6	0.5
t_w	1.06	1.11	2.75	57.88	4.80	10.24	1.80	0.67	8.29
	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.2	0.2
WTA	1.41	4.13	2.46	36.06	4.78	29.81	0.67	0.81	7.09
	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
WTA_w	1.83	4.22	1.91	29.87	4.31	30.58	0.76	0.53	7.13
	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.6	0.5
WWT_s	1.41	4.13	2.46	36.09	4.79	29.84	0.67	0.81	7.10
	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

8. Conclusion and future work

As Grids become more prevalent, the problem of resource allocation and scheduling is crucial, not only to achieve high Grid performance, but also to satisfy various demands. While theoretical Grid scheduling models based on worst case analysis are beginning to emerge, fast statistical techniques applied to real data have empirically been shown to be effective.

In this paper, we address non-preemptive on-line scheduling of parallel jobs on a Grid. We present a detailed performance evaluation of job allocations algorithms. Our extensive study results in several contributions.

Firstly, we derived a competitive bound of 11 for the on-line adaptive Grid scheduling algorithms MLB_a+PS and MCT_a+PS with admissible job allocation improving bound of 17 as presented by the authors in paper [1], and provide a new off-line approximation bound of 9.

Table A.3
Grid 3. Percentages of the performance degradation of strategies with $a = 1$ and $a = 0.5$.

Metric	Strategy																	
	MCT_a	MCT	MLB_a	MLB	$LBal\_S_a$	$LBal\_S$	MLp_a	MLp	MPL_a	MPL	$Random_a$	$Random$	MST_a	MST	$MWWT\_S_a$	$MWWT\_S$	MWT_a	MWT
ρ	1	2	20	23	20	19	41	87	15	17	41	83	0	0	43	45	22	26
SD	79	81	139	139	94	102	0	192	102	114	140	154	76	80	174	176	137	156
SD_b	62	63	114	115	73	79	0	168	81	91	118	136	59	62	142	143	112	130
Th	1	2	16	19	19	17	29	47	17	14	30	45	0	1	25	28	19	19
TA	43	45	87	89	52	56	0	137	58	66	93	115	41	44	110	111	85	101
U	1	2	16	18	16	16	29	46	13	15	30	46	0	0	29	30	18	20
t_w	44	45	87	90	52	57	0	137	58	66	93	115	41	44	110	111	85	102
WTA	1	3	24	29	6	9	36	113	11	17	49	113	0	1	12	13	25	35
WTA_w	1	3	19	25	8	11	43	104	12	17	47	112	0	1	11	12	22	31
WWT_s	1	3	24	29	6	9	37	114	11	17	49	113	0	1	12	13	25	35
Average	23.4	24.9	54.6	57.6	34.6	37.5	21.5	114.5	37.8	43.4	69	103.2	21.7	23.4	66.8	68.2	55	65.5

Table A.4
Grid 3. Percentages of the performance degradation of strategies with $a = 0.5$.

Metrics	Strategy								
	MCT_a	MLB_a	$LBal\_S_a$	MLp_a	MPL_a	$Random_a$	MST_a	$MWWT\_S_a$	MWT_a
ρ	1	20	20	41	15	41	0	43	22
SD	79	139	94	0	102	140	76	174	137
SD_b	62	114	73	0	81	118	59	142	112
Th	1	16	19	29	17	30	0	25	19
TA	43	87	52	0	58	93	41	110	85
U	1	16	16	29	13	30	0	29	18
t_w	44	87	52	0	58	93	41	110	85
WTA	1	24	6	36	11	49	0	12	25
WTA_w	1	19	8	43	12	47	0	11	22
WWT_s	1	24	6	37	11	49	0	12	25
Average	23.4	54.6	34.6	21.5	37.8	69	21.7	43	55

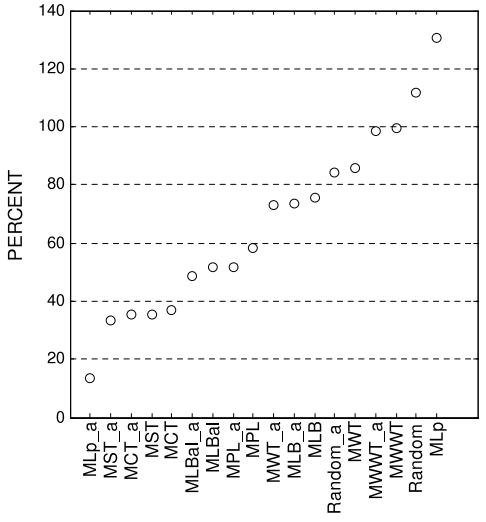


Fig. 8. Grid3. Mean degradation ranking for all test cases.

We also adapted our problem to the one described in [2], where all jobs fit to the smallest machine, and derived an approximation factor of 3 and a competitive factor of 5.

Note that in [2] a 5/3 approximation algorithm is provided only for off-line problem, while on-line scheduling was not considered. Moreover, our algorithms are simple, proceed on a job-by-job basis, and allow an efficient implementation in real systems.

Secondly, we study nine allocation strategies depending on the type and amount of information they require together with *EASY* backfilling local scheduling algorithm. We conduct a comprehensive evaluation of their practical performance using simulation for three different Grid scenarios over ten metrics.

Thirdly, we present a detailed study of the impact of the admissible factor incorporated into the job allocation policies on overall Grid performance. We show that the algorithms are beneficial under certain conditions.

The selection of the appropriate strategy for Grids depends on preferences of different decision makers of Grids (end-users, local resource providers, and Grid administrators). To provide effective guidance in choosing a good strategy, we performed a joint analysis of three metrics based on the degradation in performance of each strategy under each metric in all test cases.

Simulation results presented in the paper reveal that in terms of considered criteria, admissible allocation strategies outperforms algorithms that use all available sites for job allocation. We conclude that adaptive admissible allocation strategies are robust and stable even in vastly different conditions, and able to cope with different workloads.

In real Grids, the admissible factor can be dynamically adjusted to cope with the dynamic workload situation. To this end, the past workload must be analyzed for a certain time interval to determine an appropriate admissible factor. The time interval should be set according to workload characteristics, and Grid configurations.

When we examine the overall Grid performance on the real data, we find that strategies perform well if they take into account job processor requirements (job size) for job allocation. Allocation strategies that are based on knowledge of user run time estimates and local schedules show lower performance.

The final results suggest simple schedulers with admissible job allocation, which require minimal information and little computational complexity.

Acknowledgments

The authors would like to thank all students who contributed to this study, and the anonymous referees whose valuable remarks and comments helped us to improve the paper. This

work is partly supported by CONACYT (Consejo Nacional de Ciencia y Tecnología de México) under grant no. 48385 and DAAD (Deutscher Akademischer Austauschdienst) Section:414, A/09/03178.

Appendix

See Tables A.1–A.4.

References

[1] A. Tchernykh, U. Schwiegelshohn, R. Yahyapour, N. Kuzjurin, On-line hierarchical job scheduling on Grids with admissible allocation, Journal of Scheduling (ISSN: 1094-6136) 13 (5) (2010) 545–552. doi:10.1007/s10951-010-0169-x. Springer-Verlag, Netherlands (Print).

[2] M. Bougeret, P.-F. Dutot, K. Jansen, C. Otte, D. Trystram, A fast 5/2 approximation algorithm for hierarchical scheduling, in: 16th International European Conference on Parallel and Distributed Computing, Euro-Par 2010, Ischia, Italy, 2010.

[3] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, in: P.L. Hammer, E.L. Johnson, B.H. Korte (Eds.), Discrete Optimization II, in: Annals of Discrete Mathematics, vol. 5, North-Holland, 1979, pp. 287–326.

[4] D. Lifka, The ANL/IBM SP scheduling system, in: Proc. 1st Workshop on Job Scheduling Strategies for Parallel Processing, in: LNCS, vol. 949, 1995, pp. 295–303.

[5] U. Schwiegelshohn, An owner-centric metric for the evaluation of online job schedules, in: Multidisciplinary International Conference on Scheduling. Theory and Applications, MISTA 2009, Dublin, Ireland, 2009, pp. 557–569.

[6] J.-M. Ramírez-Alcaraz, A. Tchernykh, R. Yahyapour, U. Schwiegelshohn, A. Quezada-Pina, J.-L. González-García, A. Hirales-Carbajal, Job Allocation Strategies with User Run Time Estimates for Online Scheduling in Hierarchical Grids, Springer, 2011, ISSN: 1570-7873.

[7] V. Hamscher, U. Schwiegelshohn, A. Streit, R. Yahyapour, Evaluation of job-scheduling strategies for Grid computing, in: Proc. of GRID 2000 GRID 2000, First IEEE/ACM International Workshop, Bangalore, India, December 2000, pp. 191–202.

[8] S. Zikos, H.D. Karatza, Resource allocation strategies in a 2-level hierarchical Grid system, in: Proceedings of the 41st Annual Simulation Symposium, ANSS, IEEE Computer Society Press, SCS, Ottawa, Canada, 2008, pp. 157–174.

[9] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen, R. Freund, A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, Journal of Parallel and Distributed Computing 61 (6) (2001) 810–837.

[10] P. Huang, H. Peng, P. Lin, X. Li, Static strategy and dynamic adjustment: an effective method for Grid task scheduling, Future Generation Computer Systems 25 (8) (2009) 884–892.

[11] K. Kurowski, B. Ludwiczak, J. Nabrzyski, A. Oleksiak, J. Pukacki, Improving Grid level throughput using job migration and rescheduling, Scientific Programming 12 (4) (2004) 263–273.

[12] A. Kuijl, M. Emmerich, H. Li, A novel multi-objective optimization scheme for Grid resource allocation, in: Proceedings of the 6th International Workshop on Middleware for Grid Computing, MGC'08, ACM, New York, NY, USA, 2008.

[13] A. Aggarwal, R. Kent, An adaptive generalized scheduler for Grid applications, in: Proc. of the 19th Annual International Symposium on High Performance Computing Systems and Applications, HPCS'05, pp. 15–18, Guelph, Ontario, Canada, May 2005.

[14] L.-T. Lee, C.-H. Liang, H.-Y. Chang, An adaptive task scheduling system for Grid computing, in: Sixth IEEE International Conference on Computer and Information Technology, CIT'06, 2006, p. 57.

[15] M. Rahman, R. Ranjan, R. Buyya, B. Benatallah, A taxonomy and survey on autonomic management of applications in Grid computing environments, Journal of Concurrency and Computation: Practice and Experience (ISSN: 1532-0626) 23 (16) (2011) 1990–2019. Wiley Press, New York, USA.

[16] I. Foster, Globus toolkit version 4: software for service-oriented systems, in: The Proc. of IFIP International Conference on Network and Parallel Computing, in: LNCS, vol. 3779, Springer-Verlag, Beijing, China, 2005, pp. 2–13.

[17] R. Buyya, D. Abramson, J. Giddy, H. Stockinger, Economic models for resource management and scheduling in Grid computing, Journal of Concurrency and Computation: Practice and Experience 14 (13–15) (2002) 1507–1542. Wiley Press.

[18] I. Rodero, F. Guim, J. Corbalan, L. Fong, S. Masoud Sadjadi, Grid broker selection strategies using aggregated resource information, Future Generation Computer Systems 26 (1) (2010) 72–86.

[19] Y.-H. Lee, S. Leu, R.-S. Chang, Improving job scheduling algorithms in a grid environment, Future Generation Computer Systems 27 (8) (2011) 991–998.

[20] S. Zhuk, A. Chernykh, A. Avetisyan, S. Gaissaryan, D. Grushin, N. Kuzjurin, A. Pospelov, A. Shokurov, Comparison of scheduling heuristics for Grid resource broker, in: Third International IEEE Conference on Parallel Computing Systems, PCS 2004, IEEE, Colima, Colima, México, 2004, pp. 388–392.

[21] J. Yu, R. Buyya, A taxonomy of workflow management systems for Grid computing, Journal of Grid Computing 3 (3–4) (2005) 29. Springer.

[22] Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, Matthew Shields (Eds.), Workflows for e-Science: Scientific Workflows for Grids, Springer, 2007.

[23] H. Dail, H. Casanova, F. Berman, A decoupled scheduling approach for the GrADS environment, in: Proc. 2002 ACM/IEEE conference on Supercomputing, Baltimore, Maryland, USA, November 2002, pp. 1–14.

[24] D. Silva, W. Cirne, F. Brasileiro, Trading cycles for information: using replication to schedule bag-of-tasks applications on computational Grids, in: Proc of Euro-Par 2003, Klagenfurt, Austria, August 2003, pp. 169–180.

[25] H. Li, R. Buyya, Model-based simulation and performance evaluation of Grid scheduling strategies, Future Generation Computer Systems (ISSN: 0167-739X) 25 (4) (2009) 460–465. Elsevier Science, Amsterdam, The Netherlands.

[26] U. Schwiegelshohn, A. Tchernykh, R. Yahyapour, On-line scheduling in Grids, in: IEEE International Symposium on Parallel and Distributed Processing 2008, IPDPS 2008, Miami, FL, USA, 2008, pp. 1–10.

[27] M.R. Garey, R.L. Graham, Bounds for multiprocessor scheduling with resource constraints, SIAM Journal on Computing 4 (1975) 187–200.

[28] A. Tchernykh, J. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin, S. Zhuk, Two level job-scheduling strategies for a computational Grid, in: R. Wyrzykowski, J. Dongarra, N. Meyer, J. Wasniewski (Eds.), 6th International Conference on Parallel Processing and Applied Mathematics PPAM 2005, in: LNCS, vol. 3911, Springer, Berlin, Heidelberg, Poznan, Poland, 2006, pp. 774–781.

[29] A. Bar-Noy, A. Freund, On-line load balancing in a hierarchical server topology, SIAM Journal of Computing 31 (2001) 527–549.

[30] F. Pascual, K. Rzadca, D. Trystram, Cooperation in multi-organization scheduling, Journal of Concurrency and Computation: Practice and Experience 21 (2009) 905–921.

[31] A. Tchernykh, U. Schwiegelshohn, R. Yahyapour, N. Kuzjurin, On-line hierarchical job scheduling on Grids, in: T. Priol, M. Vanneschi (Eds.), From Grids to Service and Pervasive Computing, Springer, US, 2008, pp. 77–91.

[32] E. Naroska, U. Schwiegelshohn, On an on-line scheduling problem for parallel jobs, Information Processing Letters 81 (2002) 297–304.

[33] PWA: parallel workloads archive. http://www.cs.huji.ac.il/labs/parallel/workload/.

[34] GWA: Grid workloads archive, TU Delft. http://gwa.ewi.tudelft.nl.

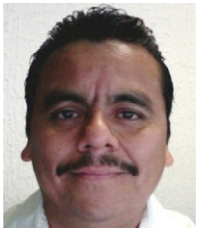
[35] D. Tsafrir, Y. Etsion, D.G. Feitelson, Backfilling using system-generated predictions rather than user runtime estimates, IEEE Transactions on Parallel and Distributed Systems 18 (2007) 789–803.

Ariel Quezada-Pina received a bachelor's degree in Electronics Engineering from Technological Institute of Sonora, Mexico in 2001. He earned the M.S. degree in Computer Sciences from CICESE Research Center in 2007. Actually he is a Ph.D. student working on Grid scheduling problems.

Andrei Tchernykh is a researcher in the Computer Science Department, CICESE Research Center, Ensenada, Baja California, Mexico. From 1975 to 1990 he was with the Institute of Precise Mechanics and Computer Technology of the Russian Academy of Sciences. He received the Ph.D. in Computer Science in 1986. In CICESE, he is a coordinator of the Parallel Computing Laboratory. He is PI of several national and international research projects. He is active in grid research with a focus on resource optimization. He served as a program committee member of several professional conferences and a general co-chair for International Conferences on Parallel Computing Systems. His main interests include scheduling, multi-objective optimization, resource management in GRID and Cloud computing.

José Luis González-García is currently a Ph.D. student at Georg-August-Universität Göttingen in Germany. He received the M.S. degree in Computer Science from CICESE Research Center, Ensenada, BC, México in 2009. His current research focuses on resource management in FEM simulations, and his interests include Grid scheduling parallel and distributed algorithms.

Adán Hirales-Carbajal graduated from the Faculty of Sciences of the Autonomous University of Baja California in 1999. He earned the M.S. degree in Computer Science from CICESE Research Center in 2001. Currently he is a Ph.D. student at the Department of Computer Science in CICESE. His research deals with workflow scheduling problems on Grid architectures. His main interests include parallel algorithms, distributed algorithms, scheduling, multi-objective optimization, and resource management in GRID computing.



Juan Manuel Ramírez-Alcaraz is an associate professor at University of Colima. He received the Ph.D. in Computer Science from CICESE Research Center in 2011. His work is focused on the area of job scheduling in hierarchical computational Grids. He received his B.S. degree in Computational Systems Engineering in 1994 and his Master degree in Telematics in 2001, both from University of Colima, Colima, México. He has worked as a professor in Computer Science at University of Colima for 22 years. His principal interest areas include parallel programming, job scheduling, Grid and Cloud computing.

Uwe Schwiegelshohn has been the head of the Robotics Research Institute at TU Dortmund University since 2005. For many years, he has been active in grid research with a focus on resource management. For instance, he was leading a working group and a research group in OGF (GGF) and was a prominent member of the network of excellence CoreGRID. In D-Grid, he was responsible for the provision of middleware and now addresses sustainability issues. In addition, his group is involved in more grid projects. Since 2008 he has been the managing director of the D-Grid Corporation, a nonprofit corporation that coordinates the various projects in D-Grid.

Ramin Yahyapour is the managing director of the GWDG University of Göttingen. He is active in research on Clouds, Grid and Service-oriented Infrastructures for several years. His research interest lies in resource management. He is a steering group member and is on the Board of Directors in the Open Grid Forum. He has been active in several national and European research projects. Notably, he is the scientific coordinator of the FP7 IP SLA@SOI and was a steering group member in the CoreGRID Network of Excellence.

Vanessa Miranda-López received a bachelor's degree in Electronics Engineering from Technological Institute of Sonora, Mexico in 2006. She earned the M.S. degree in Computer Sciences from CICESE Research Center in 2010.