Volume: 8 Issue: VI Month of publication: June 2020
DOI: http://doi.org/10.22214/ijraset.2020.6118

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VI June 2020- Available at www.ijraset.com

A Survey on Grid Computing Scheduling Algorithms

Lav Ram Gabri[1], Yash Agrawal[2], BK Srinivas[3]
[1,2]*Student, Information Science and Engineering, R.V. College of Engineering, Bengaluru*
[3]*Assistant Professor, Information Science and Engineering, R.V. College of Engineering, Bengaluru*

Abstract: Grid computing is a group of physically connected computers that performs dedicated tasks. And to use grid computing efficiently it is necessary to schedule the jobs on correct resources. To serve this purpose several scheduling algorithms have been developed specifically for grid computing environment. Some of these algorithms are traditional which can be applied to other job scheduling environment too such as in operating system etc. But many algorithms are designed particularly for grid environment and thus are too complex to be used in other cases. This survey paper presents few such algorithms that have been developed long ago but are in constant research and thus proved to be more efficient than traditional ones. The paper presents the theory behind these algorithms and various assumptions and steps involved.
Keywords: Grid computing, Ant Colony Algorithm, Compact Genetic Algorithm, Load Balanced Min-Min Algorithm, Divisible Load Scheduling

I. INTRODUCTION

Grid Computing can be described as a network of computers working together to perform a task that would be difficult for a single machine. All computers on the network run under the same protocol to serve as a virtual supercomputer. The job they focus on can include the study of vast databases or the simulation of scenarios involving high computational power. These networks of computers add resources such as power generation and storage space to the network.
Scheduling in grid computing is always an active area of research. Though scheduling aspect is also found in operating system, but it is important to understand that both the scheduling of OS and of Grid computing are totally different. It is also important to understand that the algorithms that are used in OS scheduling could not be used as scheduling algorithms in Grid. The main objective of this paper is to present an extensive study of different algorithms related to scheduling in grid computing.
This paper is structured as follows. Section II discuss the ant-colony algorithm that is mainly derived from [1]. Section II discuss the compact genetic algorithm from [2], Section III discuss the load balanced min-min algorithm from [3], Section IV presents the divisible load scheduling from [4].

II. ANT COLONY ALGORITHM

It has been explained in [1] that grid computing is better than cluster computing because of the flexibility the former provides. Various resources from different geographical areas come together to develop a grid environment. Resources are not centrally controlled in the grid computing system and can enter and leave the grid environment at any time.
To find the best resources (high CPU speed, large memory) to process a particular job in terms of minimizing work time is the main challenge for any grid computing scheduler. To solve this and the problem of Stagnation effective scheduling algorithms must be used.

A. Related Works

Ant Colony Optimization (ACO) algorithm is one of the efficient algorithms for Grid Computing. In this algorithm ants work together to find the shortest route between their nest and the food source. Each ant deposits pheromone – a chemical substance in which higher value of pheromone means shorter routes, and vice versa. The tasks among ants are divided as the Soldiers protect, Scouts search for food and Queen reproduces. The ACO algorithm is used in grid computing because it is easily adapted to solve both static and dynamic combination optimization problems.
Balanced job assignment based on ant colony algorithm for computing grids called BACO was proposed by [5]. It minimizes the computation time of job execution, which focuses on the load balancing factor. It chooses the optimal resources to process submitted jobs by applying local and global pheromone techniques to balance system loads. The local pheromone update function updates the status of the selected resource after the job has been assigned, and the job scheduler depends on the latest information from the selected resource for the next job submission. Global pheromone update function updates the status of each resource for all jobs after jobs have been completed. Job Scheduler uses resource availability statistics using the above status.

An ACO proposed by [6] aims to minimize total tardiness time. The initial pheromone value of each resource is dependent on the estimated time of execution and the actual time of execution of each task. The process to update the pheromone value for each resource is based on local update and global update rules, as in Ant Colony System.
The study by [7] proposed a bio-inspired adaptive job scheduling which has good adaptability and robustness in a dynamic computational grid. It also minimizes the execution time by using large distributed resources.
In [8] an optimized version of ant colony algorithm is presented in which Pheromone update function is performed by adding encouragement, punishment coefficient and load balancing factor. The job having the max pheromone level is chosen. The pheromone strength of each resource will be updated after completion of the job. If a job is completed for a particular resource, more encouragement will be added which will raise pheromone concentration increasing priority. Else, punishment by adding less pheromone value. It leads to better average utilization; response time and task fulfil proportion.

B. *Proposed Algorithm*

The proposed algorithm minimizes the computational time of each job that must be processed by available resources in grid computing system using pheromone value selection.
The algorithm works as follows:

1) The user will send a request to process a job. Job details like the size of each job, total jobs, CPU time per job, etc. will be sent with the request.
2) The calculation of relevant parameters is started by the grid resource broker post receiving the user's message. The grid resource broker fetches some of its resources from the information server.
3) The proposed technique selects the largest entry in the pheromone value (PV) matrix as a resource to process the submitted job. After a job is assigned to a resource, a local pheromone update is performed.
4) After a resource has completed processing a job, a global pheromone update is performed.
5) The execution results will be sent to the user.

Pheromone value will be determined by two types of pheromone update technique that are local pheromone updates in ACS [9] and global pheromone updates in MMAS [10].
This algorithm is based on local pheromone update and trail value. In this the initial value of pheromone is based on transmission time, characteristics of jobs and resources.
To ensure the resource is less desirable for other ants, the local pheromone trail update will reduce the amount of pheromone in assigned resources.
The trail limit (allowed range of the pheromone strength), in this case, is limited to a maximum and minimum trail strength. TIt helps to control the value of pheromone updated on each resource and ensures that already assigned optimal resources will not be chosen for fresh jobs. The proposed algorithm is simple to be implemented due to the existing of information of each resources and jobs. This algorithm was able to minimize the completion time of each job.

III. COMPACT GENETIC ALGORITHM

A heuristic approach based on compact genetic algorithm is presented to solve the problem of grid scheduling tasks. The goal of this method is to create an optimum schedule such that the minimum time of completion can be reached while the tasks are completed.
It is presented in [2] that Job sharing (computational burden) is one of the major difficult tasks in a computational grid environment. Also, classical algorithms for scheduling could not be used as they are not dynamic and solving task scheduling in grid is one of the NP-Complete problem which requires dynamic scheduling for efficient working.

A. *Problem Identification*

The purpose of task scheduling is to minimize the time of completion and to allow effective use of resources [11][12]. The scheduling problem occurs in a case where there are more tasks than the resources available. We need to build new algorithms for task scheduling, since the inefficient allocation of resources will significantly hinder the performance and throughput of the scheduler.
To solve task scheduling problem, Compact Genetic Algorithm (CGA) is used:

1) Set an initial population by selecting random starting sequence from set x! where x is number of tasks.
2) After getting initial solution, calculate fitness value of each solution as per equation.
3) Calculate best among entire solution and set as initial global best.

4) GA update equation is used to update old population and generate new sequences and then their resources are calculated.
5) These sequences, along with their resources are then used to find the fitness value of each individual of each solution of the Population [13].
6) After this if crossover criterion is satisfied, then crossover operation performed over two randomly selected parents and as a result a new sequence is generated. The resource of this offspring is calculated after this.
7) Using the sequence and its resources the fitness value of the offspring is calculated.
8) If the offspring is better than its worst parent, then this solution replaces that parent [14].

B. Genetic Algorithm And Scheduling

GA is a meta-heuristic search technique that enables large solution spaces to be partially searched in polynomial time, using evolutionary techniques of nature. It uses historical info to exploit best solutions from previous searches. There are 3 steps in GA – selection, crossover and random mutations. The fitness function of each individual in the population is evaluated to produce a value that indicates the goodness of the solution. Selection takes a certain number of people into the population and carries them to the next generation. Crossover brings in pairs of individuals and uses parts of each to create new individuals. To prevent the GA from getting caught in a local minimum, random mutations swap parts of an individual.

C. Proposed Methodology

[2] presented a compact genetic algorithm with linear Crossover operator. In this CGA with Linear Crossover is based on population concept and each individual in population represents a solution, (solution is sequence of tasks). Also, Fitness function calculate fitness value of each individual.

D. Proposed Algorithm

Compact Genetic Algorithm (CGA) represents the population as a distribution of probability over a set of solutions. It sample individuals according to the probabilities defined in the probability vector. Individuals are evaluated and the probability vector is modified for a better individual. The CGA cycle is repeated until the probability vector has converged.

CGA with linear crossover initializes the Probability Vector (PV), where each part of the PV initializes with a parameter of 0.5, and then two solutions are randomly generated using this PV. The solutions created are graded on the basis of their fitness values. The PV will then be updated on the basis of these solutions. This adaptation cycle continues until the PV has converged.

Thus, CGA is an algorithm that mimics the order of action of a simple genetic algorithm with a given population size and selection rate but reduces its memory requirement.

IV. LOAD BALANCED MIN-MIN ALGORITHM

[3] proposed a load balanced min-min algorithm that produces schedule to minimize makespan and to increase resource utilization. The three main phases in grid scheduling according to [15] are resource discovery, gathering resource information and job execution. There are several algorithms in place which minimizes the overall completion time of the task as a whole but that does not mean that they minimize the completion time of individual task.

The min-min and max-min algorithms [16], [17], [18], [19], [20], [21] considers the execution and completion time of each task in each available grid resource.

The Min-Min algorithm first finds the minimum execution time of all tasks and then chooses the task with the least execution time among all the tasks. It then proceeds by assigning the task to the resource that produces the minimum completion time and then the same procedure is repeated by Min-Min until all tasks are scheduled.

The drawback of the Min-Min algorithm is that it first selects smaller tasks that allow use of a resource with high computational capacity. As a result, the schedule generated by Min-Min is not optimal when the number of smaller tasks exceeds the larger ones.

To solve this difficulty [22], the Max-Min algorithm first schedules larger tasks. For certain instances, however, the makespan can first increase due to the execution of larger tasks. The processing period for smaller tasks is also decreased in Max-min.

The proposed two-phase algorithm applies the min-min algorithm in the first phase and then reschedule it by considering the maximum execution time that is still less than the makespan obtained from the first phase. In this way not only the completion time of the whole task reduces but also the time required to complete an individual task also reduces.

A. Assumptions

The proposed algorithm by [3] has certain assumptions which are:

1) The applications to be performed consist of a set of indivisible tasks that do not depend on each other, called a meta-task.
2) Tasks have no deadlines or goals correlated with them.
3) Estimates of planned task execution times for each system in the HC suite are known. Such estimates can be given before the assignment is submitted for execution or at the time it is submitted.
4) The mapping method is to be done statically in batch mode.
5) The mapper runs on a separate computer and manages the execution of all tasks on all computers in the system.
6) Each computer executes a single task at a time in the order in which the tasks are assigned (First Come First Served-FCFS).
7) The scale of the meta-tasks and the number of machines in the heterogeneous computing system are known.

In static heuristics, the correct calculation of the estimated time of execution for each operation on each system is known a priori and is found within the ETC (estimated time to compute) matrix. The main aim of this algorithm is to minimize the makespan.

B. Proposed Algorithm

The steps involved in the algorithm are as follows:

1) LBMM run Min-Min in the first half. In the second stage, the resources with heavy load are selected and reassigned to light load resources.
2) Identifies heavy load resources (high makespan resources).
3) The maximum completion time for this task is then compared to the makespan provided by Min-Min. If it is less than makespan, the job will be rescheduled in the resource that generates it, and the time of both resources will be changed.
4) Otherwise, the next estimated completion time for this task is chosen and the steps are repeated again. The cycle stops if all the resources and tasks allocated to it have been considered for rescheduling.
5) In this way idle or min load resources are rescheduled.

As earlier described Min-Min and Max-Min algorithms are applicable to small scale distributed systems. When the number of small tasks is greater than the number of large tasks in a meta-task, the Min-Min algorithm cannot properly schedule tasks, and the makespan of the program is fairly high. To overcome this difficulty the load balanced Min-Min algorithm works in two phases and try to make use of all the available resources. In this way not only the makespan time reduces but also each of the resource gets fair share of tasks.

V. DIVISIBLE LOAD SCHEDULING

Over the past 15 years or so, a modern mathematical framework has been built to allow for a workable performance analysis of systems integrating communication and computing problems, such as parallel and distributed processing. A main aspect of this divisible load distribution scheduling theory (known as DLT) is the use of a linear mathematical model. Since the initial work on this topic in 1988 by [23], an expanded body of work on this topic has been published worldwide.

In the divisible load distribution principle, it is assumed that the computation and communication loads can be divided arbitrarily between a number of processors and ties, respectively. A continuous mathematical formulation shall be used. In addition, there is no clear relationship of precedence between the results. Load may also be arbitrarily allocated to the network links and processors. This class of problems is well suited for modelling a wide class of parallel computational data problems. As a secondary advantage, it sheds light on architectural problems relating to parallel and distributed computing. In addition, the principle of divisible load scheduling is practically deterministic. Although stochastic features may be implemented, there are no statistical assumptions in the basic model that may be the Achilles heel of the performance assessment model.

Usually, divisible load models can be resolved algebraically for optimum load allocation and timing to processors and linkages. Here, the optimality is specified in the sense of the particular interconnection topology and scheduling policy used. Solution is typically obtained by forcing all processors to finish processing at the same time (otherwise, intuitively, loads could be moved from busy to idle processors, see [24] for formal proof).

In the case of a single level tree (star) network, this constraint leads to a set of clustered equations that can be recursively resolved for optimal load allocation to each processor in the sense of a given load distribution schedule.

A valuable concept in the theory of divisible load is that of an equivalent processor [25]. One may replace a sub-network with a single equivalent processor that has exactly the same total processing power (i.e. speed) as the original sub-network. In addition, for homogeneous networks, it is possible to solve the output of infinite size networks by setting up an implicit equation.

The actual load scheduling policy used is also relevant. Linear divisible load theory can be used to model a wide range of approaches. For example, a load can be distributed either sequentially or simultaneously. Under sequential (bus-like) load distribution, the method used in most of the literature to date, the node can transmit load to one of its children at a time. This results in saturation of the speed as the size of the network increases. Quality may be enhanced by distributing load from the node to children in periodic instalments, but output is still saturated as the number of instalments increases [26].
[4] proposed balanced load sharing by the use of divisible load distribution scheduling theory and algorithms that focus on the characteristics of physical processes and make the most of computing and data resource scheduling efforts to avoid "hot spots" in the grid. They proposed this algorithm to be used by large physics experiments sites such as Solenoidal Tracker at RHIC (STAR), to improve the performance and make efficient use of all the available grids which proved to be helpful when done in simulation.

VI. CONCLUSIONS

Grid computing tough replaced by cloud in many organisations is still being widely used for scientific and medical purposes. Since the organisations that still make use of it are very large and where grid network consists of hundreds or thousands of physical systems using old simple algorithms for scheduling of jobs might not help. This paper thus presents some complex algorithms proposed by different works at a single place. Due to complexity of the mentioned algorithms, mathematics and other related formulas have not been presented and readers are advised to view the original papers for more details.

VII. ACKNOWLEDGMENT

A lot of analysis and reading was done prior to the curation of the content of the given paper. This would definitely not be possible under the guidance of our mentor, Prof. B.K. Srinivas, who constantly guided us and gave us a direction in the due course of the given paper. We would also like to thank our Head of Department, Dr. B.M. Sagar, who gave us this opportunity to work on this paper.

REFERENCES

[1] Ku Ruhana Ku-Mahamud, Husna Jamal Abdul Nasir, "Ant Colony Algorithm for Job Scheduling in Grid Computing," 2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, May 2010.

[2] Prateek kumar Singh, Neelu Sahu, "Task Scheduling in Grid Computing Environment Using Compact Genetic Algorithm," International Journal of Science, Engineering and Technology Research (IJSETR), Volume 3, Issue 1, January 2014.

[3] T. Kokilavani, Dr. D.I. George Amalarethinam, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing," International Journal of Computer Applications (0975 – 8887) Volume 20– No.2, April 2011.

[4] Dantong Yu and Thomas G. Robertazzi, "Divisible Load Scheduling for Grid Computing," semanticsscholar.org, 2003.

[5] R. Chang, J. Chang, and P. Lin, "Balanced Job Assignment Based on Ant Algorithm for Grid Computing," presented at Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference, pp. 291-295, 2007.

[6] S. Lorpunmanee, M. Sap, A. Abdullah, and C. Chompoo-inwai, "An ant colony optimization for dynamic job scheduling in grid environment," International Journal of Computer and Information Science and Engineering, vol. 1(4), pp. 207-214, 2007.

[7] Y. Li, "A Bio-inspired Adaptive Job Scheduling Mechanism on a Computational Grid," International Journal of Computer Science and Network Security (IJCSNS), vol. 6(3), pp. 1-7, 2006.

[8] H. Yan, X. Shen, X. Li, and M. Wu, "An improved ant algorithm for job scheduling in grid computing," presented at Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, vol. 5, pp. 2957-2961, 2005.

[9] M. Dorigo and T. Stützle, "Ant colony optimization," Cambridge, Massachusetts, London, England: MIT Press, 2004.

[10] T. Stutzle and H. Hoos, "MAX-MIN ant system," Future Generation Computer Systems, vol. 16, pp. 889-914, 2000.

[11] Abraham, A., Buyya, R., Nath, "Natures heuristics for scheduling jobs on computational grids," The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000). pp. 45,52. Citeseer (2000)

[12] Lei, chen, jing and bo yang, "Task scheduling algorithm based on pso for grid computing," International Journal of Computational Intelligence Research. Vol.4, No.1 (2008), pp. 37–43.

[13] Chatchawit Aporntewan, Prabhas Chongstitvatana, "A Hardware Implementation of the Compact Genetic Algorithm," IEEE congress of evolutionary computation Seoul Korea, may 2001.

[14] George, Goldberg and lobo, "The Compact genetic algorithm," 1998 IEEE.

[15] Kokilavani.T and George Amalarethinam.D.I, "Applying Non-Traditional Optimization Techniques to Task Scheduling in Grid Computing," International Journal of Research and Reviews in Computer Science, Vol. 1, No. 4, Dec 2010, pp. 34 – 38

[16] Braun, T.D., Siegel, H.J., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., et al. "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," Journal of Parallel and Distributed Computing, Vol. 61, No. 6, pp.810–837, 2001.

[17] He. X, X-He Sun, and Laszewski. G.V, "QoS Guided Min- min Heuristic for Grid Task Scheduling," Journal of Computer Science and Technology, Vol. 18, pp. 442-451, 2003.

[18] Dong. F, Luo. J, Gao. L and Ge. L, "A Grid Task Scheduling Algorithm Based on QoS Priority Grouping," In the Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06), IEEE, 2006.

[19] Etminani .K, and Naghibzadeh. M, "A Min-min Max-min Selective Algorithm for Grid Task Scheduling," The Third IEEE/IFIP International Conference on Internet, Uzbekistan, 2007.

[20] Maheswaran. M, Ali. Sh, Jay Siegel. H, Hensgen. D, and Freund.R.F, "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems," Journal of Parallel and Distributed Computing, Vol. 59, pp. 107-131, 1999.

[21] Ullah Munir. E, Li. J, and Shi. Sh, "QoS Sufferage Heuristic for Independent Task Scheduling in Grid," Information Technology Journal, 6 (8): 1166-1170.

[22] Saeed Parsa, Reza Entezari-Maleki, "RASA: A New Grid Task Scheduling Algorithm," International Journal of Digital Content Technology and its Applications Volume 3, Number 4, December 2009

[23] Cheng, Y.C. and Robertazzi, T.G., "Distributed Computation with Communication Delays," IEEE Transactions on Aerospace and Electronic Systems, Vol. 24, No. 6, Nov. 1988, pp. 700-712.

[24] Sohn, J. and Robertazzi, T.G., "Optimal Load Sharing for a Divisible Job on a Bus Network," IEEE Transactions on Aerospace and Electronic Systems, Vol. 32, No. 1, Jan. 1996, pp. 34-40.

[25] Robertazzi,T.G.,"Processor Equivalence for a Linear Daisy Chain of Load Sharing Processors," IEEE Transactions on Aerospace and Electronic Systems, Vol. 29, No. 4, Oct. 1993, pp. 1216-1221.

[26] Bharadwaj, V., Ghose, D., Mani, V. and Robertazzi, T.G., "Scheduling Divisible Loads in Parallel and Distributed Systems," IEEE Computer Society Press, Los Alamitos CA, Sept. 1996, 292 pages.