Two Level Job-Scheduling Strategies for a Computational Grid⋆

Andrei Tchernykh[1], Juan Manuel Ramírez[2], Arutyun Avetisyan[3],
Nikolai Kuzjurin[3], Dmitri Grushin[3], and Sergey Zhuk[3]

[1] CICESE Research Center, Ensenada, México
`chernykh@cicese.mx`
[2] University of Colima, México
`jmramir@ucol.mx`
[3] Institute of System Programming RAS, Moscow
`{arut, nnkuz, grushin, zhuk}@ispras.ru`

Abstract. We address parallel jobs scheduling problem for computational GRID systems. We concentrate on two-level hierarchy scheduling: at the first level broker allocates computational jobs to parallel computers. At the second level each computer generates schedules of the parallel jobs assigned to it by its own local scheduler. Selection, allocation strategies, and efficiency of proposed hierarchical scheduling algorithms are discussed.

1 Introduction

Recently, parallel computers and clusters have been deployed to support computation-intensive applications and become part of so called computational grids (C-GRIDs) or metacomputers [10, 8]. Such C-GRIDs are emerging as a new paradigm for solving large-scale problems in science, engineering, and commerce [15]. They comprise heterogeneous nodes (typically, clusters and parallel supercomputers) with a variety of computational resources. The efficiency of scheduling policies is crucial to C-GRID performance. The job scheduling solutions for a single parallel computer significantly differs from scheduling solution in such a grid. The scheduling problem becomes more complicated because many computers of different sizes are involved with different local scheduling policies [11, 12, 13, 14]. One possible solution is to consider two-level scheduling schemes: at the first level jobs are allocated to parallel computers by a GRID resource broker and then local schedulers are used at each computer. Typically, the broker is responsible for resource discovery, resource selection, and job assignment to ensure that the user requirements and resource owner objectives are met. The broker acts as a mediator between users and resources using middleware services. It is responsible for presenting the grid to the user as a single, unified resource. One of the broker's major responsibilities is to provide centralized access to distributed

resources. This simplifies the use of the computational GRID aggregating available computational resources, and collecting information on the current state of these resources.

In this paper, we discuss several scheduling policies for two level hierarchy: at the first level the broker allocates computational jobs to C-GRID node according to some selection criteria taking parameters of jobs and computers into consideration. At the second level, each node generates schedules by its own local scheduler. We present scheduling strategies based on combination of some selection strategies and scheduling algorithms. We limit our consideration to the scenario where jobs are submitted to the broker from a decentralized environment of other brokers, and can be processed into the same batch. The main objective of the paper is to compare different scheduling strategies and estimate their efficiency. In Section 2, we present a brief overview on two level hierarchy scheduling strategies, and compare their the worst case behavior in Section 3, followed by concluding remarks in Section 4.

2 Scheduling Strategies

2.1 Model

Let we have n jobs $J_1, J_2, ..., J_n$, and m uniform C-GRID nodes $N_1, N_2, ..., N_m$, characterized by $M = [m_1, m_2, ..., m_m]$, where m_i is the number of identical processors of the node N_i. We assume that there is no inter-communications between jobs, that they can be executed at any time, in any order, and on any node.

Each job is described by 2-tuple $(s_j, p_{s_j}^j)$, where s_j is a job size that is referred to as the job's *degree of parallelism* or number of processors required for J_j, $p_{s_j}^j$ is the execution time of job J_j on s_j processors. The job work also called job area is $W_j = p_{s_j}^j \cdot s_j$. Each job can be executed at a single node, so the maximum size of a job is less than or equal to the maximum number of processors in a node. This means that system resources are not crossed, and co-allocation problem is not considered. All strategies are analyzed according to their approximation ratio. Let $C_{\mathrm{opt}}(I)$ and $C_A(I)$ denote makespans of an optimal schedule and of a strategy A for a problem instance I, respectively. The approximation ratio of the strategy A is defined as $\rho_A = \sup_I C_A(I)/C_{\mathrm{opt}}(I)$, and we call A an ρ approximation algorithm.

In this paper, we restrict our analysis to the scheduling systems where all jobs are given at time 0 and are processed into the same batch. This means that a set of available ready jobs is executed up to the completion of the last one. All jobs which arrive in the system during this time will be processed in the next batch. A relation between this scheme and the scheme where jobs arrived over time, either at their release time, according to the precedence constraints, or released by different users is known and studied for different scheduling strategies. Using results [6] the performance guarantee of strategies which allows release times is 2-competitive of the batch style algorithms.

2.2   Two Level Hierarchy Scheduling

The scheduling consists of two parts: selection of a parallel node for a job and then local scheduling at this node.

Selection Strategies. We consider the following scenario. On the first stage, to select a node for the job execution the broker analyzes the job request, and current C-GRID resources' characteristics, such as a load (number of jobs in each local queue), parallel load (sum of jobs' sizes or jobs' tasks), work (sum of jobs work), etc. The parameters of the jobs already assigned to nodes and known by the broker are used only. All nodes are considered in non decreasing order of their sizes m_i, $m_1 \leq m_2 \leq \cdots \leq m_m$. Let $first(J_j)$ be the minimum i such that $m_i \geq s_j$. Let $last(J_j)$ be the maximum i such that $m_i \geq s_j$. If $last(J_j) = m$ we denote the set of nodes N_i, $i = first(J_j), ..., last(J_j)$ as the set of available nodes *M-avail*. If $last(J_j)$ is the minimum r such that $\sum_{i=first(J_j)}^{r} m_i \geq \frac{1}{2} \sum_{i=first(J_j)}^{m} m_i$, we denote the set of nodes N_i, $i = first(J_j), ..., last(J_j)$ as the set of admissible nodes *M-admis*. The broker selects node for a job request using the following strategies:

- *Min-Load (ML)* strategy takes the node with the lowest load per processor (number of jobs over number of processors in the node).
- *Min-Parallel-Load (MPL)* strategy takes the node with the lowest parallel load per processor (the sum of job sizes over number of processors in the node).
- *Min-Lower-Bound (MLB)* strategy chooses the node with the least possible lower bound of completion time of previously assigned jobs, that is the node with the lowest work per processor. Instead of the actual execution time of a job that is an offline parameter, the value provided by the user at job submission, or estimated execution time is used.
- *Min-Completion-Time (MCT).* In contrast to *MLB*, the earliest possible completion time is determined based on a partial schedule of already assigned jobs [9, 7]. For instance, Moab [3] can estimate the completion time of all jobs in the local queue because jobs and reservations possess a start time and a wallclock limit.

Local Scheduling Algorithms. We address the space sharing scheduling problem, hence scheduling can be viewed as a problem of jobs packing into strips of different width. In such geometric model each job corresponds to a rectangle of width s_j and height $p_{s_j}^j$. One known strategy for packing is the *Bottom-Left(BL)*. Each rectangle is slid as far as possible to the bottom and then as far as possible to the left [16]. It is known that for some problems *BL* can not find constant approximation to the optimal packing, but a successful approach is to apply *BL* to the rectangles ordered by decreasing sizes that is referred as *Bottom Left Decreasing(BLD)* or *Larger Size First(LSF)*. In this paper we use *LSF* for local scheduling.

3 Analysis

The *LSF* has been shown to be a 3-approximation [2]. Some results about asymptotic performance ratio of different strategies for this problem and improvements are presented in [1, 4, 5].

Below we will consider *LSF* for local scheduling and the following selection strategies: *Min-Load(ML)*, *Min-Load-admissible(ML-a)*, *Min-Parallel-Load(MPL)*, *Min-Parallel-Load-admissible(MPL-a)*, *Min-Lower-Bound(MLB)*, *Min-Lower-Bound-admissible(MLB-a)*, *Min-Completion-Time(MCT)*, and *Min-Completion-Time-admissible(MCT-a)*.

3.1 (ML, ML-a, MPL, MPL-a)-LSF

The simple example below shows that ML, ML-a, MPL, MPL-a selection strategies combined with *LSF* cannot guarantee constant approximation in the worst case. It is sufficient to consider m nodes of width 1 and the following list of jobs: $m - 1$ jobs J_1, then J_2, then $m - 1$ jobs J_1, etc., where $J_1 = (1, \varepsilon)$, $J_2 = (1, E)$. Suppose $n = rm$, where $r \in \mathbb{N}$. Note that $C_{(\mathrm{ML,MPL})-\mathrm{LSF}} = rE$ and $C_{\mathrm{opt}} \le \lceil \frac{(m-1)r}{m} \rceil \varepsilon + \lceil \frac{r}{m} \rceil E$. If $E/\varepsilon \to \infty$, $m \to \infty$, and $r \to \infty$ then $\rho_{(\mathrm{ML,MPL})-\mathrm{LSF}} \to \infty$.

3.2 MCT-LSF

In the following two theorems we prove that constant approximation for *MCT-LSF* strategy is not guaranteed and that *MCT-a-LSF* is a 10 approximation algorithm.

Theorem 1. *For a set of grid nodes with identical processors and for a set of rigid jobs the constant approximation for MCT-LSF strategy is not guaranteed (in the worst case).*

Proof. Let us consider grid nodes and jobs that are divided into groups according to their sizes. Let there are $k + 1$ groups of nodes and $k + 1$ sets of jobs. The number of nodes in group i is equal $M_i = 2^i$ for $0 \le i \le k$. The number of jobs in a set i is equal $n_i = (i + 1) \cdot 2^i$. The size of the nodes in group i is equal to the job size in the set i, $s_i = m_i = 2^{k-i}$. The execution time (height) of jobs in the set i is $p^i = \frac{1}{i+1}$. Since $p^i n_i s_i / M_i m_i = \frac{1}{i+1}(i+1)2^i/2^i = 1$, obviously $C_{\mathrm{opt}} = 1$.

However, $n_i s_i = 2^{k-i}(i+1) \cdot 2^i = (i+1) \cdot 2^k = \sum\limits_{j=0}^{i} M_j m_j = \sum\limits_{j=0}^{i} 2^j 2^{k-j}$. Hence, any set of jobs may completely fill one layer of available nodes, and if jobs come in increasing order of their sizes $C_{\mathrm{MCT-LST}} = \sum\limits_{j=0}^{k} \frac{1}{j+1} \sim lnk$ that means that the ratio $C_{\mathrm{MCT-LSF}}/C_{\mathrm{opt}}$ may be arbitrary large. □

3.3 MCT-a-LSF

Theorem 2. *For any list of rigid jobs and any set of grid nodes with identical processors the MCT-a-LSF is a 10 approximation algorithm.*

Proof. Let the maximum completion time is achieved at the kth node, J_a be the job that has been received last from the broker by this node, and $f = first(J_a)$, $l = last(J_a)$. Let $Y_f, \ldots, Y_l$ be the sets of jobs that were allocated on the nodes $f, \ldots, l$ (admissible for J_a), just before getting the job J_a. Because the job was sent to the kth node, then the completion time of the node $C_i + p^a \geq C_k + p^a \geq C_{\mathrm{MCT-a-LSF}} \; \forall i = f, \ldots, l$. Let in the node N_k the job with maximum completion time be J_c. Over all jobs with maximum completion time the job with largest processing time is chosen. Let t_c be the time when this job has started the execution, and let $r_c = t_c - p^a$. We have $t_c + p^c \geq C_{\mathrm{MCT-a-LSF}}$, $r_c + p^a + p^c \geq C_{\mathrm{MCT-a-LSF}}$

$$\sum_{i=f}^{l} m_i r_c + p^a \cdot \sum_{i=f}^{l} m_i + \sum_{i=f}^{l} m_i p^c \geq C_{\mathrm{MCT-a-LSF}} \cdot \sum_{i=f}^{l} m_i \tag{1}$$

Before the time t_c the kth node is filled at least half (the property of the BLD algorithm) [2], hence $W_k \geq \frac{1}{2} \cdot m_k \cdot r_c$.

Let J_b be the job which requires minimal number of processors among the jobs allocated on nodes $f, \ldots, l$, and let $f_0 = first(J_b)$. Then all jobs allocated on nodes $f, \ldots, l$ cannot be allocated on nodes N_i with $i < f_0$. Since J_b is allocated on one of the nodes $f, \ldots, l$ then $last(J_b) \geq f \Rightarrow \sum_{i=f_0}^{f-1} m_i \leq \frac{1}{2} \sum_{i=f_0}^{m} m_i \Rightarrow \sum_{i=f}^{m} m_i \geq \frac{1}{2} \sum_{i=f_0}^{m} m_i$. Since $l = last(J_a)$, then $\sum_{i=f}^{l} m_i \geq \frac{1}{2} \sum_{i=f}^{m} m_i$ and

$$\sum_{i=f_0}^{m} m_i \leq 2 \sum_{i=f}^{m} m_i \leq 4 \sum_{i=f}^{l} m_i \tag{2}$$

Thus, $C_{\mathrm{opt}} \cdot \sum_{i=f_0}^{m} m_i \geq S(\bigcup_{i=f}^{l} W_i) \geq \frac{1}{2} \sum_{i=f}^{l} m_i r_i$, where $S(\bigcup_{i=f}^{l} W_i)$ denote the sum of jobs' areas allocated at nodes $f, \ldots, l$. By (2)

$$\sum_{i=f}^{l} m_i r_i \leq 2 \cdot C_{\mathrm{opt}} \cdot \sum_{i=f_0}^{m} m_i \leq 8 \cdot C_{\mathrm{opt}} \cdot \sum_{i=f}^{l} m_i \tag{3}$$

The inequalities $C_{\mathrm{opt}} \geq p^j, \forall j$, (1) and (3) imply

$$8 \cdot C_{\mathrm{opt}} \cdot \sum_{i=f}^{l} m_i + p^a \cdot \sum_{i=f}^{l} m_i + \sum_{i=f}^{l} m_i p^i \geq C_{\mathrm{MCT-a-LSF}} \cdot \sum_{i=f}^{l} m_i,$$

$$8 C_{\mathrm{opt}} \cdot \sum_{i=f}^{l} m_i + p^a \cdot \sum_{i=f}^{l} m_i + C_{\mathrm{opt}} \cdot \sum_{i=f}^{l} m_i \geq C_{\mathrm{MCT-a-LSF}} \cdot \sum_{i=f}^{l} m_i,$$

$$8 C_{\mathrm{opt}} + p^a + C_{\mathrm{opt}} \geq C_{\mathrm{MCT-a-LSF}}, \; 8 C_{\mathrm{opt}} + C_{\mathrm{opt}} + C_{\mathrm{opt}} \geq C_{\mathrm{MCT-a-LSF}},$$

and, finally $C_{\mathrm{MCT-a-LSF}} \leq 10 \cdot C_{\mathrm{opt}}$ □

3.4   MLB-LSF

Theorem 3. *For a set of grid nodes with identical processors, and for a set of rigid jobs the constant approximation for MLB-LSF strategy is not guarantied (in the worst case).*

The proof is similar to the proof of Theorem 1, so we omit it here.

3.5   MLB-a-LSF

The strategy is similar to *MLB-LSF* with only one difference: only admissible nodes are considered for the selection. Selecting admissible nodes prevents narrow jobs filling wide nodes causing wide jobs waiting for execution. It also allows us to find a constant approximation for the algorithm.

Theorem 4. *For any list of rigid jobs and any set of grid nodes with identical processors the MLB-a-LSF is a 10 approximation algorithm.*

Proof. Let the maximum completion time be at the kth node when algorithm terminates. Let the job J_a be the last job with the execution time p^a that was added to this node, and $f = first(J_a)$, $l = last(J_a)$. Let $Y_f, \ldots, Y_l$ be the sets of jobs that had been already allocated at nodes $N_f, \ldots, N_l$ admissible for J_a before adding J_a, and let W_i be the total area of all jobs of Y_i, $(i = f, \ldots, l)$. Since J_a was added to the kth node of width $m_k, \frac{W_k}{m_k} \leq \frac{W_i}{m_i}, \forall i = f, \ldots, l$. Therefore,

$$\sum_{i=f}^{l} W_i = \sum_{i=f}^{l} \frac{W_i}{m_i} m_i \geq \sum_{i=f}^{l} \frac{W_k}{m_k} m_i = \frac{W_k}{m_k} \sum_{i=f}^{l} m_i \qquad (4)$$

Let in packing by the *LSF*(BLD) algorithm, the set of rectangles corresponding to jobs allocated at the kth strip be $Y_k \bigcup \{J_a\}$, J_T be a job with maximum completion time, and t_T be the time when this job has started the execution, hence $C_{\mathrm{MLB-a-LSF}} = t_T + p^T$, where p^T is the processing time of J_T, and $C_{\mathrm{MLB-a-LSF}}$ is the completion time of the *MLB-a-LSF* algorithm. Let $r_k = t_T - p^a$. Then

$$C_{\mathrm{MLB-a-LSF}} = r_k + p^T + p^a \qquad (5)$$

By the property of the *LSF(BLD)* algorithm [2]

$$W_k \geq \frac{1}{2} m_k r_k \Rightarrow r_k \leq \frac{2W_k}{m_k}. \qquad (6)$$

Let J_b be the job having the smallest size among rectangles that packed at strips and let $f_0 = first(J_b)$. Hence any of the rectangles packed at $N_f, \ldots, N_l$ cannot be packed at a strip with number $< f_0$. As far as J_b is packed at one of the strips $f, \ldots, l$, $last(J_b) \geq f \Rightarrow \sum_{i=f_0}^{f-1} m_i \leq \frac{1}{2} \sum_{i=f_0}^{m} m_i \Rightarrow \sum_{i=f}^{m} m_i \geq \frac{1}{2} \sum_{i=f_0}^{m} m_i$. Since $l = last(J_a)$ and $\sum_{i=f}^{l} m_i \geq \frac{1}{2} \sum_{i=f}^{m} m_i$, we obtain $\sum_{i=f_0}^{m} m_i \leq 2 \sum_{i=f}^{m} m_i \leq 4 \sum_{i=f}^{l} m_i$.

Then, clearly $C_{\text{opt}} \sum\limits_{i=f_0}^{m} m_i \geq \sum\limits_{i=f}^{l} W_i$. Substituting (4) in this formula, we have

$C_{\text{opt}} \sum\limits_{i=f_0}^{m} m_i \geq \frac{W_k}{m_k} \sum\limits_{i=f}^{l} m_i \geq \frac{1}{4} \frac{W_k}{m_k} \sum\limits_{i=f_0}^{m} m_i$. Taking into account (6) we obtain

$$r_k \leq \frac{2W_k}{m_k} \leq 8C_{\text{opt}} \tag{7}$$

As far as $C_{\text{opt}} \geq p^j, \forall j$, (5) and (7) imply $8C_{\text{opt}} + C_{\text{opt}} + C_{\text{opt}} \geq C_{\text{MLB}-\text{a}-\text{LSF}}$ and, $C_{\text{MLB}-\text{a}-\text{LSF}} \leq 10 \cdot C_{\text{opt}}$. □

4   Concluding Remarks

In this paper, we discuss approaches and present solutions to multiprocessor job scheduling in computational Grid hierarchical environment that includes a resource broker and a set of clusters or parallel computers. The selection and allocation strategies are discussed. We show that our strategies provide efficient job management with constant approximation guarantee despite they are based on relatively simple schemes. The comparison of *MLB-a-LSF* and *MCT-a-LSF* strategies shows that *MLB-a-LSF* has the same worst case bound as *MCT-a-LSF*, however the *MCT* selection strategy is based on a partial schedule of already scheduled jobs and requires more computational effort than *MLB* strategy that based only on the job parameters from the list of assigned job. With *MLB-a-LSF* the broker can select appropriate node without feedback about the schedule from the node. The results are not meant to be complete, but give an overview on the methodology and some interesting relations. These results motivate finding approximation bounds of other two level hierarchy scheduling strategies. Another interesting question is how fuzzy execution time affects the efficiency. It seems important also to study moldable (or malleable) jobs hierarchical scheduling when the number of processors for a job is not given explicitly by a user but can be chosen by a broker or a local scheduler. Simulations are planned to evaluate proposed strategies considering real and synthetic workload models.

References

1. B. Baker, D. Brown, H. Katseff, A 5/4 algorithm for two-dimensional packing, J. of Algorithms, 1981, v. 2, pp. 348-368.
2. B. Baker, E. Coffman, R. Rivest, Orthogonal packings in two dimensions, SIAM J. Computing, 1980, v. 9, 4, pp. 846-855.
3. www.clusterresources.com
4. K. Jansen, Scheduling malleable parallel jobs: an asymptotic fully polynomial-time approximation scheme, Euro. Symp. on Algorithms, 2002.
5. C. Kenyon, E. Remila, A near optimal solution to a two dimensional cutting stock problem, Math. of Operations Res., 25 (2000), 645-656.

6. D. Shmoys, J.Wein, D.Williamson. Scheduling parallel machines on-line. SIAM J. Comput., 24:1313-1331, 1995.
7. S.Zhuk, A.Chernykh, N.Kuzjurin, A.Pospelov, A.Shokurov, A.Avetisyan, S.Gaissaryan, D.Grushin. Comparison of Scheduling Heuristics for Grid Resource Broker. PCS2004 Third International Conference on Parallel Computing Systems (in conjunction with ENC'04), IEEE, p. 388-392. 2004
8. Foster, C. Kesselman, editors. The Grid: Blueprint for a future computing infrastructure, Morgan Kaufmann, San Fransisco, 1999.
9. G. Sabin, R. Kettimuthu, A. Rajan, and P. Sadayappan, Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment, in the Proceedings of the 8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP), 2003.
10. L. Smarr and C. Catlett. Metacomputing. Communications of the ACM, 35(6): 44-52, June 1992.
11. S. S. Vadhiyar and J. J. Dongarra, "A Metascheduler for the Grid," Proc. of 11-th IEEE Symposium on High Performance Distributed Computing (HPDC 2002), July 2002.
12. J. Gehring and A. Streit, "Robust Resource Management for Metacomputers," In Proc. HPDC '00, pages 105-111, 2000.
13. V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. Evaluation of Job-Scheduling Strategies for Grid Computing. R. Buyya and M. Baker (Eds.) In Proc. Grid 2000, LNCS 1971, pp. 191-202, 2000.
14. A. James, K. A. Hawick, and P. D. Coddington, "Scheduling Independent Tasks on Metacomputing Systems," In Proc. Conf. on Parallel and Distributed Systems, 1999.
15. The Grid Forum, http://www.gridforum.org/
16. E. Hopper, B. C. H. Turton, An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, Europian Journal of Operational Research, 2001.