REVIEW OF WORKFLOW SCHEDULING ALGORITHMS IN CLOUD/GRID COMPUTING

Salisu Musa Borodo[*]

Aboamama Attahir Ahmed[**]

Abstract

In this paper, a review is carried out on the Workflow scheduling algorithms that have been devised by researchers in cloud/Grid computing environment. The taxonomy of the workflow scheduling algorithms is stated, the constraint the algorithms look into such as make span, user priority, cost. The results obtained by the algorithms after experiments are compared with other algorithms that were used in the same context are also provided.

Keywords – Cloud, Grid, Scientific Workflow, Scheduling

* Universiti Tekonologi Malaysia, Skudai, Johor, Malaysia

** Universiti Tekonologi Malaysia, Skudai, Johor, Malaysia

1. INTRODUCTION

Cloud creates an infrastructure for enabling users to consume services (infrastructure, software and platform) transparently over a secure, shared, scalable, sustainable and standard worldwide network environment. In addition to this flexibility, Cloud infrastructure services provide the illusion of limitless resource allocation for most users. Grids have emerged as a global cyber-infrastructure for the next-generation of e-Science and e-business applications, by integrating large-scale, distributed and heterogeneous resources. More recently, Grid Computing is a high performance computing environment that allows sharing of geographically distributed resources 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. Grid computing has progressed towards a service-oriented paradigm which defines a new way of service provisioning based on utility computing models. Within utility Grids, each resource is represented as a service to which consumers can negotiate their usage and quality of service [1]. A Scientific Workflow Systems is a specialized form of a workflow management system designed specifically to compose and execute a series of computational or data manipulation steps, or a workflow, in a scientific application. Scientific workflows are usually both computation and data-intensive since they often consist of hundreds of thousands of tasks, consume gigabytes and terabytes input datasets and generate similar amounts of intermediate data. Therefore, it is common for scientists to execute these workflows in distributed large scale computational environments such as clusters and grids. There are many scientific problems that require grid environment to get solved. It provides an environment to solve problems in physics, chemistry, nuclear fusion, earth science, space, human health, agriculture, medicine, education, research etc. In medical and biomedical fields, grid computing is useful in digital x-ray image analysis, radiation therapy simulation and protein folding. In chemistry, problems related to quantum chemistry, organic chemistry and polymer modeling makes use of grid computing. In physics, high energy physics, theoretical physics, lattice calculations, combustion and neutrino physics use grid environment. Effective scheduling is a key concern for the execution of performance driven applications, such as workflows in dynamic and cost driven environments including Cloud [2].The main challenge of dynamic workflow scheduling on virtual clusters lies in how to reduce scheduling overhead and handle workload dynamics [3]. This paper would review different workflow scheduling algorithms that have been devised by researchers in the area of cloud/Grid computing, it would also highlight the performance of some selected workflow scheduling algorithms.

2. WORKFLOW SCHEDULING ALGORITHMS IN CLOUD/GRID COMPUTING

Before scheduling tasks in a grid environment, the characteristics of the grid should be taken into account. Some of the characteristics of the grid includes (i) geographical distribution where the resources of grid may be located at distant places (ii) heterogeneity, a grid consists of hardware as well as software resources that may be files, software components, sensor programs, scientific instruments, display devices, computers, supercomputers networks etc (iii) resource sharing, different organizations may own the resources of the grid (iv) multiple administrations, each organization may establish different security and administrative policies to access their resources (v) resource coordination, to get combined computing capabilities, grid resources must be coordinated [4]. Scheduling is highly complicated by the distributed ownership of the grid resources as consumers and providers of the grid resources have their own access policy, scheduling strategy and optimization objectives [5]. 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 etc. There are a number of grid scheduling architectures available. For small set of machines, a centralized architecture with a single scheduler is enough, but it wouldn't scale and not fault tolerant in a geographically distributed systems. User level grid schedulers are used to select the local schedulers to submit the applications [6]. Another approach is one where grid schedulers are organized into a tree structure [7]. Like the above said architectures, 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 [8]. Even though different grid architectures exist, there are also some common features for all the grid schedulers. The grid schedulers deal with organizing the information providers in such a way that the users can have an easy access to the data. They can also recognize the file system or whether any type of resource is cached or which resource is rapidly available. There are two major types of workflow scheduling (see Figure 1), best-effort based and Quality of Service (QoS) constraint based scheduling. The best-effort based scheduling attempts to minimize the execution time ignoring other factors such as the monetary cost of accessing resources and various users' QoS satisfaction levels. On the other hand, QoS constraint based scheduling attempts to minimize performance under most important QoS constraints, for example time minimization under budget constraints or cost minimization under deadline constraints.

A Best-effort based workflow scheduling

Best-effort based workflow scheduling algorithms attempt to complete execution at the earliest time, or to minimize the make span of the workflow application. The make span of an application is the time taken from the start of the application, up until all outputs are available to the user [9]. There are two types, heuristic and meta heuristic

I. Heuristics Based

Individual Task Scheduling: The individual task scheduling is the simplest scheduling method for scheduling workflow applications and it makes schedule decision based only on one individual task. The Myopic algorithm is an example

List Scheduling: A list scheduling heuristic prioritizes workflow tasks and schedules the tasks based on their priorities. There are two major phases in a list scheduling heuristic, the task prioritizing phase and the resource selection phase. We categorize workflow-based list scheduling algorithms as either batch mode(Min min, Max min, Sufferage), dependency mode(HEFT) or dependency-batch mode(Hybrid).

Cluster and Duplication Based Scheduling: Both cluster based scheduling and duplication based scheduling are designed to avoid the transmission time of results between data interdependent tasks, such that it is able to reduce the overall execution time. The cluster based scheduling clusters tasks and assign tasks in the same cluster into the same resource, while the duplication based scheduling use the idling time of a resource to duplicate some parent tasks, which are also being scheduled on other resources. An example is the TANH algorithm which makes of use of both Cluster and Duplication based scheduling

II. Meta heuristics Based

Meta heuristics provide both a general structure and strategy guidelines for developing a heuristic for solving computational problems. They are generally applied to a large and complicated problem. They provide an efficient way of moving quickly toward a very good solution. Many Meta heuristics have been applied for solving workflow scheduling problems, including GRASP, Genetic Algorithms and Simulated Annealing.

III Comparison of Best Effort Scheduling Algorithms

In comparison with the heuristics based scheduling approaches, the advantage of the meta heuristics based approaches is that it produces an optimized scheduling solution based on the performance of entire workflow, rather than the partial of the workflow as considered by heuristics based approach. However, the scheduling time used for producing a good quality solution required by Meta heuristics based algorithms is significantly higher than that of a heuristic algorithm.

B. Quality of Service constraint based workflow scheduling

Many workflow applications require some assurances of quality of services (QoS). For these applications, workflow scheduling is required to be able to analyze users' QoS requirements and map workflows on suitable resources such that the workflow execution can be completed to satisfy users' QoS constraints. However, completing the execution within a required QoS not only depends on the global scheduling decision of the workflow scheduler but also depends on the local resource allocation model of each execution site. The users sometimes may prefer to use cheaper services with a lower QoS that is sufficient to meet their requirements. Two heuristic QoS based scheduling and metaheuristic algorithms are explained below.

1. Deadline constraint Scheduling

Some workflow applications are time critical and require the execution to be completed within a certain time frame. Deadline constrained scheduling is designed for these applications to deliver results before the deadline. The distinction between the deadline constrained scheduling and the best-effort scheduling is that the deadline constrained scheduling also need to consider monetary cost when it schedules tasks. In general, users need to pay for service assessed. The deadline constrained scheduling algorithm intends to minimize the execution cost while meeting the specified deadline constraint.

II. Budget constrained scheduling

Some users would like to execute workflows based on the budget they have available. Budget constrained scheduling intends to minimize workflow execution time while meeting users' specified budgets. Tsiakkouri et al [10] present budget constrained scheduling algorithms called LOSS and GAIN.

III. Meta heuristic Based constrained workflow scheduling

A genetic algorithm (GA) in [11] was also developed to solve the deadline and budget constrained scheduling problem. It defines a fitness function which consists of two components, cost-fitness and time-fitness. For the budget constrained scheduling, the cost-fitness component encourages the formation of the solutions that satisfy the budget constraint. For the deadline constrained scheduling, it encourages the genetic algorithm to choose individuals with less cost.

IV. Comparison of QoS-constraint scheduling algorithms

When comparing two heuristics for the deadline constrained problem, the back-tracking approach is more naive. It is like a constrained based myopic algorithm since it makes a greedy decision for each ready task without planning in the view of entire workflow. It is required to track back to the assigned tasks once it finds the deadline constraint cannot be satisfied by the current assignments. It is restricted to many situations such as data flow and the distribution of execution time and cost of workflow tasks. It may be required to go through much iteration to modify the assigned schedule in order to satisfy the deadline constraint. In contrast, the deadline distribution makes a scheduling decision for each task based on a planned sub-deadline according to the workflow dependencies and overall deadline. Therefore, it has a better plan while scheduling current tasks and does not require tracing back the assigned schedule. Unlike best-effort scheduling in which only one single objective (either optimizing time or system utilization) is considered, QoS constrained scheduling needs to consider more factors such as monetary cost and reliability. It needs to optimize multiple objectives among which some objectives are conflicting. However, with the increase of the number of factors and objectives required to be considered, it becomes infeasible to develop a heuristic to solve QoS constrained scheduling optimization problems. For this reason, we believe that Meta heuristics based scheduling approach such as GA's will play more important role for the multi-objective and multi-constraint based workflow scheduling.

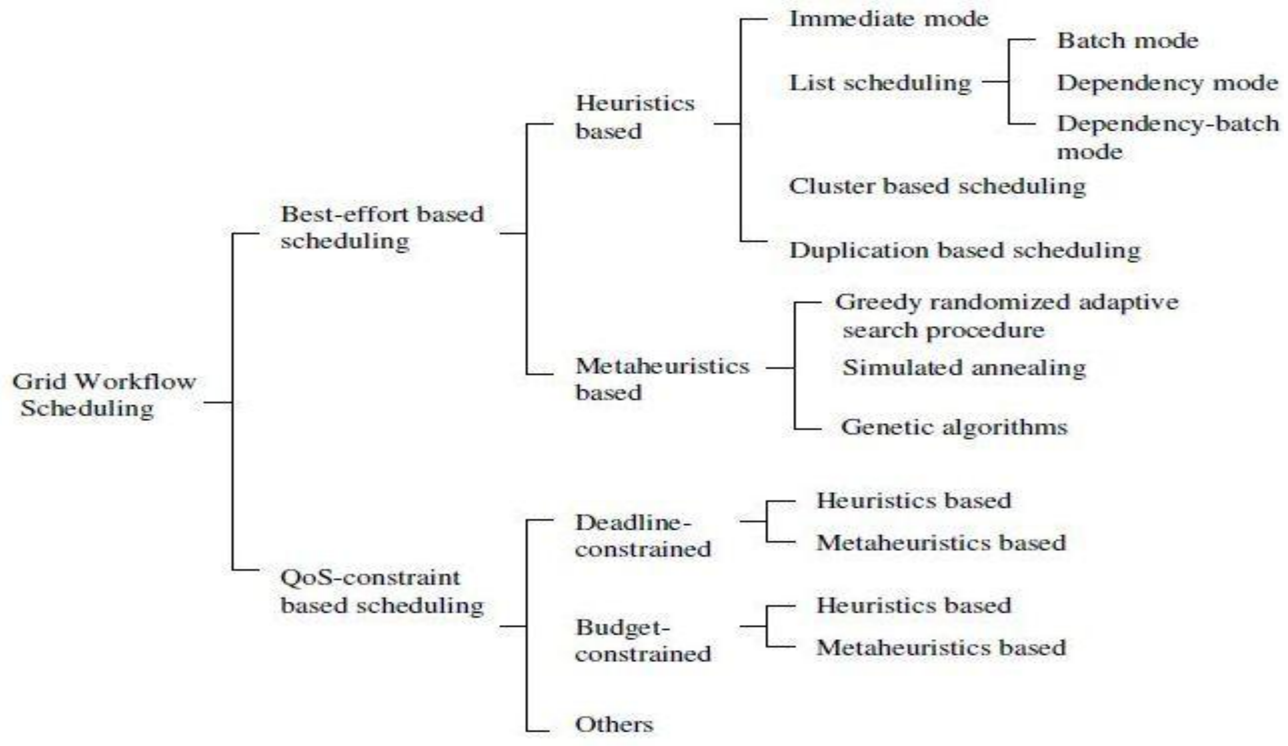

FIGURE1. Taxonomy of Workflow scheduling Algorithms

3. PERFORMANCE OF SOME SELECTED WORKFLOW SCHEDULING ALGORITHMS

The scheduling algorithm used by [12] is based on Iterative Ordinal optimization (IOO), which is an extension of the Ordinal Optimization used for fast dynamic workflow scheduling. The IOO system accommodates dynamic changes in resource provisioning against the workload variations. The purpose is to generate better schedule from a global perspective over a sequence of workload prediction periods. The effectiveness of IOO is measured with Laser Interferometer Gravitational-wave Observatory (LIGO) Data. Results show that as the IOO cluster size scales from 16 to 128, we observe that IOO method has a scheduling overhead reduction of tens or hundreds times than using the Monte Carlo method. Also the IOO method offers 20 to 40% times faster throughput than that of the Blind-Pick method as the task number varies. Priority Impact Scheduling Algorithm (PISA) was used by [13] to tackle the constraint of user priority which may vary based on the fee paid by the user or by something else. The experiment conducted is a simulation which consists of 30 services, 10 users with different priority levels and tasks with a weight that can only be executed by a single service at a time. The results showed that PISA can meet the priority requirement of different users, the most important workflows and tasks can get the resource firstly, so they can be executed successfully and more quickly. [14] Also used the list scheduling method which guarantees the precedence constraints in a workflow application. The algorithm was able to handle data dependency issues related to concurrent tasks being executed. [15] Used the Scalable Heterogeneous Earliest Finish Time (SHEFT) algorithm which is an extension of HEFT, which is applied for mapping a workflow application to a bounded number of processors. Experimental results proved that SHEFT outperforms HEFT in workflow make span. SHEFT also enables resources to scale elastically during workflow execution. Other workflow algorithms look at the economic cost perspective of scheduling rather than the usual make span and time complexity constraints. Some of these economic algorithms are LOSS and GAIN, BATS, PBTS (Partition Balanced Time Scheduling), CCSH, HCOC, Back-Tracking heuristic , Deadline-MDP heuristic , Genetic meta-heuristic implemented in GridBus, CTC scheduling algorithm, SwinDeW-C (Swinburne Decentralized Workflow for Cloud) ScaleStar and BaTS. The ScaleStar is able to compute the monetary cost of scheduling based on two models. The first is based on the Pay for use of Virtual Machine, which considers the customers' perspective. The second model charge fees such as machine rent until users release/return those resource to Cloud Service Providers. Experimental results of Scale Star are compared with LOSS3 (also a HEFT extension). ScaleStar was found to surpass LOSS3 in terms of Computation to Communication Ratio (CCR), Parallelism factor as well as budget types [16].

Sunflower by [17] is an innovative P2P agent-based framework for configuring, enacting, managing and adapting workflows on hybrid Grid-Cloud infrastructures. To orchestrate Grid and Cloud services, Sunflower uses a bio-inspired autonomic choreography model and integrates the scheduling algorithm with a provisioning component that can dynamically launch virtual machines in a Cloud infrastructure to provide on-demand services in peak-load situations. Experimental results carried out on IcarGrid and 5 Eucalyptus instances types that provide an interface similar to Amazon EC2 showed that when the Grid is at its peak due to numerous workflows, the Eucalyptus interface dynamically create instance of Virtual Machines to augment the Grid situation in real time. The algorithms put up by [18],[19] are also similar to Sunflower because the scheduler dynamically load balances in peak periods of workflow execution by making use of underutilized virtual machines or creating new virtual machines on the fly. [20] Designed a service oriented testbed, where experiments can be conducted to evaluate scheduling algorithms, policies, and strategies, mainly focusing on service workflows. The testbed has an emulation service that allows the characterization of workflows through a description of their read and write requirements and execution time of each service that is part of the workflow. Experimental results show that the emulator is capable of mimicking the real application executions. Also, the results show that the emulated execution times for three different scheduling algorithms are equivalent to the real execution times given for the median filter application workflow.

[21] Proposed Revised Discrete Particle Swarm Optimization (RDPSO) to schedule applications among cloud services that takes both data transmission cost and computation cost into account. Experimental results show that the proposed RDPSO algorithm can achieve much more cost savings and better performance on make span and cost optimization. Finally [22] looked at the aspect of parallelizing scientific workflow scheduling due to their computational

complexity as well as the vast amount of data used in the workflows. The work came up with a model based on Quality of Service (Performance and cost) restrictions imposed on the workflow. The model is implemented on top of Scimulus (an infrastructure for running parallel scientific workflows in clouds).

In a real Grid environment, it is hard, and perhaps even impossible, to perform scheduler performance evaluation in a repeatable and controllable manner for different scenarios— the availability of resources and their load continuously varies with time and it is impossible for an individual user/domain to control activities of other users in different administrative domains. To overcome this limitation, [23] developed a Java-based discrete-event Grid simulation toolkit called GridSim that supports discrete-event based simulation of Grid environments to allow repeatable performance evaluation under different scenarios. The toolkit was used by [23] to evaluate the performance of deadline and scheduling algorithms through a series of simulations by varying the number of users, deadlines, budgets, and optimization strategies and simulating geographically distributed Grid resources. [23] Also demonstrated the effectiveness and application of Grid technologies for solving real-world problems such as molecular modeling for drug design on the WWG (World-Wide Grid) testbed.. Some three dimensional graphs (report) of the simulations carried out with GridSim by [23] for Deadline and Budget Constraint Scheduling algorithms are shown below.

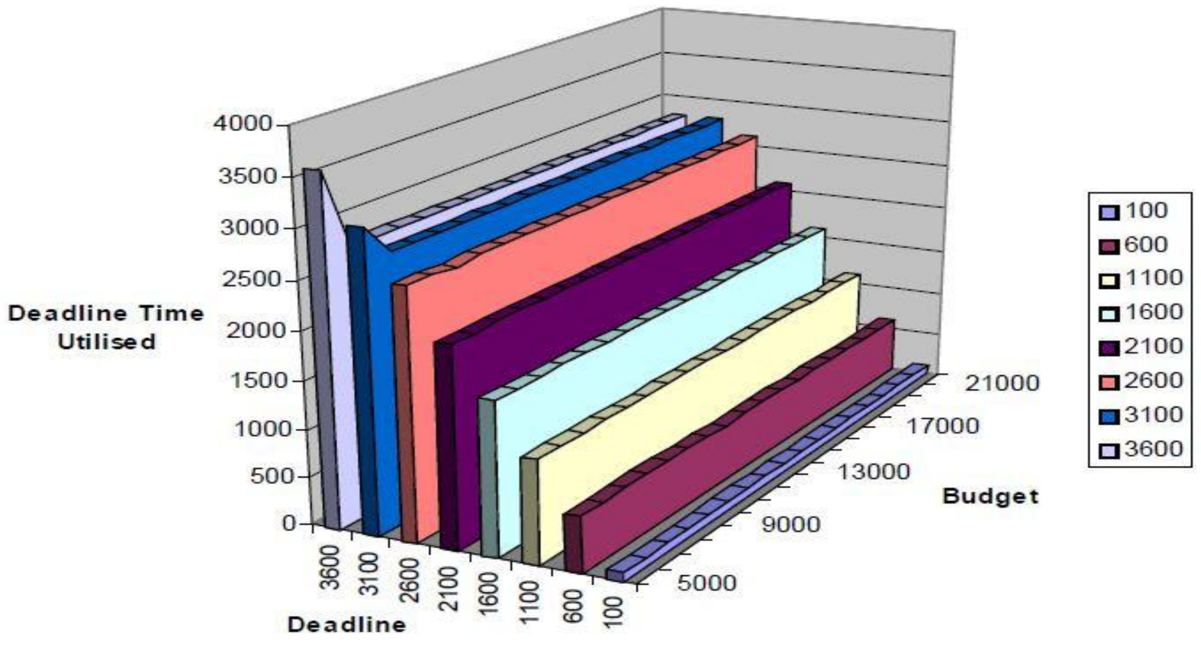

FIGURE2. Deadline time utilized for processing Grid jobs for different values of deadline and budget.

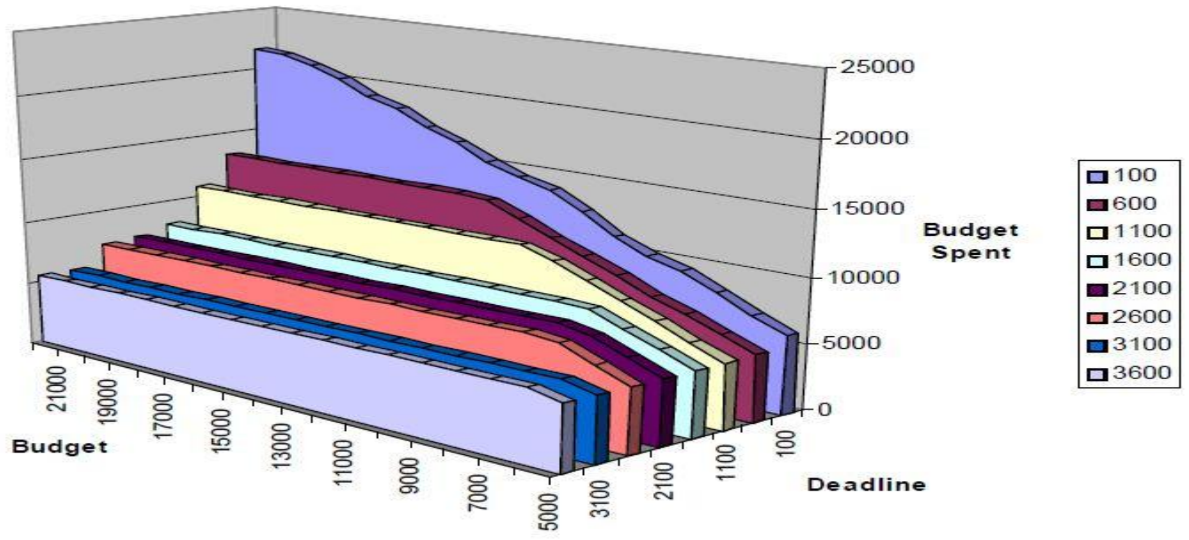

FIGURE3. Budget spent for processing Grid jobs for different values of deadline and budget.

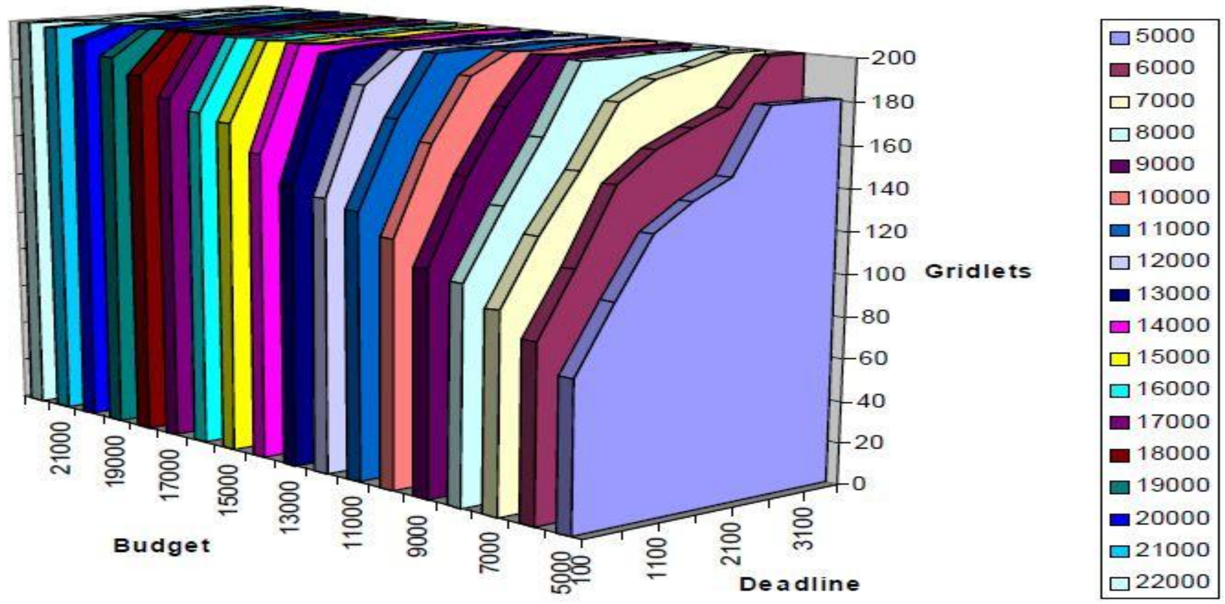

FIGURE4. No. of Gridlets (jobs) processed for different deadline limits with a fixed budget for each.

After reviewing some workflow scheduling algorithms in cloud and computing environments, the recommended are the meta heuristic ones because they provide more optimal solutions despite the higher time they incur. The Simulation framework brought by [23] named GridSim is also appreciated because it overcomes numerous challenges faced by researchers involved in Grid computing.

4 CONCLUSION

In this paper, the Grid/Cloud workflow scheduling algorithms were reviewed which provided a broad understanding of the domain. From the reviewed algorithms no best algorithm is suggested, but a user or researcher would ultimately use the algorithm that suites his requirements, the heuristic algorithms execute in a lesser time, producing less optimal solutions. While the Meta heuristic ones provide more optimal solutions but take longer computational time than the heuristic ones.

REFERENCES

[1] Dr.K.Vivekanandan, D.Ramyachitra (2011) “A Study on Scheduling in Grid Environment”, International Journal on Computer Science and Engineering (IJCSE), 2011

[2] Mustafizur Rahman, Xiaorong Li and Henry Palit. “Hybrid Heuristic for Scheduling Data Analytics Workflow Applications in Hybrid Cloud Environment”. IEEE International Parallel & Distributed Processing Symposium. PP 1, 2011

[3] A. Benoit, L. Marchal, J. Pineau, Y. Robert and F. Vivien. “Resource-aware Allocation Strategies for Divisible Loads on Large-scale Systems”. Proc. of IEEE Int’l Parallel and Distributed Processing Symposium. (IPDPS’09), Rome, Italy. PP 2-4, 2009

[4] Miguel L. Bote-Lorenzo, Yannis A. Dimitriadis, and Eduardo Gomez-Sanchez, “Grid Characteristics and Uses: a Grid Definition,” In the Postproc. Of the first European Across Grid Conference (ACG’03), Springer-Verlag LNCS 2970, pp.219-298, Santago de Compostela, Spain, Feb. 2004.

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

[6] Fran Berman, Rich Wolski, Silvia Figueira, Jennifer Schopf, and Gary Shao, “Application-Level Scheduling on Distributed Heterogeneous Networks,” In Supercomputing ‘96, 1996.

[7] V. Hamscher, U. Schwiegelshohn, A. Streit and R.Yahyapour, “Evaluation of job-scheduling strategies for Grid Computing,” 1st International Workshop on Grid Computing, vol. LNCS 1971, pages 191-202, 2000.

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

[9] 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, pp.500- 507, 2006

[10] H. Casanova et al., Heuristics for Scheduling Parameter Sweep Applications in Grid Environments, In 9th Heterogeneous Computing Workshop (HCW’00), Apr. 2000

[11] E. Tsiakkouri et al., “Scheduling Workflows with Budget Constraints”, In the CoreGRID Workshop on Integrated research in Grid Computing, S. Gorlatch and M. Danelutto (Eds.),

Technical Report TR-05-22, University of Pisa, Dipartimento Di Informatica, Pisa, Italy, Nov. 28-30, 2005, pages 347-357.

[12] J. Yu and R. Buyya, “Scheduling Scientific Workflow Applications with Deadline and Budget Constraints using Genetic Algorithms”, Scientific Programming Journal, 14(3-4): 217 - 230, IOS Press, Amsterdam, The Netherlands, 2006

[13] Hu Wu, Zhuo Tang, Renfa Li, “A Priority Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing”. IEEE. PP 2-4, 2011

[14] Dongjin Yoo, Kwang Mong Sim, “A Scheduling Mechanism for Multiple MapReduce Jobs in a Workflow Application”, IEEE. PP 2-6, 2012

[15] Cui Lin, Shiyong Lu, ”Scheduling Scientific Workflows Elastically for Cloud Computing”. IEEE 4th International Conference on Cloud Computing PP 2-3, 2011

[16] Lingfang Zeng, Bharadwaj Veeravalli, Xiaorong Li, “ScaleStar: Budget Conscious Scheduling Precedence-Constrained Many-task Workflow Applications in Cloud”, 26th IEEE International Conference on Advanced Information Networking and Applications. PP 2-8, 2012

[17] Giuseppe Papuzzo and Giandomenico Spezzano. “Autonomic Management of Workflows on Hybrid Grid-Cloud Infrastructure 2-5”

[18] Zhangjun Wu et al. “A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling”, International Conference on Computational Intelligence and Security, PP 1-3, 2010

[19] Tim D¨ornemann, Ernst Juhnke, Bernd Freisleben, “On-Demand Resource Provisioning for BPEL Workflows Using Amazon’s Elastic Compute Cloud”, 9th IEEE/ACM International Symposium on Cluster Computing and the Grid PP1-4, 2011

[20] Carlos R. Senna, Luiz F. Bittencourt, and Edmundo R. M. Madeira, “An Environment for Evaluation and Testing of ServiceWorkflow Schedulers in Clouds”, IEEE, 2011

[21] Zhangjun Wu et al. “A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling”, 2010 International Conference on Computational Intelligence and Security, PP 1-3

[22] Vitor Viana, Daniel de Oliveira and Marta Mattoso, “Towards a Cost Model for Scheduling Scientific Workflows Activities in Cloud Environments”. IEEE. PP 1-4, 2011

[23] Rajkumar Buyya, “Economic-based Distributed Resource Management and Scheduling for Grid Computing”, IEEE 2002