A Compendium of Heuristic Methods for Scheduling in Computational Grids

Fatos Xhafa[1] and Ajith Abraham[2]

[1] Department of Languages and Informatics Systems
Technical University of Catalonia, Barcelona, Spain
fatos@lsi.upc.edu
[2] Center of Excellence for Quantifiable Quality of Service
Norwegian University of Science and Technology, Norway
ajith.abraham@ieee.org

Abstract. Scheduling in large scale distributed computing environments such as Computational Grids, is currently receiving a considerable attention of researchers. Despite that scheduling in such systems has much in common with scheduling in traditional distributing systems, the new characteristics of Grid systems make the problem more complex and versatile to match different needs of Grid-enabled applications. In this work, by conceiving scheduling problem as a family of problems, we first identify most common versions of the scheduling problem based on six dimensions: type of the environment, architecture type of the scheduler, immediacy of the processing, type of interrelations among tasks, type of preemptive policy and type of optimization model. Then, we review different families of heuristic methods used for the resolution of the problem, including ad hoc methods, local search methods and population-based methods.

1 Introduction

The fast development and deployment of Grid systems world-wide are increasingly raising the need for the design of efficient Grid schedulers. Grid systems are expected to leverage unprecedented larger computing capacities by virtually joining together geographically distributed resources at large scale. To achieve this objective, scheduling of tasks and applications onto Grid resources is a key issue. Scheduling in Grid systems can be seen as the predecessor of Grid scheduling. Thus, computational models for formalizing the problem as well as methods for its resolutions are quite often borrowed from the scheduling in conventional distributed systems. However, Grid systems introduce new complexities to the problem due to their intrinsic characteristics. Also, the versatility of Grid systems and that of user's tasks and applications impose new restrictions and requirements on Grid schedulers. In fact, the problem becomes even more complex at the modelling phase, which requires the selection of a careful setting and computational model to match the real Grid system setting.

In this paper we first identify the most important new characteristics of Grid systems that influence the complexity of the problem. Then, we concentrate on

E. Corchado and H. Yin (Eds.): IDEAL 2009, LNCS 5788, pp. 751–758, 2009.

the Grid scheduling as a family of problems. Often, the scheduling problem in the Grid computing literature is vaguely defined and our observations prompt for precise definitions of the problem by identifying the main dimensions of the problem. Finally, we briefly review the heuristic methods reported in the literature for the case of scheduling independent tasks in Grid systems.

The rest of the paper is organized as follows. In Section 2, we briefly present the characteristics of Grid systems that increase the complexity of scheduling problems as compared to scheduling in traditional distributed systems. Scheduling as a family of problems is presented in Section 3 based on six dimensions: type of the environment, architecture type of the scheduler, immediacy of the processing, type of interrelations among tasks, type of preemptive policy and type of optimization model. Next, in Section 4, we give the most commonly used heuristic methods for the resolution of Grid scheduling problems. We end the paper in Section 5 with some conclusions.

2 Characteristics of Grid Systems

The scheduling problem in distributed systems is one of the most studied problems in the optimization research community. The intrinsics characteristics of Grid systems add new complexities to the problem as compared to its traditional version of conventional distributed systems. We briefly consider below some of these characteristics:

The dynamic structure of the Grid systems. Unlike traditional distributed systems such as clusters, resources in a Grid system can join or leave the Grid in an unpredictable way.

The *high heterogeneity of resources*. Grid systems act as large virtual supercomputers, yet the computational resources could be very disparate in their computing capacities.

The *high heterogeneity of tasks*. Tasks arriving to any Grid system are diverse and heterogenous in terms of their computational needs.

The *high heterogeneity of interconnection networks.* Grid resources will be connected through Internet using different interconnection networks and time needed for data transmission should be taken into account.

The *existence of local schedulers* in different organizations or resources. Grids are cross-domain platforms and particular domains could account for using their own local schedulers. In such case, Grid schedulers should co-exists and be used in conjunction with local schedulers.

The *existence of local policies on resources*. Again, due to the cross-domain nature of Grid systems, local policies on access and usage of resources are also to be taken into account.

Task-resource restrictions. Many restrictions and/or incompatibilities could be derived from task and resource specifications, which Grid schedulers should take into account.

Scalability and efficiency. Grid schedulers should be scalable since Grid systems are expected to be large scale. Different types of schedulers (super-schedulers,

meta-schedulers, decentralized schedulers, local schedulers, etc.) should be considered in order to achieve scalability and efficiency.

Security. This characteristic, which is not a requirement in classical scheduling, is an important issue in Grid scheduling, although not yet fully addressed in the current Grid schedulers.

3 Family of Scheduling Problems in Grid Systems

Scheduling problem in Grid systems, in its general form, can be stated as in the case of scheduling problem for distributed systems: compute a mapping of a given set of tasks onto a set of available machines. However, behind this general definition, there are many concrete versions of the problem. Essentially, these concrete versions are obtained by considering the following six dimensions.

Type of the environment: this dimension refers to the dynamics of the system being *static* and *dynamic* two most common settings. The former assumes that all resources are available and fixed in number while the later considers that resource availability can vary over time. For instance, in Cluster Grids, which provide computing services to a group or institution level, the environment could be considered static. Or, in the case of Enterprise Grids, enabling multi-departmental access to shared resources within an enterprise or campus. However, in global Grids one should assume the dynamics of the resources as an important requirement for the Grid scheduler.

Architecture type of the scheduler: Grid schedulers can be organized in different ways, being the most used architectures the centralized and decentralized ones. This characteristics essentially depends on the knowledge and control over resources. Centralized schedulers assume full knowledge and control over resources while decentralized ones assume partial knowledge of available resources. There is also the hierarchical architecture which combines features of both centralized and decentralized schedulers.

Immediacy of the processing: this dimension indicates whether the scheduler should immediately or in batch mode. In fact, in a global setting, both modes could co-exists since tasks or applications are submitted independently from many users and some could have the immediate processing requirements while others, most commonly periodic tasks or large applications, could be processed in batches.

Type of interrelations among tasks: there could be dependencies among tasks spawned by applications, requiring a workflow which the scheduler should take into account. One common scenario in large Grid systems is the submission of independent tasks or applications, therefore, independent task scheduling is particularly important for Grid systems.

Type of preemptive and re-scheduling policy: tasks or applications can be processed in preemptive or non-preemptive mode. In the first, a task can be temporarily interrupted while in the second, a task processing cannot be interrupted until it is finished. It should be noted that the dynamics of Grid systems could

cause premature interruption of task processing due to resource drop from the system. In such case, Grid schedulers, especially in case of existence of task dependencies, should include re-scheduling policies, which could be to either resume the task and submit it to another available resource or send it to task pool and schedule it again as a new incoming task. Rescheduling can thus be seen as a form of adaptive scheduling.

Type of optimization model: this dimension refers to single *vs.* multi-objective optimization model. In classical version of scheduling, usually a few objectives are considered, most commonly makespan. However, in a Grid context a larger number of objectives should be considered, including: makespan, flowtime, resource utilization, load balancing, matching proximity, turnaround time, total weighted completion time, lateness, weighted number of tardy tasks, weighted response time, resource owner's benefits, etc. Thus, scheduling problem is multi-objective in nature. It should also be noticed that some criteria are conflicting among them, especially close to optimality.

Thus, a precise definition of Grid scheduling problem depends on the concrete setting of each dimension above as well as their combinations.

Another issue is how all the above dimensions are formalized into an optimization problem. This depends on the computational model used such as Expected Time To Compute (ETC model), Total Processor Cycle Consumption model (TPCC model), etc. (See Xhafa and Abraham [10] for more details.)

To exemplify the different types of scheduling problem that can be obtained by fixing the setting of the different dimensions, we describe next the version of Grid scheduling known as "scheduling of independent tasks".

Scheduling of independent tasks in computational grids. This version of the problem is obtained by considering the following setting:

- *type of the environment*: dynamic
- *architecture type of the scheduler*: centralized
- *immediacy of the processing*: batch mode
- *type of interrelations among tasks*: independent tasks
- *type of preemptive and re-scheduling policy*: non-preemptive; re-scheduling as a new incoming task, in case of resource drop.
- *type of optimization model*: four-objective optimization model: makespan, flowtime, resource utilization and matching proximity.

The problem is formalized using the Expected Time To Compute matrix (ETC model) in which $ETC[j][m]$ indicates an estimation of how long will it take to complete task j in resource m. Under the ETC matrix model, the independent task scheduling can be defined as follows:

- A number of independent *tasks* to be allocated to Grid resources in non-preemptive mode.
- A number of *machines* candidates to participate in the allocation of tasks.
- The *workload* (in millions of instructions) of each task.

- The *computing capacity* of each machine (in *Mips*).
- The ready times indicating when machines will have finished the previously assigned tasks.
- The ETC matrix of size $nb\_tasks \times nb\_machines$, where $ETC[j][m]$ is the value of the expected time to compute task j in machine m.

The optimization objectives can be formalized under ETC model using the completion time (see [10] for their formulae):

$$completion[m] = ready\_times[m] + \sum_{\{j \in Tasks \mid schedule[j]=m\}} ETC[j][m].$$

4 Heuristic Methods for Scheduling in Grid Systems

At a higher level, the heuristic methods are usually classified as calculus based, random or enumerative. In the first group we can find greedy algorithms and ad hoc methods, which implement a direct strategy for computing the solution. In the second class, which is the largest one, we have guided and non-guided methods. The former includes Simulated Annealing, Tabu Search, and the large family of evolutionary algorithms. The final group comprises dynamic programming and branch-and-bound algorithms.

In the case of Grid scheduling problem, many methods from the aforementioned groups, have been applied to the problem. We list below the most used methods for the problem for the two groups, namely, calculus and random comprising ad hoc, local search-based and population-based methods.

4.1 Ad Hoc Methods

Ah hoc methods comprise both immediate and batch scheduling [5,9,3,1] and are usually used for single-objective optimization case.

Immediate mode methods. This group includes Opportunistic Load Balancing, Minimum Completion Time, Minimum Execution Time, Switching Algorithm and k-Percent Best.

Opportunistic Load Balancing (OLB). This method assigns a task to the earliest idle machine without taking into account the execution time of the task in the machine.

Minimum Completion Time (MCT). This method assigns a task to the machine yielding the earliest completion time. Upon a task arrival, all available resources are examined to determine the resource that yields the smallest completion time for the task.

Minimum Execution Time (MET). This method assigns a task to the machine having the smallest execution time for that task. Thus tasks are allocated to resources that best fit them with regard to the execution time.

Switching Algorithm (SA). This method combines MET and MCT cyclically based on the workload of resources. It uses MET till a threshold is reached and then use MCT to achieve improved load balancing.

k-Percent Best. For a given task, this method considers the candidate set of $nb\_machines \cdot k/100$ best resources (w.r.t. execution times). The machine from the candidate set yielding the earliest completion time is chosen.

Batch mode methods. This group comprises Min-Min, Max-Min, Sufferage, Relative Cost and Longest Job to Fastest Resource - Shortest Job to Fastest Resource.

Min-Min. For any task i, the machine m_i yielding the earliest completion time is computed by traversing the ith row of the completion matrix. Then, the task i_k with the earliest completion time is chosen and mapped to the corresponding machine m_k.

Max-Min. This method is similar to Min-Min. The difference is that, for any task i, once the machine m_i yielding the earliest completion time is computed, the task i_k with the latest completion time is chosen and mapped to the corresponding machine.

Sufferage. The idea behind this method is that better scheduling could be obtained if we assign to a machine a task, which would "suffer" more if it were assigned to any other machine.

Relative Cost. In allocating tasks to machines, this method takes into account both the load balancing of machines and the execution times of tasks in machines.

Longest Job to Fastest Resource - Shortest Job to Fastest Resource (LJFR-SJFR). This method tries to simultaneously minimize both makespan and flowtime values: LJFR minimizes makespan and SJFR minimizes flowtime.

A comparative evaluation of both immediate and ad hoc methods has been done in Xhafa et al. [16,13].

4.2 Local Search Methods

Local search is a family of methods that explore the solution space starting from an initial solution and constructing a path in solution space. Methods in this family range from simple ones such as Hill Climbing, Simulated Annealing to more sophisticated ones such as Tabu Search method.

Simple local search methods (Hill Climbing-like) have been studied for the scheduling under ETC model in Ritchie and Levine [8]. Xhafa [11] used several local search methods in implementing Memetic Algorithms for the same problem.

Simulated Annealing (SA) is more powerful than simple local search by accepting also worse solutions with certain probability. This method has been proposed for Grid scheduling by Abraham et al. [1] and Yarkhan and Dongarra [19].

Tabu Search (TS) is a more sophisticated but also more computationally expensive due to its mechanisms of tabu lists, aspiration criteria, intensification and diversification. Abraham et al. [1] considered TS as candidate solution method for the problem. Ritchie [7] implemented the TS for the problem under ETC model and used it in combination with ACO approach. Xhafa et al. [14] has presented a full featured TS for the scheduling problem under ETC model.

4.3 Population-Based Methods

Population-based heuristics is a large family of methods that use populations of individuals to explore the solution space. Although they usually require large running times, they are quite efficient when the objective is to find feasible solutions of good quality in short times, as in case of Grid scheduling. This family comprises Genetic Algorithms (GAs), Memetic Algorithms (MAs), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO).

Evolutionary algorithms. GAs for Grid scheduling have been addressed by Abraham et al. [1], Braun et al. [3], Zomaya and Teh [20], Page and Naughton [6], Gao et al. [4], Xhafa et al. [15,17]. MAs is class of population-based methods, which combine the concepts of evolutionary search and local search. Xhafa [11] applied unstructured MAs and Xhafa et al. [12] proposed Cellular MAs (structured MAs) for the independent scheduling problem under ETC model.

Swarm intelligence methods Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) have also been considered for the scheduling problem. An ACO implementation for the problem under ETC model has been reported by Ritchie [7]. Abraham et al. [2] proposed an approach using fuzzy PSO algorithm.

4.4 Hybrid Approaches

Although, meta-heuristics are in nature hybrid, higher level approaches combing stand alone heuristics are also being reported in the literature for the problem. Recently, Xhafa et al. [18] proposed a GA(TS) algorithm for the scheduling of independent tasks under ETC model.

5 Conclusions

In this work, we have reviewed the issues related to the scheduling in Grid systems. Starting with the main factors that determines its complexity, scheduling problem is conceived as a family of problems, by identifying most of common versions based on six dimensions: type of the environment, architecture type of the scheduler, immediacy of the processing, type of interrelations among tasks, type of preemptive policy and type of optimization model. Then, we have briefly reviewed different families of heuristic methods used for the resolution of the problem, including ad hoc methods, local search methods and population-based methods.

References

1. Abraham, A., Buyya, R., Nath, B.: Nature's heuristics for scheduling jobs on computational grids. In: The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), India (2000)

2. Abraham, A., Liu, H., Zhang, W., Chang, T.: Scheduling jobs on computational grids using fuzzy particle swarm algorithm. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS (LNAI), vol. 4252, pp. 500–507. Springer, Heidelberg (2006)
3. Braun, T.D., Siegel, H.J., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. of Parallel and Distributed Comp. 61(6), 810–837 (2001)
4. Gao, Y., Rong, H., Huang, J.Z.: Adaptive Grid job scheduling with genetic algorithms. Future Gener. Comput. Syst. 21(1), 151–161 (2005)
5. Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., Freund, R.F.: Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of Parallel and Distributed Computing 59(2), 107–131 (1999)
6. Page, J., Naughton, J.: Framework for task scheduling in heterogeneous distributed computing using genetic algorithms. AI Review 24, 415–429 (2005)
7. Ritchie, G.: Static multi-processor scheduling with ant colony optimisation & local search. Master's thesis, School of Informatics, Univ. of Edinburgh (2003)
8. Ritchie, G., Levine, J.: A fast, effective local search for scheduling independent jobs in heterogeneous computing environments. Technical report, Centre for Intelligent Systems and their Applications, University of Edinburgh (2003)
9. Wu, M.Y., Shu, W.: A high-performance mapping algorithm for heterogeneous computing systems. In: Proceedings of the 15th International Parallel & Distributed Processing Symposium, p. 74 (2001)
10. Xhafa, F., Abraham, A.: Meta-heuristics for Grid Scheduling Problems. In: Metaheuristics for Scheduling in Distributed Computing Environments, ch. 1. Series Studies in Computational Intelligence, pp. 1–37. Springer, Heidelberg (2009)
11. Xhafa, F.: A Hybrid Evolutionary Heuristic for Job Scheduling in Computational Grids, ch. 10. Series: Studies in Computational Intelligence, vol. 75. Springer, Heidelberg (2007)
12. Xhafa, F., Alba, E., Dorronsoro, B., Duran, B.: Efficient Batch Job Scheduling in Grids using Cellular Memetic Algorithms. Journal of Mathematical Modelling and Algorithms (2008) (accepted)
13. Xhafa, F., Barolli, L., Durresi, A.: Batch Mode Schedulers for Grid Systems. International Journal of Web and Grid Services 3(1), 19–37 (2007)
14. Xhafa, F., Carretero, J., Alba, E., Dorronsoro, B.: Tabu Search Algorithm for Scheduling Independent Jobs in Computational Grids. Computer And Informatics Journal 28(2), 237–249 (2009)
15. Xhafa, F., Carretero, J., Abraham, A.: Genetic Algorithm Based Schedulers for Grid Computing Systems. International Journal of Innovative Computing, Information and Control 3(5), 1–19 (2007)
16. Xhafa, F., Carretero, J., Barolli, L., Durresi, A.: Immediate Mode Scheduling in Grid Systems. Int. J. of Web and Grid Services 3(2), 219–236 (2007)
17. Xhafa, F., Duran, B., Abraham, A., Dahal, K.P.: Tuning Struggle Strategy in Genetic Algorithms for Scheduling in Computational Grids. Neural Network World 18(3), 209–225 (2008)
18. Xhafa, F., Gonzalez, J.A., Dahal, K.P., Abraham, A.: A GA(TS) Hybrid Algorithm for Scheduling in Computational Grids. In: Corchado, E., Wu, X., Oja, E., Herrero, Á., Baruque, B. (eds.) HAIS 2009. LNCS (LNAI), vol. 5572, pp. 285–292. Springer, Heidelberg (2009)
19. YarKhan, A., Dongarra, J.: Experiments with scheduling using simulated annealing in a grid environment. In: Parashar, M. (ed.) GRID 2002. LNCS, vol. 2536, pp. 232–242. Springer, Heidelberg (2002)
20. Zomaya, A.Y., Teh, Y.H.: Observations on using genetic algorithms for dynamic load-balancing. IEEE Transactions on Parallel and Distributed Systems 12(9), 899–911 (2001)