ISSN No. 0976-5697

Volume 3, No. 1, Jan-Feb 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Taxonomy of Scheduling in Grid Environment

Arun Baruah
133/4, 4th Crs, Munekolala
Marathalli, Bangalore India
arunbaruah@gmail.com

***Abstract*:** Grid computing is a high performance computing environment consisting of thousands of computers, sensors and other resources which are geographically distributed or globally located across multiple administrative domains and used to solve large scale computational demands. In the grid environment, users can access the resources transparently without knowing where they are physically located. To achieve the promising potentials of computational girds, job scheduling is an important issue to be considered. Scheduling is very complicated due to the unique characteristics of the grids. Scheduling using general techniques in grids could not yield better optimization result. Meta heuristic techniques are more result oriented in scheduling of computational grids. This paper focuses on the taxonomy of Grid scheduling.

Keywords: Scheduling, grids, Meta heuristic

I. INTRODUCTION

Grids [1] have emerged as a global cyber-infrastructure for the next generation of e-science applications by integrating large-scale, distributed and heterogeneous resources. Scientific communities, such as high-energy physics, gravitational-wave physics, geophysics, astronomy and bioinformatics, are utilizing Grids to share, manage and process large data sets. To achieve the potentials of computational grids, job scheduling is an important issue to be considered. The first paper on the critical path was published by Kelley Walker in March 1959 who defined the science of scheduling using critical path analysis. The evolution of scheduling was then applied in the development of computers, earlier in mainframes and latter 1980's the personal computers and now to the clusters and the grids.

The grid applications which contain many tasks that may take several days or weeks to complete whose completion time is affected by task scheduling. The delay in single tasks can affect the completion time of the entire application [2]. In order to support complex scientific experiments, distributed resources such as computational devices, data, applications needs to be orchestrated while managing the application workflow operations within Grid environments.

The main goal of scheduling is to minimize the job completion time and wastage of CPU cycles [3] but scheduling jobs in heterogeneous grid environment is different compared to parallel architectures. Scheduling is highly complicated by the distributed ownership of the grid resources as consumers of providers of the grid resources have their own access policy, scheduling strategy and optimization objectives [4]. Grid Schedulers should also support advanced features such as (i) user requested job priority (ii) advanced reservation of resources (iii) resource usage limits enforced by administrators (iv) user specifiable resource requirements. Several architectures are available to reduce the complexity of the problem for particular application scenarios. Generic features of enterprise grids, high performance computing grids and global grids have been identified to develop a scheduling instance for the scheduling solutions [5].

In the grid system, an end user submits the job that has to be executed with some constraints like job execution deadline, cost for the execution and the time required for the execution. Grid resource manager estimates the resource requirements and provides the functionality for discovery and publishing of resources as well as scheduling, submission and monitoring of jobs [6]. Thus difference performance goals of grid scheduling includes: maximizing system throughput [7], maximizing resource utilization, minimizing execution time [8], minimizing cost on the user side and fulfilling economic constraints [9]. Thus this paper focus on scheduling algorithms in grid environment based on traditional as well as meta-heuristic techniques. The remaining section of the paper is organized as follows: Section II deals with the scheduling problems in grid computing. Section III deals with types of grid scheduling. Section IV deals with grid system performance and scheduling optimization criteria. Section V deals with resource utilization. Section VI deals with heuristics and meta-heuristic methods for scheduling in grids and Section 7 concludes this paper.

II. SCHEDULING PROBLEMS IN GRID COMPUTING

Grid computing scheduling is NP hard because there are many intervening parameters of scheduling. There are several characteristics that make grid scheduling problems different from distributed systems. The following are some of the characteristics:

a. The dynamic structure of the Computational Grid: In grid systems, computing resources for example the processors, storage device etc., can take part in the grid system or abort the grid at any time in a volatile manner. This may be because of wilful nature of the participants or volunteers or it can be due to loss of connection because of internet failure.
b. Heterogeneity of resources: Grid systems are very highly distributed and the computational resources are very disparate, ranging from desktops, high performance clusters, or Beowulf clusters, laptop or even handheld computational resources. Current Grid infrastructures are not yet much flexible but

heterogeneity is among the most important characteristics of existing grid system.

c. High heterogeneity of jobs: The tasks or the jobs to existing grid systems are too diverse and heterogeneous with respect to their computational needs. They can be computing power intensive or data intensive. Jobs could be very large applications or it can be an atomic tasks. Grid systems cannot be attentive of the tasks or jobs arriving in the grid.
d. Existence of local schedulers or resources: As the Grid systems are developed with the help of many organizations, research institutions and Universities; there is a possibility of running local schedulers such as Condor batch system.

III. TYPE OF GRID SCHEDULING

Different types of scheduling approach can be followed because of the nature of Grid systems. We can have independent scheduling, workflows, static and dynamic, centralized, hierarchical and de-centralize, immediate and batch mode, adaptive scheduling.

a. Independent Scheduling: Independent scheduling is required in some specific applications and data intensive computing where CPU is use intensively by the application.

b. Grid workflows: In complex problem solving mechanism in grid computing has several services. So the dependency arises. There need a mechanism of solving problems is called a grid workflows.

c. Centralized, hierarchical and decentralized scheduling: More control of resources are possible in centralized scheduling. The scheduler monitors the systems resources and has knowledge of the systems, but it suffers limited scalability because of the centralized scheduling mechanism in the highly distributed nature of grids. Whereas in decentralized scheduling, all the resources i.e., computers attached to the larger grids has no central schedulers. Here the local scheduler plays an important role.

d. Static vs. dynamic scheduling: Dynamic Grid scheduling is determined by two types i.e., the dynamic of job execution and the other is the dynamics of resources. Dynamic execution of jobs means the execution of jobs is currently in the preemptive mode or the execution of jobs has stopped because of the interrupts from other job request of high priority jobs. Under dynamic scheduling, the resources can join or leaves the grid at any moment. It can also be in an un-predictable way. Under the static method, job is assigned once to a resource. In Static method resource information and performance parameters are assumed to be known. Depending on how a job can be divided, relevant research can be categorized into two different areas: divisible workload scheduling [10 - 12], where they can divide workload into arbitrary-sized pieces, and fixed-sized independent tasks scheduling [13].

e. Immediate vs. batch mode scheduling heuristics: Immediate and batch scheduling heuristics are useful scheduling heuristic algorithm for Grids. Immediate scheduling heuristics works immediately when any jobs / tasks arrive, the immediate scheduling scheduler schedules the jobs and executed with available resources. Immediate scheduling heuristics includes the minimum execution time scheduling algorithm, the Opportunistic load balancing algorithm, the minimum completion time algorithm, Min-min, Max-min algorithm, Relative cost, Shortest job fast. Whereas in the batch mode scheduling, grouping of jobs together is done in batches and scheduled as a batch. In batch mode system, it is advantageous in allocating and distributing jobs based on available resources. It is also one of a simple and powerful heuristics that are distinguished for efficiency.

f. Adaptive Scheduling: As Grid computing environment is evolving and it is becoming bigger and bigger and is composed of many different types of Grid resources, so now a more sophisticated scheduling techniques are required. Dynamic scheduling heuristics [14] is one of the most vital scheduling heuristics that is concern for both current status of the resources and predictions for the future status with the motive of avoiding performance degradation. Adaptive scheduling weakening can be seen in Rescheduling heuristics in which running jobs migrates to other available suitable resources. The authors used NetSolve [15] as a test bed for evaluating the proposed algorithm.

g. Adaptive Partitioning: Adaptive partitioning gives each job a dedicated partition of the machine, where the partition size is a compromise between the scheduler and the job that takes both the job's requirements and the current load into account. Users can specify a set of possible partition sizes when a job is submitted, rather than demanding a single size. The scheduler allocates the largest size that is available. This has the desired outcome: The system automatically allocates the largest size that is available that is suitable for the job thus allowing it to start running as soon as possible.

IV. GRID SYSTEM PERFORMANCE & SCHEDULING OPTIMIZATION CRITERIA

In Grid scheduling processes, several optimization criteria and performance requirements can be considered which is multi-objective in its formulation. Optimization objectives are to establish the overall Grid performance criteria of the grid. The performance criteria includes the utilization of resources like CPU, waiting time and response time, throughput, turnaround time, load balancing. Scheduling optimization criteria include: the makespan, the flowtime, resource utilization, load balancing, matching proximity, turnaround time, total weighted completion time, lateness, weighted number of tardy jobs, weighted response time, etc. Both performance criteria and optimization criteria are desirable for any Grid system; however, their achievement depends also on the considered model (batch system, interactive systems etc). It is important to note that these criteria are conflicting among them; for instance, minimizing makespan conflicts with resource usage and response time. An extensively studied optimization criteria is the minimization of the makespan.

The productivity of the grid system is indicated by the makespan where small values of makespan means scheduling is providing robust and efficient utilizing of tasks to the available resources. Another optimization

criteria is the flowtime, which states to the response time of the user submitted tasks executions. The average response time of the grid system is reduced by minimizing the value of the flowtime. We want to maximize the throughput of the grid and at the same time also maintaining the Quality of Service.

Makespan, completion time and flowtime: In grid scheduling, it is aimed to minimize the makespan and flowtime. Makespan is the time when finishes the latest task and the flowtime is the sum of the finalization times of all the tasks. Formally they can be defined as follows:

Minimization of makespan: minSi€ Sched {maxj€Jobs Fj } and,

Minimization of flowtime: minSi € Sched{ ∑ i €Jobs Fj }

Whereas Fj means the time when the task j finalizes, Sched is the set of all schedules and Jobs the set of all jobs to be scheduled. Please note that makespan is not affected by any particular order of execution of tasks. Tasks should be executed in a ascending order of their workload (computation time) in order to minimize flowtime of a resource. m is the completion time in which m finalize the previous assigned jobs and also planned tasks for the machine.

The expression of makespan, flowtime and completion time depends on the computational model. For instance, in the ETC model, completion[m] is calculated as follows:

completion[m] = ready_times[m] + ∑ ETC[j][m]
{j€Tasks | schedule[j] = m}

where ready_times[m] is the time when machine m will have finished the previously assigned tasks. Makespan can be expressed in terms of the completion time of a resource, as follows:

Makespan = max{completion[i] | I € Machines}

Flow_time utlilze completion times of machines in the same way, but now by first sorting in ascending order as per their ETC values assigned to the machine. Thus for machine mCTR the flow_time flowtime[mCTR] can be expressed as follows (S[mCTR] is a vector representing the schedule for machine mCTR):

V. RESOURCE UTILIZATION

Another important objective of the grid system is the maximization of resource utilization. Due to the contribution of Grid resources by individuals or institutions in exchange of economic benefits, this criterion is gaining high importance.

Accomplishing a greater resource reutilization becomes a challenge in Grid based systems gives the differences of computational resources of the Grid. One of the possible definition of this parameter is to consider the average utilization of resources. For instance, in the ETC model, for a schedule S, it can be defined as follows:

$$\text{avg\_utilization} = \frac{\sum\{i \in \text{Machines}\} \text{completion}[i]}{\text{makespan} * \text{nb\_machines}}$$

Matching Proximity: matching proximity could be defined as the degree of proximity of a given schedule with regard to the schedule produced by MET (minimum execution time) method in ETC model. MET assigns the jobs to the resources having smallest execution time for that job. For a schedule S, matching proximity can be computed as follows:

$$\text{Matching-proximity} = \frac{\sum i \in \text{Tasks ETC}[i][s[i]]}{\sum i \in \text{Tasks ETC}[i][\text{M ET}[i]]}$$

Turnaround time: The elapsed time of computation, from the submission of the first tasks to the completion of the last task. Dominguez et al., considered this objective for scheduling bags-of-tasks applications in the desktop Grids. Kondo [16] characterized four real desktop grid systems and designed scheduling heuristics based on the resource prioritization, resource exclusion, and task replication for fast application turnaround time.

VI. HEURISTICS AND META-HEURISTIC METHODS FOR SCHEDULING IN GRIDS

From the above explanation, we are clear that the Grid scheduling is highly challenging, dealing with many optimization criteria. Meta-heuristics are now considered the de facto approach to grid scheduling. The meaning of Heuristic is “to find” and the word “Meta” means a broad way, so meta-heuristic is to find in broad way. Many libraries and frameworks are also developed and mentioned in the literature. For example, Easy Local++, is one of the available libraries which can be easily use for Grid scheduling problems.

Local Search Based Heuristic Approaches - families are Hill Climbing, Simulated Annealing (SA), Tabu Search method. The Hill Climbing method is important for two reasons: (1) in a very short time they can produce a feasible solution of some quality and, (2) they can be initialized in population-based meta-heuristics. Ritchie and Levine [17] use several local search method in implementing Memetic Algorithms for the same problem. Abraham et al. [16] have proposed SA (simulated annealing) as more powerful algorithm then local search by accepting worse solutions with certain probability. Tabu Search is more sophisticated and usually requires more computation time for computing good solutions.

Opulation- Based Heuristic Approaches: large combinatorial optimization problems can be effieicntly solved by using population-based heuristic approach. Population-based methods are: Genetic Algorithm (GA), Memetic Algorithms (MA) Ant Colony Optimization (ANZ) and Particle Swarm Optimization (ParticleZ).

Memetic Algorithms (MA) is a relatively new class of population-based methods, which combine the concepts of evolutionary search and local search by taking the advantage of good characteristics of both of them.

An ant colony optimization for the problem is proposed by Ritchie under ETC model. Abraham et al. [18] proposed an approach for scheduling jobs on Computational Grids using fuzzy Particle Swarm Optimization algorithm.

Hybrid Heuristics Approach: Meta-heuristic approaches are hybrid approach. Ritchie and Levine combine ant colony optimization (ANZ) with the Tabu Search (TS) algorithm. In a basic unstructured Memetic Algorithms is combined with 16 local search algorithms in order to identify the base performance of the resultant Memetic Algorithm. Simulated annealing and Tabu search heuristics; the hybridization Genetic Algorithm + Simulated annealing is expected to have a better convergence than pure Genetic Algorithm search and the GA + TS could improve the efficiency of the GA.

Genetic Algorithm heuristics: Genetic algorithm heuristics have been developed by John Holland, his colleagues, and his students at the University of Michigan. The goals of their research have been twofold: (a) to

abstract and rigorously explain the adaptive processes of natural systems, and (b) to design artificial systems software that retains the important mechanisms of natural systems. Genetic Algorithm heuristics are an evolutionary techniques that is used to search for optimal solution in a very large space. Genetic algorithm heuristics are inspired from human genetics and generally work by encoding the problem in the form of chromosomes. Genetic algorithm operators like crossover and mutations are applied and new generations are evolved. Fitness is computed after every generations and further exploration is stopped as soon as an acceptable fitness value is achieved. Genetic algorithm heuristics are now best known heuristics for scheduling.

Simulated Annealing heuristic techniques: Simulated annealing heuristic technique is an iterative (looping) technique that considers only one possible mapping for each tasks at a time. This heuristic solution also uses the same representation as the chromosome for the Genetic Algorithm heuristics. Simulated annealing heuristic technique uses a procedure that probabilistically allows poorer solutions to be accepted to attempt to obtain a better search for the solution space.

VII. CONCLUSION

This paper has given a detailed study on scheduling and on taxonomy of scheduling has been given in different perspectives. It is observed that the heuristic based algorithms, in particular population based heuristics are most suitable for scheduling the task in the grid environment. But there are population based heuristics which are complex in nature and takes a long execution time. For instance, ant colony optimization (ANZ), when run in a normal PC, it takes hours to execute an algorithm to schedule more than 1000 processes. On the other hand this algorithm gives better results compared to other population based heuristics such as genetic algorithm, due to its longer execution time of the algorithm, some other algorithm which executes faster are considered.

VIII. REFERENCES

[1]. I. Foster and C. Kesselman. The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann Publishers, USA, 1999.

[2]. Dr. K. Vivekanandan et al, D. Ramyachitra "A study on Scheduling in Grid Environment" International Journal On Computer Science and Engineering (IJCSE) Vol.3 No.2 Feb 2011, pp. 940 – 950.

[3]. Harumasa Tada, Makoto Imase, Masayuki Murata, "On the Robustness of the Soft State for Task Scheduling in Large-scale Distributes Computing Rnvironment," Proceedings of the International Multiconference on Computer Science and Information Technology, pp. 475 – 480, 2008.

[4]. Yanmin Zhu, Lijuan Xiao, Lionel M. Ni, Zhiwei Xu, "Incentve Based P2P Scheduling in Grid Computing, " Grid and Cooperative Computing, LNCS 3251, Springer Verlag, pp. 209 – 216, 2004

[5]. N. Tonellotto, R. Yahyapour, Ph. Wieder, "A Proposal for a Generic Grid Scheduling Architecture", CoreGRID Technical Report, Number TR-0015, Jan. 11, 2006.

[6]. Ajith Abraham, Hongbo Liu, Weishi Zhang, and Tae-Gyu Chang, "Scheduling Jobs on Computational Grids Using Particle Swarm Algorithm", 10th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, UK, 2006, pp.500-507

[7]. Michael Litzkow, Miron Livny, and Matt Mutka, "Condor-A Hunter of Idle Workstations", In Proceedings of Eight International Conference of Distributed Computing Systems, California, June, 1988, pp. 204 – 211

[8]. David Abramson, Rock Sosic, J. Giddy, B. Hall, "Nimrod: A tool for performing parameterised simulation using distributed workstations", In HPDC, pp 121 – 121, 1995

[9]. Rajkumar Buyya, David Abramson, Jonathan Giddy, "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid", The 4th International Conferrence on High Performance Computing in Asia-Pacific Region (HPC Asia 2000), May 2000, Beijing, China, IEEE Computer Society Press, USA

[10]. O. Beaumont, A. Legrand, Y. Robert, "Seheduling divisible workloads on heterogeneous platforms", Parallel Computing, 29(9), September 2003, pp 1121-1152

[11]. T. Robertazzi, "Ten Reasons to Use Divisible Load Theory", Computer, 36(5), May 2003.

[12]. V. Bharadwaj, D. Ghose, V. Mani, T. Robertazzi, "Scheduling Divisible Loads in Parallel and Distributed Systems", IEEE Computer Society Press, Los Alamitos, CA, Sept. 1996

[13]. T. D. Braun, H. J. Siegel, N. Beck, "A Comparision of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing, 2001

[14]. I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputer Applications, 2001

[15]. I. Foster and C. Kesselman (editors), "The Grid: Blueprint for a Future Computing Infrastructure", Morgan Kaufmann Publishers, USA, 1999.

[16]. Kondo, D., Chien, A., Casanova, H, " Scheduling Task Parallel Applications for Rapid Turnaround on Enterprise Desktop Grids", Journal of Grid Computing 5(4), 379-405 (2007).

[17]. Casanova, H., Kim, M., Plank, J.S., Dongarra, J.J, " Adaptive Scheduling for Task Farming with Grid Middleware", Int. J. High Perform. Comput. Apple. 13(3), 231-240 (1999)

[18]. A. Abraham, H. Liu, W. Zhang, T.G. Chang, Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm, Springer-Verlag Berlin Heidelberg (2006) 500-507.