Comparison of Scheduling Heuristics for Grid Resource Broker

Sergey Zhuk
MIPT, Moscow
zh_sergey@mail.ru

Andrey Chernykh
CICESE Research Center,
Ensenada, Mexico
chernykh@cicese.mx

Arutyun Avetisyan, Sergey Gaissaryan,
Dmitry Grushin
ISP RAS, Moscow
{arut, ssg, grushin}@ispras.ru

Nikolai Kuzjurin, Alexey Pospelov,
Alexander Shokurov
ISP RAS, Moscow
{nnkuz, pospelov, shok}@ispras.ru

Abstract

In this paper, we consider parallel tasks scheduling problems for hierarchical decentralized systems that consist of homogeneous computational resources such as clusters, PCs and supercomputers, and geographically dispersed. We concentrate on two-level hierarchy scheduling: at the first level, broker allocates computational tasks to the resource. At the second level, each resource schedules the tasks assigned to it using heuristics based, for instance, on strip-packing algorithms. The allocation strategies and efficiency of proposed hierarchical scheduling algorithms are discussed.

1. Introduction

Computational Grids [9] are emerging as a new paradigm for solving large-scale problems in science, engineering, and commerce. They enable the sharing and aggregation of millions of resources, geographically distributed across organizations and administrative domains [10]. They comprise heterogeneous resources (PCs, work-stations, clusters, and supercomputers), policies, and applications (scientific, engineering, and commercial) with varied requirements. The resources are owned by different organizations with their own management policies, usage and cost models. Moreover, the availability of resources and their load dynamically vary in time. The service producers (resource owners) and service consumers (resource users) have different goals, objectives, strategies, and supply-and-demand patterns. Therefore, a very important problem is to have an efficient resource management system intended to optimize relations between producers and consumers according to some adequate resource management strategy.

Most of the systems for Grid resource management and scheduling (such as Legion, Condor, AppLeS PST, NetSolve, PUNCH, XtremWeb, etc.) adopt a conventional strategy, where a scheduler selects jobs are to be executed on a particular resource based on objective cost functions driven by system-centric parameters. These schedulers aim to enhance the system throughput, utilization, and complete execution at the earliest possible time rather than to improve application processing. Two key players: resource providers and resource consumers; have their own expectations and strategies. Resource providers use strategies of good investment and try to maximize resource utilization, while resource consumers adopt the strategies of solving their problems within a required time-frame and budget. Also it should be taken into account that a user is in competition with other users, and a resource owner is in competition with other resource owners.

The resource consumer interacts with brokers to express requirements such as the budget to solve a given problem, its deadline, and possible trade-off these two requirements. He can has an option of what provider best meets his requirements. The resource providers also need tools for expressing their pricing policies, and mechanisms that help them to maximize the profit and resource utilization.

Rajkumar Buyya [6] proposed an economic-based approach, when scheduling decisions are made dynamically at runtime and they are driven by the end-users requirements. Such an approach is implemented in GRACE system. In such an approach, based on a specific metric of the price, management systems dynamically bargain about available resources, and schedule computations on these resources to meet user requirements.

Computations as objects with their own goals, resources,

and actions, can be viewed as an economic object. With the proliferation of networks, when high-end computing systems has moved from using centralized models of control and action toward decentralized ones, the projection of economic driven market mechanisms to resource management would be a natural.

The resource broker consists of the following components:

- Job Control Agent: it is a persistent control engine responsible for shepherding a job through the system. It coordinates with schedule adviser for schedule generation, handles actual creation of jobs, maintenance of job status, interacting with clients/users, schedule adviser, and dispatcher.
- Schedule Adviser: it is responsible for resource discovery (using the Grid explorer), resource selection and job assignment (schedule generation) to ensure that the user requirements are met.
- Grid Explorer: it is responsible for resource discovery by interacting with the Grid-information server and identifying the list of authorized machines, and keeping track of resource status information.
- Deployment Agent: it is responsible for activating task execution on the selected resource as per the scheduler's instruction and periodically updates the status of task execution to Job Control Agent.

The Grid resource broker acts as a mediator between the user and Grid resources using middleware services. It is responsible for resource discovery, resource selection, binding of software, data, and hardware resources, initiating computations, adapting to the changes in Grid resources, and presenting the Grid to the user as a single, unified resource.

In this paper, we address the problem of scheduling parallel jobs in such systems. We consider two-level hierarchy: at the first level the broker allocates computational tasks to clusters according some criteria. At the second level, each cluster schedules the parallel tasks assigned to it by its own local scheduler. Such local schedulers can use heuristics, in particular based on strip-packing algorithms. Here we do not consider implementation issue and issues related with resource discovery and selection. The main objective is to compare different scheduling strategies and estimate their efficiency.

We restrict our analysis to the scheduling systems where all the tasks are given at time 0 and are processed into the same batch. That means that a set of available ready tasks will be executed up to the completion of the last one. All tasks which arrive in the system during this time will be processed in the next batch. A relation between this scheme and the scheme where tasks 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 for general or restricted cases [21]. Using most basic results, the performance guarantee of our strategies which allows release times is 2-competitive of the batch style algorithms discussed in the paper.

2. Scheduling strategies

We consider the following simple model. Let we have n tasks and m clusters with identical processors. We assume that all tasks are independent, i.e. there is no communications between tasks. Each task can be described by a pair: the size p_j (number of requested processors), and execution time t_j on a p_j processors. Let w_i be the number of processors of the ith cluster. Each task can be run on some cluster, so that the maximum size of the task is less than the maximum number of processors in a cluster ($\max_j p_j \leq \max_i w_i$). Our objective is to minimize total completion time.

Scheduling multiprocessor tasks with unit processing times is a strongly NP-hard problem [1], and can be considered as a strip-packing problem where a set of rectangles (size-time task rectangles) must be packed into a strip of limited width and unbounded height. One of the known heuristics is the Bottom-Left (BL): put a rectangle as low as possible, and move it to the left. It is known that for some problems BL can not find optimal packing [4, 19], nor does it perform well in practice when applied for random ordering (that is a case for heuristics H_1 described below). However, a very successful approach is to apply BL to the tasks ordered by decreasing width (the case of heuristics H2-4 described below) that is referred as Bottom Left Decreasing (BLD). The BLD has been shown to be a 3-approximation [4]. Some results about asymptotic performance ratio of different heuristics for this problem and recent improvements are presented in [3, 4, 8, 14, 15, 16, 18]. In particular, in [8] it was shown that algorithms where the rectangles are placed on "shelves" using one-dimensional bin-packing heuristics have asymptotic performance ratio of 2.7 when the rectangles are sorted by decreasing height (First-Fit shelf algorithms). The asymptotic performance ratio of the best heuristic was further reduced to 2.5 in [18], then to 4/3 (see [15]), and finally to 5/4 [3]. Recent improvements are presented in [14, 15]. Scheduling tasks on a set of clusters can be viewed as a packing problem of tasks into a few strips of different width. We consider the following scenario. On the first stage, Broker analyzes a task request, and broadcasts task parameters to all clusters. Each cluster returns an estimation of the corresponding completion time. Broker assigns the task to the cluster that has minimum estimated completion time. On the second stage, clusters allocate tasks on the processors using the BL heuristic. In this

paper, we consider different algorithms for these two stages, and estimate the quality of such hierarchical scheduling. Now, we present different scheduling strategies where allocation of tasks on processors (by a cluster local scheduler) is performed by the BL or BLD heuristics.

Heuristic 1: MCT+BL

On the first stage, the broker sends the task description to all clusters. Each cluster returns an estimation of the completion time. Broker allocates the task on a cluster that provides a minimum completion time (MCT). On the second stage, the task is executed on the cluster using the BL heuristic. Allocation and execution are performed in the order tasks arrived.

Heuristic 2: MCT-SORT+BL

On the first stage, the broker sorts all available tasks by decreasing their size (widths), and then it functions similar to MCT+BL.

Heuristic 3: MCT+BLD

On the first stage, the broker functions similar to MCT+BL, and then to execute tasks the BLD strategy is used

3. Comparison of scheduling strategies

We use the following notations.

$T = \{T_1, \ldots, T_n\}$ is the set of tasks,
$h(T_j)$ execution time for task T_j,
$w(T_j)$ is the task size (the number of required processors) by T_j,

$C = \{C_1, \ldots, C_m\}$ the set of clusters,
w_i is the size (width) of the ith cluster.
We will assume that the clusters are sorted by their widths, i.e.

$$w_1 \le w_2 \le \cdots \le w_m.$$

Let H_O be the minimum completion time of a given list of tasks (that is optimum over all possible schedules). Let H_i be the completion time obtained by the ith heuristics (i=1,2,3) described above. It can be shown that the ratio H_1/H_O (the approximation ratio) can not be bounded by a constant. To show this, it is sufficient to consider only one cluster of width w and the following list of tasks $\{T_1, T_2, T_1, T_2, \ldots\}$, where $w(T_1) = w$, $h(T_1) = \varepsilon$, $w(T_2) = 1$, $h(T_2) = H$, and H is sufficiently large.

For Heuristic 2 the constant approximation ratio for the worst case has been proved.

Theorem 1 [19]. For any list of tasks and any set of clusters the following inequality holds

$$H_2/H_O \le 3.$$

This constant approximation ratio that Heuristic 2 can guarantee in the worst case is a very desirable situation. A drawback of the Heuristic 2 is the requirement for a broker to do a preprocessing step (sorting tasks) that looks slightly unrealistic (a broker can manage a lot of clusters). It is more reasonable to put this problem on clusters' side and to use Heuristic 3, so that the broker will manage tasks on-line.

It is interesting whether the analogue of Theorem 1 could be proved for the ratio H_3/H_O? In some cases (all jobs' widths are small enough) it can be shown, that the ratio is bounded by a constant. For example, the following theorem can be proved

Theorem 2. For any list of tasks and any set of clusters such that for any j $w(T_j) \le w_1$ the following inequality holds

$$H_3/H_O \le 4.$$

For the general case, the analogue of Theorem 1 can not be proved that can be shown by the following example.

Example. Let the clusters and the tasks are divided into groups according to their widths. Let there are $n+1$ groups of width w_i, $i = 0, 1, \ldots, n$. We use the following notations

- $n+1$ is the number of groups of machines (tasks),
- $M_i, i = \overline{0,n}$ — the number of machines in group i,
- $N_i, i = \overline{0,n}$ — the number of tasks in group i,
- $W_i, i = \overline{0,n}$ — width of machines (tasks) in group i,
- $h_i, i = \overline{0,n}$ — height of tasks in group i,
- H_O — height of optimal packing,
- H_3 — height of tasks obtained by Heuristic 3.

The instance below shows that the ratio H_3/H_O may be arbitrary large. Let

$$\begin{aligned} M_i &= 2^i, && i = \overline{0,n}, \\ N_i &= (i+1)\cdot 2^i, && i = \overline{0,n}, \\ W_i &= 2^{n-i}, && i = \overline{0,n}, \\ h_i &= \frac{1}{i+1}, && i = \overline{0,n}. \end{aligned}$$

Let tasks come in increasing order. Since

$$\frac{h_i N_i W_i}{M_i W_i} = \frac{\frac{1}{i+1}(i+1)2^i}{2^i} = 1, \quad i = \overline{0,n}.$$

obviously

$$H_O = 1.$$

But

$$N_i W_i = 2^{n-i}(i+1)\cdot 2^i = (i+1)\cdot 2^n =$$

$$\sum_{k=0}^{i} M_k W_k = \sum_{k=0}^{i} 2^k 2^{n-k}.$$

Hence, any group of tasks may completely fill one layer on all clusters, and

$$H_3 = \sum_{k=0}^{n} \frac{1}{k+1} \sim \ln n,$$

that means the ratio H_3/H_O may be arbitrary large.

Heuristic 3 can be modified to achieve constant approximation ratio. Let us consider the following modification.

$first(T_j)$ be the minimum i: $w_i \geq w(T_j)$,

$last(T_j)$ be the minimum r such that

$$\sum_{i=first(T_j)}^{r} w_i \geq \frac{1}{2} \sum_{i=first(T_j)}^{m} w_i$$

H_4 the completion time by Heuristic 4.

$BL_i(W)$ packing of the set of tasks W on the ith cluster obtained by the BLD-algorithm.

$S(W)$ area of all tasks from the set W.

Clusters $first(T_j)$, $first(T_j)+1, \ldots, last(T_j)$ we will call **admissible** for the task T_j.

Heuristic 4. On the first stage broker functions as in Heuristic 3 with only one difference: It sends a task description to **admissible** (for this task) clusters only. Every of these clusters uses BLD-algorithm and broker chooses the cluster which gives minimum completion time.

Theorem 3. For any list of tasks and any set of clusters

$$\frac{H_4}{H_O} \leq 10.$$

Proof. Let maximum completion time is given by kth cluster, and let T_a be the task that has been received last from the broker by this cluster, and

$$f = first(T_a), \qquad l = last(T_a).$$

Let $W_f, \ldots, W_l$ be the sets of tasks that were allocated on the clusters $f, \ldots, l$ (admissible for T_a), just before getting the task T_a. Because the task was sent to the kth cluster, then

$$BL_i(W_i \cup \{T_a\}) \geq H_4 \quad \forall i = f, \ldots, l$$

Let in $BL_i(W_i \cup \{T_a\})$ the task with maximum completion time be T_{c_i}. Let $h_i = h(T_{c_i})$, t_i be the time when this task has started the execution, and $r_i = t_i - h(T_a)$. We have

$$t_i + h_i \geq H_4 \quad \forall i = f, \ldots, l$$

$$r_i + h(T_a) + h_i \geq H_4 \quad \forall i = f, \ldots, l$$

$$\sum_{i=f}^{l} w_i r_i + h(T_a)\cdot \sum_{i=f}^{l} w_i + \sum_{i=f}^{l} w_i h_i \geq H_4 \cdot \sum_{i=f}^{l} w_i. \quad (1)$$

Before the time t_i the ith cluster is filled at least half (Property of the BL-algorithm). Hence,

$$S(W_i) \geq \frac{1}{2}\cdot w_i \cdot r_i.$$

Let T_b be the task which requires minimal number of processors among the tasks allocated on clusters $f, \ldots, l$, and let $f_0 = first(T_b)$. Then all tasks allocated on clusters $f, \ldots, l$ cannot be allocated on clusters C_i with $i < f_0$.

Since T_b is allocated on one of the clusters $f, \ldots, l$ then

$$last(T_b) \geq f \implies \sum_{i=f_0}^{f-1} w_i \leq \tfrac{1}{2} \sum_{i=f_0}^{m} w_i \implies \sum_{i=f}^{m} w_i \geq \tfrac{1}{2} \sum_{i=f_0}^{m} w_i$$

Since $l = last(T_a)$, then

$$\sum_{i=f}^{l} w_i \geq \frac{1}{2} \sum_{i=f}^{m} w_i,$$

and

$$\sum_{i=f_0}^{m} w_i \leq 2 \sum_{i=f}^{m} w_i \leq 4 \sum_{i=f}^{l} w_i. \quad (2)$$

Thus,

$$H_O \cdot \sum_{i=f_0}^{m} w_i \geq S(\bigcup_{i=f}^{l} W_i) \geq \frac{1}{2} \sum_{i=f}^{l} w_i r_i,$$

and by (2)

$$\sum_{i=f}^{l} w_i r_i \leq 2 \cdot H_O \cdot \sum_{i=f_0}^{m} w_i \leq 8 \cdot H_O \cdot \sum_{i=f}^{l} w_i. \quad (3)$$

The inequalities $H_O \geq h(T_j)$, $\forall j$, (1) and (3) imply

$$8 \cdot H_O \cdot \sum_{i=f}^{l} w_i + h(T_a)\cdot \sum_{i=f}^{l} w_i + \sum_{i=f}^{l} w_i h_i \geq H_4 \cdot \sum_{i=f}^{l} w_i.$$

$$8 \cdot H_O \cdot \sum_{i=f}^{l} w_i + h(T_a)\cdot \sum_{i=f}^{l} w_i + H_O \cdot \sum_{i=f}^{l} w_i \geq H_4 \cdot \sum_{i=f}^{l} w_i.$$

$$8 \cdot H_O + h(T_a) + H_O \geq H_4.$$

$$8 \cdot H_O + H_O + H_O \geq H_4,$$

and, finally

$$\frac{H_4}{H_O} \leq 10.$$

4. Concluding remarks

In this paper, we discuss approaches for multiprocessor task scheduling on computational Grid. We present solutions for hierarchical systems that include broker and clusters. The worst case analysis shows that our strategies can provide efficient task management with constant guarantee despite they are based on relatively simple (in fact, on-line) heuristic algorithms.

There are many problems left for future research. From the practical point of view, it is interesting to study the scheduling algorithms on different sets of real instances. These results motivate finding average case approximation bounds that can better predict the algorithms behavior.

It is also interesting to study different algorithms for the local cluster scheduling, other than BL and BLD, say, shelf or split algorithms, that can improve the total approximation quality. Another interesting questions is: how fuzzy execution time affects the efficiency? It seems important also to study malleable tasks hierarchical scheduling when the number of processors for a task is not given explicitly by a user but can be chosen by a broker.

Acknowledgment

The authors would like to thank the Russian Foundation for Basic Research for partial support of this work (grant 02-01-00713).

References

[1] Abraham, R. Buyya, B. Nath, Nature's heuristics for scheduling jobs on Computational Grids, International Conference on Advanced Computing and Communications (2000).

[2] P. Brucker, Scheduling Algorithms, Springer Ferlag (1998), 217-218.

[3] B.S. Baker, D.J. Brown and H.P. Katseff, A 5/4 algorithm for two-dimensional packing, J. of Algorithms, 1981, v. 2, pp. 348-368.

[4] B.S. Baker, E.J. Coffman and R.L. Rivest, Orthogonal packings in two dimensions, SIAM J. Computing, 1980, v. 9, pp. 846-855.

[5] D.J.Brawn. An improved BL lower bound, Information processing Letter. 11, pp. 37-38, 1980.

[6] R. Buyya, D. Abramson, J. Giddy, An Economy Driven Resource Management Architecture For Global Computational Power Grids, International Conference on Parallel and Distributed Processing Techniques and Applications, 2000.

[7] L. Hluchy, V.D. Tran, D. Froehlich, and W. Castaings, Methods and Experiences of Parallelizing Flood Models, The 10th EuroPVM/MPI conference. LNCS 2840. Sept. 2003, Venice. pp. 677-681.

[8] E.J. Coffman, M.R. Garey, D.S. Johnson and R.E. Tarjan, Performance bounds for level-oriented two-dimensional packing algorithms, SIAM J. Computing, 1980, v. 9, pp. 808-826.

[9] The Global Grid Forum. http://www.gridforum.org

[10] Foster, C. Kesselman, editors. The Grid: Blueprint for a future computing infrastructure, Morgan Kaufmann, San Fransisco, 1999.

[11] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, Grid Information Services for Distributed Resource Sharing, 10th IEEE International Symposium on High-Performance Distributed Computing, 2001.

[12] M. Drozdowski, Scheduling multiprocessor tasks - an overview, European J. of Oper. Research, 1996, v. 94, pp. 215-230.

[13] S. Orlando, P. Palmerini, R. Perego, F. Silvestri, Scheduling high performance data mining tasks on a data Grid environment, Euro-Par 2002, LNCS 2400, Springer-Ferlag Berlin Heidelberg 2002, pp. 375-384.

[14] K. Jansen, Scheduling malleable parallel tasks: an asymptotic fully polynomial-time approximation scheme, Proc. European Symposium on Algorithms, ESA, 2002.

[15] C. Kenyon and E. Remila, A near optimal solution to a two-dimensional cutting stock problem, Mathematics of Operations Research, 25 (2000), 645-656.

[16] W. Ludwig and P. Tiwari, Scheduling malleable and nonmalleable parallel tasks, Proc. 5th ACM-SIAM Symposium on Discrete Algorithms, SODA (1994), 167-176.

[17] S. Orlando, P. Palmerini, R. Perego, F. Silvestri, Scheduling high performance data mining tasks on a data Grid environment, Euro-Par 2002, LNCS 2400, Springer-Ferlag Berlin Heidelberg 2002, pp. 375-384.

[18] D.D.Sleator, A 2.5-times optimal algorithm for bin packing in two dimensions, Inf. Processing Letters, 1980, v. 10, pp. 37-40.

[19] A.I. Pospelov, On a problem of packing rectangles into a set of strips, Proc. of the Institute for System Programming, v. 6, 2004 (to appear).

[20] N. Sample, P. Keyani, G. Wiederhold, Scheduling under uncertainty: Planning for the Ubiquitois Grid, International Conference on Coordination models and languages (2002).

[21] D. Shmoys, J. Wein, D. Williamson. Scheduling parallel machines on-line. SIAM J. Comput., 24:1313-1331, 1995.