A Taxonomy and Survey of Scheduling Algorithms in Cloud: based on Task Dependency

Ruby Annette. J
Research Scholar
B.S.Abdur Rahman University

Aisha Banu .W, Ph.D
Professor, Department of CSE
B.S.Abdur Rahman University

Shriram, Ph.D
Professor, Department of CSE
B.S.Abdur Rahman University

ABSTRACT

Cloud computing has made the dream of scalability of resources on demand come true. As the usage of the resources on the cloud involves cost, their optimal utilization is vital. Various scheduling algorithms are being designed and implemented seamlessly to achieve this goal. One of the factors that have a high impact on the scheduling algorithm design is the dependency of the tasks. Dependency implies that the tasks are executed in some precedence order. This survey provides a review of the various scheduling algorithms in cloud mainly from the perspective of task dependency. The broad categorization, advantages and the disadvantages of the various scheduling algorithms available for both dependent and independent tasks are discussed. Based on a comprehensive understanding of the challenges and the current research trends, some open issues worthy of further exploration are proposed.

General Terms

Cloud computing, Task dependency, Independent tasks and Dependent tasks.

Keywords

Resource scheduling, Scheduling algorithms, Hybrid cloud, Task dependency, IaaS.

1. INTRODUCTION

The emergence of various technologies like virtualization, service oriented computing and the availability internet connection of higher bandwidth, has paved the way for the evolution of the cloud computing technology.

Using the cloud computing technology, one can lease the required computing resources, software or a development platform from the cloud service provider and pay as per the usage. This is similar to using the utility services like electricity and paying the bills without worrying about the technical aspects of electricity generation[1].

The three services delivery models of cloud are:

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

In the SaaS model, the software or the applications are hosted over the cloud and are made available to the customers based on the pay-as-per-use model. The advantage of the SaaS is that, the clients need not spent huge amount in buying the software licenses. Google Apps and Salesforce [2] are examples of this model. The PaaS model provides a hosting environment for the client's application. Examples for PaaS model are Google App Engine [3] and Amazon Web Services. The IaaS model lets the client to dynamically scale up or scale down the resources like processing power, storage capacity, network bandwidth etc. Example: Amazon EC2 [4], Eucalyptus [5], etc.

Based on the scalability and pooling up of the resources, the cloud computing is of three types namely:

- Private or Internal Cloud
- Public Cloud
- Hybrid Cloud

The private clouds enable pooling up of the resources owned by the users and utilize them without any charge. Apart from the resources in the private cloud, the public cloud service providers enable the users to scale up or scale down the resources usage according to the demand and charge them in a per-use basis. The hybrid cloud is a combination of both the Public and Private cloud. Thus, the demand for the resources clearly indicates the need for the selection of efficient scheduling algorithms that ensure the optimal utilization of the resources available through all these three types of clouds.

One of the important factors that have a high impact on the selection of the scheduling strategy is the task dependency. Based on the dependency, the tasks may be classified as independent or dependent tasks. The independent tasks have no dependencies among the task and have no precedence order to be followed during scheduling. In contrary, the dependent tasks have task-precedence order to be met during the scheduling process. Thus, the strategies or approaches used for scheduling these types of tasks differ drastically and has been studied widely. To throw light on the various scheduling algorithms available in the literature for both the independent and dependent tasks scheduling, the taxonomy of scheduling algorithm based on the Task dependency is discussed in this survey.

In the earlier works, Casavant et al [6] has proposed a hierarchical taxonomy for scheduling algorithms in general-purpose parallel and distributed computing systems. In [7] Fangpeng et al has extended the taxonomy proposed by Casavant et al for Grid computing environment. Since cloud computing has evolved from grid computing, distributed computing and parallel computing paradigms, the scheduling algorithms taxonomy developed for these systems can also be applied in cloud. However a detailed survey of the algorithms based on task dependency for the cloud computing environment is yet to be done and this survey tries to bridge this gap and extends the taxonomy of scheduling algorithms for grid proposed by Fangpeng et al in [7] to the cloud computing environment based on the task dependency.

The remainder of this paper is organized as follows: An overview of the process of scheduling of tasks in the cloud computing environment is presented in Section 2 with a generalized scheduling architecture. In Section 3, the taxonomy, design and analysis of scheduling algorithms based on the task dependency for cloud computing environment is discussed. The open issues in cloud computing for further research are explored in section 4 and the conclusion is given in Section 5.

2. OVERVIEW OF TASK SCHEDULING

In the cloud computing environment, a task is defined as an atomic unit to be scheduled by the scheduler and assigned to a resource. A job or a meta-task is a set of atomic tasks that is considered for scheduling. The task scheduling is defined as the mapping of tasks to a selected group of resources which may be distributed in multiple administrative domains [7]. The resource required for the completion of the task may be a processor for data processing, a data storage device, or a network link for transporting data. The properties of a task are parameters like CPU/memory requirement, deadline, priority, etc. For example, in a hybrid cloud computing environment the tasks are scheduled onto the resources available in both the public and the private cloud.

2.1 Components of task scheduling in a hybrid cloud

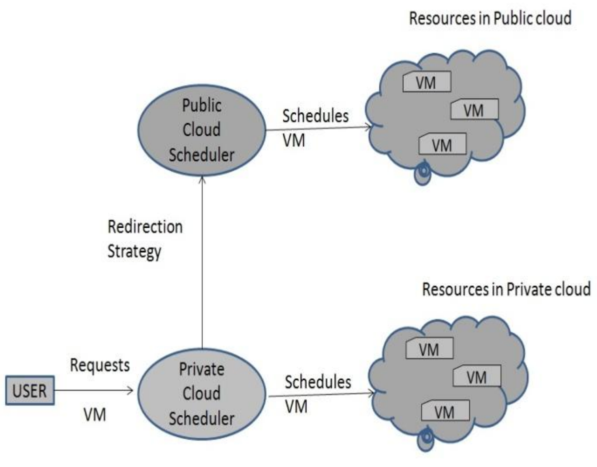

Fig 1: Components of task scheduling in a hybrid cloud

A hybrid cloud is a combination of both a private and a public cloud. The Figure 1 given above describes various the components of scheduling tasks in a hybrid cloud computing environment. The advantage of a hybrid cloud is the scalability of the resources. In a hybrid cloud computing environment the resources already available, is pooled up as a private cloud and extra resources required can be got from the public cloud on demand. To schedule the tasks in a hybrid cloud environment, the user submits the tasks to the scheduler in the private cloud and requests the Virtual Machine (VM) required for completing the task. The private cloud scheduler schedules the tasks on to the available virtual machines in the private cloud based on the scheduling strategy defined. If the task could not be completed using the resources in the private cloud then additional resources required are acquired from the public cloud service provider. A redirection strategy is used to redirect the tasks to the public cloud scheduler and the tasks are scheduled onto virtual machines in the public cloud [8].

3. TAXONOMY OF SCHEDULING ALGORITHMS

The Taxonomy of scheduling algorithms in the cloud environment based on task dependency for both the independent and dependent tasks is given in figure 2.

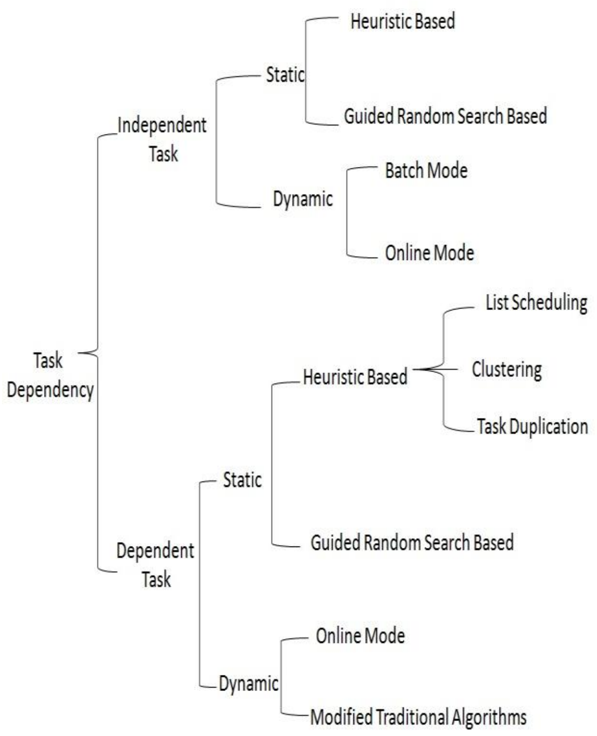

Fig 2: Taxonomy of scheduling algorithms in Cloud based on Task dependency

Based on the task dependency, the tasks can be classified as independent and dependent tasks. The tasks which do not require any communication between the tasks are called independent tasks. The dependent tasks differ from the independent tasks as the former have precedence order to be followed during the scheduling process. The main objective in scheduling the dependent tasks is to minimize the make span which is the total length of the schedule, by decreasing the time taken to execute each node called the Computation cost and the communication cost which is, the time taken to transfer data between the two nodes. Thus, the tasks dependency plays a vital role in deciding the appropriate scheduling strategy.

3.1 Static vs. Dynamic scheduling algorithms

As given in figure 2, the algorithms for task scheduling can be broadly classified as static or dynamic based on the time at which the scheduling or assignment decisions are made. In the case of static scheduling, information regarding all the resources in the cloud and the complete set of tasks as well as all the independent sub tasks involved in a job is assumed to be available by the time the task is scheduled on the cloud. But in dynamic scheduling, a prior knowledge of the resources needed by the task and the environment in which it would be executed is unavailable as the jobs arrive in a real-time mode.

The static scheduling algorithms are further classified as heuristics based and guided random search based [9] algorithms. The heuristics based class of algorithms makes the most realistic assumptions about a priori knowledge concerning process and system loading characteristics. It can be used in the scheduling problem which cannot give optimal answers but only require the most reasonable amount of cost and other system resources to perform their function. The guided random search based algorithms make random choices and guide them through the problem space. These algorithms are also called as "nature's heuristics" [10] as they have a close resemblance to the phenomenon existing in nature. Genetic algorithm is an example of this type, which searches for a near-optimal solution in large solution spaces.

The dynamic scheduling algorithms can be used in two fashions namely on-line mode and batch mode. In the online mode, a task is scheduled onto a machine as soon as it arrives. Each task is scheduled only once and the scheduling result cannot be changed. Hence, on-line mode of dynamic scheduling can be applied, if the arrival rate of the tasks in the real-time is low. However, in the batch mode the tasks are collected into a set that is examined for scheduling at prescheduled times. While online mode heuristics consider a task for scheduling only once, batch mode heuristics consider a task for scheduling at each scheduling event until the task begins execution. [26].

3.2 Scheduling independent tasks

The various static and dynamic algorithms commonly used in scheduling the independent tasks are given in the figure 3 and are discussed briefly below [13], [16], and [26].Independent tasks may be further classified as coarsely grained and fine grained tasks based on the granularity.

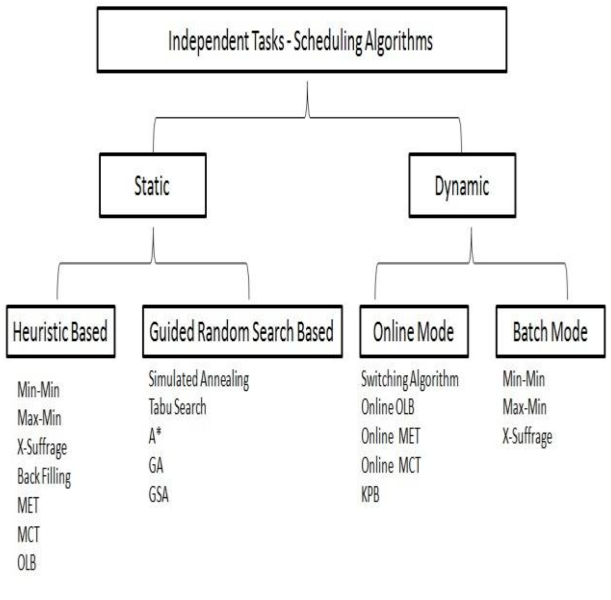

Fig: 3 Classification of Scheduling Algorithms for independent tasks

Granularity is the ratio of computation to the amount of communication. The Coarse-grained tasks communicate data infrequently and only after larger amounts of computations. The fine-grained tasks are small individual tasks in terms of code size and execution time. They have greater potential for parallelism but also greater overheads of processing and communication time.

In [11] Muthuvelu et al and [12] liu et al have proposed scheduling algorithms for grouping fine grained tasks and Scheduling them on to the grid.

3.2.1 Static algorithms for independent tasks

The static algorithms used for scheduling the independent tasks can be broadly classified as Heuristic based and Random search based algorithms. The various heuristic based algorithms are discussed briefly in the following section.

3.2.1.1 Static - heuristic based algorithms

Braun et al have done a detailed comparative study of various static scheduling algorithms for scheduling independent tasks. Some of the popular algorithms are OLB, MET, MCT, Max-Min etc.

Opportunistic Load Balancing (OLB)

The OLB scheduling algorithm [13] tries to keep all the machines or resources available as busy as possible. Thus it assigns each task to the next immediately available machine in a random order. The tasks are assigned randomly irrespective of the expected execution time on that machine.

The advantage of OLB that it is simple and easy to implement as it does not require any extra calculation. The disadvantage in OLB is that the mappings it finds can result in very poor make span as OLB does not consider expected task execution times, when assigning the tasks to the resources.

Minimum Execution Time (MET)

In contrast to OLB, the main objective of the Minimum Execution Time (MET) is to give each task to its best machine. Thus MET assigns each task, in arbitrary order, to the machine with the best expected execution time for that task, regardless of that machine's availability [13], [14]. Though MET is also very simple to implement, it can cause a severe load imbalance across machines as it ignores the availability of machines during the scheduling process.

Minimum Completion Time (MCT)

The motivation behind MCT is to combine the benefits of both OLB and MET and to avoid the circumstances in which OLB and MET perform poorly. The Minimum Completion Time algorithm schedules each task, in arbitrary order to the machine with the minimum completion time for that task [13]. But the limitation of this approach is that all the tasks cannot be assigned to the machines that have the minimum execution time for them.

The min-min heuristic

In the Min-min heuristic, the minimum completion time C for each task to be scheduled is calculated and the task with the overall minimum completion time is selected and assigned to the corresponding machine. Hence the algorithm is given the name Min-min. The scheduled task is removed from the set of tasks to be scheduled, and the process is repeated until all tasks are scheduled[14], [15].

Similar to MCT the Min-min algorithm is based on the minimum completion time. However, MCT considers only one task at a time but the Min-min considers all the unscheduled tasks for each decision making step. The advantage of Min-min is that it assigns the tasks in the order that changes the machine availability status by the least amount that any assignment could. Thus more tasks can be assigned to the machines that complete them the earliest and also execute them the fastest.

The max-min heuristic

In Max-min, the task with the longer execution time is assigned to the best machine available first and is executed concurrently with the remaining tasks with shorter execution times. This helps to minimize the penalties incurred from performing tasks with longer execution times at last.

The Max-min heuristic approach gives a more balanced load across machines and a better make span than the Min-min. Because in Min-min all of the shorter tasks would execute first, and then the longer running task would be executed while several machines sit idle. Thus the Max-min performs better than the Min-min heuristic if the number of shorter tasks is larger than that of longer tasks.

Min-min and Max-min algorithms are simple and can be easily amended to adapt to different scenarios. QoS Guided Min-min heuristic [17], Segmented Min-min algorithm [18] are modifications of the existing min-min algorithm.

Xsuffrage

The idea behind Suffrage [19] approach is, the task that would suffer the most, if it is not assigned to a machine should be given the first priority than the other tasks. The suffrage value of each task is defined as the difference between its best MCT and its second-best MCT. Conventional suffrage algorithms may have problems when the resources are clustered. An improved approach called XSuffrage [20] by Casanova et al, gives a cluster level suffrage value to each task and experiments show it outperforms the conventional Suffrage approach.

Backfilling algorithms

Backfilling is a policy of strategically allowing tasks to run out of order to reduce fragmentation or idle holes. The most popular backfilling strategies are the Conservative, Aggressive and Selective backfilling strategies. In conservative backfilling [21], jobs are given reservation as and when they arrive. Hence, if the jobs are longer, it is very difficult for other jobs to get a reservation ahead of previously arrived jobs. With Aggressive Backfilling [22], only the request at the head of the waiting queue called the pivot is granted a reservation and other requests are allowed to move ahead in the queue if they do not delay the pivot. Under selective backfilling [23] a request is granted a reservation if its expected slowdown exceeds a threshold. In [8] Assuncao et al have evaluated seven strategy sets including the backfilling algorithms to Evaluate the Cost-Benefit of Using Cloud Computing to Extend the Capacity of Clusters.

3.2.1.2 Static - random search based algorithms for independent task

Genetic Algorithm

GA is an evolutionary technique to perform search in a large solution space. In GA the various steps like population selection, seeding, crossover, and mutation are carried out for mapping the tasks on to the machines. While the advantages of the GAs are the generation of the good quality of output schedules, the disadvantages are: The scheduling times are usually much higher than the heuristic- based techniques. Also, the optimal set of control parameters obtained for one set of task scheduling may not be optimal for another set of tasks [24].

Simulate Annealing (SA)

SA is a search technique based on the physical process of annealing, which is the thermal process of obtaining low-energy crystalline states of a solid. SA theory states that if temperature is lowered sufficiently slowly, the solid will reach thermal equilibrium, which is an optimal state. By analogy, the thermal equilibrium is an optimal task-machine mapping (optimization goal), the temperature is the total completion time of a mapping (cost function), and the change of temperature is the process of mapping change. If the next temperature is higher, which means a worse mapping, the next state is accepted with certain probability [25].

Tabu Search (TS)

Tabu Search is a meta-strategy for guiding known heuristics to overcome local optimality. It is an iterative technique which explores a set of problem solutions, denoted by X, by repeatedly making moves from one solution to another solution *s'* located in the neighborhood *N(s)* of *s*. These moves are performed with the aim of efficiently reaching an optimal solution by minimizing some objective functions [13].

GSA

The Genetic Simulated Annealing (GSA) [25] heuristic is a combination of the GA and SA techniques. In general, GSA follows procedures similar to the GA. However, for the selection process, GSA uses the SA cooling schedule and system temperature and a simplified SA decision process for accepting or rejecting a new chromosome.

*A**

A* is a tree based search heuristic beginning at a root node that is a null solution. As the tree grows, nodes represent partial scheduling (a subset of tasks is assigned to machines), and leaves represent final scheduling (all tasks are assigned to machines). The partial solution of a child node has one more task scheduled than the parent node. Each parent node can be replaced by its children. To keep execution time of the heuristic tractable, there is a pruning process to limit the maximum number of active nodes in the tree at any one time. If the tree is not pruned, this method is equivalent to an exhaustive search. This process continues until a leaf (complete scheduling) is reached.

3.2.1.3 Dynamic algorithms for independent tasks

Online mode

The OLB, MET, MCT algorithms discussed in the previous section can also be used to schedule independent tasks dynamically in the online mode [7],[26]. The other algorithms include Switching Algorithm and the k-percent best (KPB).

SA (Switching Algorithm)

SA (Switching Algorithm) uses the MCT and MET heuristics in a cyclic fashion depending on the load distribution across the machines. MET can choose the best machine for tasks but might assign too many tasks to same machines, while MCT can balance the load, but might not assign tasks machines that have their minimum executing time. If the tasks are arriving in a random mix, it is possible to use the MET at the expense of load balance up to a given threshold and then use the MCT to smooth the load across the machines.

K-Percent Best (KPB)

KPB (K-Percent Best) heuristic considers only a subset of machines while scheduling a task. The subset is formed by picking the k best machines based on the execution times for the task. A good value of k schedules a task to a machine only within a subset formed from computationally superior machines. The purpose is to avoid putting the current task onto a machine which might be more suitable for some yet-to-arrive tasks, so it leads to a shorter make span as compared to the MCT.

Batch mode

The various scheduling algorithms discussed earlier for scheduling the independent tasks statically can also be used to schedule tasks dynamically in the batch mode. Examples of this type of algorithms include the Min-min heuristic, Max-min heuristic, Suffrage heuristic, etc.

3.3 Scheduling algorithms for dependent tasks

The Classification of the various Scheduling Algorithms for Dependent tasks is given in the figure 4. The static and the dynamic scheduling algorithms widely used for scheduling dependent tasks [9] are discussed briefly in the following section.

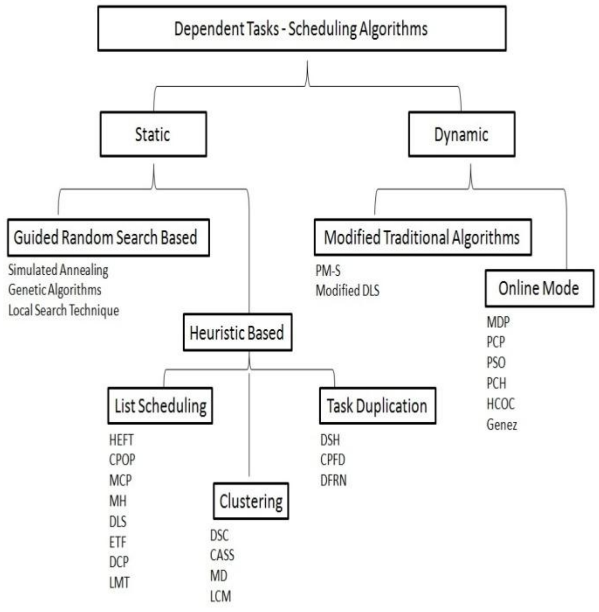

Fig: 4 Classification of Scheduling Algorithms for dependent tasks.

3.3.1 Static algorithms for dependent tasks

The static algorithm for independent coarsely grained tasks are classified into two types namely the heuristic-based and the guided random-search-based algorithms. The heuristic-based algorithms can be further classified into three types as list scheduling heuristics, Clustering heuristics, and task duplication heuristics. In [27], Topcuoglu et al has discussed briefly about all the static algorithms for dependent tasks mention below.

List scheduling heuristics

In the list scheduling heuristics, as the name suggests, all the tasks of the given graph are listed according to the priorities and an ordered list of tasks is constructed. Then the tasks are selected in the order of their priorities and scheduled to a processor which minimizes a predefined cost function. The advantages of the list scheduling heuristics is that they are generally more practical, provide good quality of schedules and also provide better performance results at a lower scheduling time than the other categories.

The Modified Critical Path (MCP), Dynamic level Scheduling (DLS), Mapping Heuristic (MH), Levelized - Min Time (LMT), The Heterogeneous-Earliest-Finish-Time (HEFT) and the Critical-Path-on-a-Processor (CPOP) algorithms are some of the examples of the list scheduling heuristics [9].

Clustering heuristics

Clustering algorithms are generally applied for unbounded number of processors. In this, the algorithm maps the tasks in a given graph to an unlimited number of clusters. The task selected for clustering can be ready task or any task. The previous clustering is refined at every iteration step by merging some clusters. If two tasks are assigned to the same cluster, they will be executed on the same processor. Since clustering algorithms are generally applied for unbounded number of processors, a cluster merging step is required in the second phase to merge the task clusters generated by the algorithm onto a bounded number of processors and a task ordering step is required to order the task executions within each processor. Examples of this type of algorithms are: Dominant Sequence Clustering (DSC), Linear Clustering method, Mobility Directed and Clustering and Scheduling system (CASS).

Task duplication heuristics

The idea behind the Task duplication algorithms is to reduce the inter process communication overhead by mapping the tasks redundantly. Duplication – based algorithms differ according to the selection strategy of the tasks for duplication. However, the task duplication based heuristics are not practical because of their significantly high time complexity than other categories of algorithms. Examples of this group of algorithms include: Critical Path Fast Duplication, Duplication Scheduling Heuristic, Bottom-Up Top-Down Duplication Heuristic, Duplication First and Reduction Next.

3.3.2 Dynamic algorithms for dependent tasks

3.3.2.1 Online mode algorithms

The online mode scheduling algorithms for scheduling the dependent tasks dynamically are discussed briefly in the following section.

Deadline-Markov Decision Process (MDP)

The deadline-driven cost-minimization algorithm [28], or the Deadline-Markov Decision Process (MDP), breaks the DAG into partitions, assigning a maximum finishing time for each partition according to the deadline set by the user. Based on this time, each partition is scheduled for that resource which will result in the lowest cost and earliest estimated finishing time. This algorithm works with on-demand resource reservation.

Partial Critical Paths (PCP)

Abrishami et al. [29] presented the Partial Critical Paths (PCP) algorithm which schedules the workflow in a backwards fashion. Constraints are added to the scheduling process when such scheduling of jobs in a partial critical path fails, so that the algorithm will be re-started. This algorithm presents the same characteristics as does MDP, although it involves greater time complexity, since a relatively large number of re-scheduling can be demanded during the execution of the algorithm.

Particle Swarm Optimization (PSO)

The self-adaptive global search optimization technique called particle swarm optimization (PSO) is utilized to schedule workflows in the algorithm proposed in [30] and which was developed to work in clouds with a single-level SLAs and on-demand resource leasing. It considers neither multi-core resources nor workflow deadlines, but focuses solely on monetary cost minimization.

Hybrid Cloud Optimized Cost (HCOC)

The Hybrid Cloud Optimized Cost (HCOC) algorithm [31], schedules workflows in hybrid clouds by first attempting costless local scheduling using HEFT[9]. If the local scheduling cannot meet the deadline, the algorithm selects jobs for scheduling in resources from the public cloud. As with MDP algorithm, the objective is to minimize the financial cost obeying the deadlines stipulate by the user in a single-level SLA contract.

3.3.2.2 Traditional algorithms modified for dynamism of cloud

Considering the dynamism of cloud, In [32], the authors propose a pM-S algorithms which extends a traditional dynamic Master-Slave scheduling model. In [33], a derived algorithm based on the Dynamic Level Scheduling (DLS) is proposed. In the original DLS, the dynamic level of a task in a DAG is used to adapt to the heterogeneity in resources, while in the newly proposed algorithm, the dynamic length of the queue on each resource is also taken into account for computing a task's level. To estimate the length of the queue, it is assumed that jobs are coming following a Poisson distribution.

4. OPEN ISSUES

The impact of communication networks and the capacities of the communication links connecting the available resources on the scheduling decisions needs to be explored further. In a hybrid cloud computing environment, the open issues for further research are the splitting of a workflow of dependent tasks to be executed in private and public cloud, deciding on whether task should be executed on private cloud or public cloud etc. Another challenging issue in hybrid clouds is how interfaces can be provided to interact automatically with different existing public clouds, so that the broker can gather information about resources and the workflow executed and monitored in a variety of public cloud infrastructures.

5. CONCLUSION

Although task scheduling strategies used in parallel and distributed systems form the base for the cloud computing, the dynamism and heterogeneity of the cloud computing environment poses various new challenges and makes task scheduling an interesting topic for research. In this literature review, the broad classification, advantages and disadvantages of the various current scheduling algorithms working in the cloud computing scenario based on the task dependency are discussed. The task scheduling problem in Cloud computing and the other open issues are also discussed briefly. Thus it could be concluded that the heterogeneity, dynamism, computation, data separation, cost reduction, and providing the Quality of Service (QoS) are the primary challenges concerned in the cloud computing environment.

6. REFERENCES

[1] M. Armbrust et al., Above the Clouds: A Berkeley View of Cloud Computing, tech. report UCB/EECS-2009-28, EECS Dept., Univ. of California, Berkeley, Feb. 2009.

[2] Salesforce.com

[3] Google App Engine: http://www.google.com/apps

[4] Amazon EC2: http://aws.amazon.com/ec2

[5] Eucalyptus http://eucalyptus.com

[6] T. Casavant, and J. Kuhl, A Taxonomy of Scheduling in General-purpose Distributed Computing Systems, in IEEE Trans. on Software Engineering Vol. 14, No.2, pp.141--154, February 1988.

[7] F. Dong and S. G. Akl, Scheduling algorithm for grid computing: state of the art and open problems, Technical Report of the Open Issues in Grid Scheduling Workshop, School of Computing, University Kingston, Ontario, January, 2006.

[8] M. Dias de Assunção, A. di Costanzo, and R. Buyya, "Evaluating the Cost-Benefit of Using Cloud Computing to Extend the Capacity of Clusters," Proc. Int'l Symp.

[9] Haluk Topcuoglo, Salim Hariri, Min-You Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Transactions on parallel and distributed systems, No.3, March 2002.

[10] A. Abraham, R. Buyya and B. Nath, Nature's Heuristics for Scheduling Jobs on Computational Grids, in Proc. of 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), pp. 45-52, Cochin, India, December 2000.

[11] N. Muthuvelu, J. Liu, N. L. Soe, S.rVenugopal, A. Sulistio and R. Buyya, A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids, Proceedings of the 3rd Australasian Workshop on Grid Computing and e-Research (AusGrid 2005), Newcastle, Australia, January 30 – February 4, 2005.

[12] Quan Liu, Yeqing Liao, "Grouping based Fine-Grained job Scheduling in Grid Computing", First International Workshop on Education Technology and Computer Science, Vol.1,pp. 556-559, IEEE, 2009.

[13] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson,M. Theys, B. Yao, D. Hensgen and R. Freund, A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems, in J. of Parallel and Distributed Computing, vol.61, No. 6, pp. 810-837, 2001.

[14] R.Armstrong, D.Hensgen, and T.Kidd, "The relative performance of various mapping algorithms is independent of sizable variance in run-time predictions," 7th IEEE Heterogeneous Computing Workshop (HCW'98), March. 1998, pp.79 87

[15] R. F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D.Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore,B. Rust, and H. J. Siegel, "Scheduling resources in multi-user, heterogeneous,computing environments with SmartNet," 7th IEEE HeterogeneousComputing Ubrkshop (HCW '98), Mar. 1998, pp. 184-199.

[16] 0. H. Sbarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," Journal of the ACM, Vol. 24, No. 2, Apr. 1977, pp. 280-259.

[17] X. He, X. Sun and G. Laszewski, A QoS Guided Min-Min Heuristic for Grid Task Scheduling, in J. of Computer Science and Technology, Special Issue on Grid Computing, Vol.18, No.4, pp.442--451, July 2003.

[18] M. Wu, W. Shu and H. Zhang, Segmented Min-Min: A Static Mapping Algorithm for Meta-Tasks on Heterogeneous Computing Systems, in Proc. of the 9th Heterogeneous Computing Workshop (HCW'00), pp. 375--385, Cancun, Mexico, May 2000.

[19] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen and R. F. Freund, Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems, in J. of Parallel and Distributed Computing, Vol. 59, No. 2,pp.107--131, November 1999.

[20] H. Casanova, A. Legrand, D. Zagorodnov and F. Berman, Heuristics for Scheduling Parameter Sweep Applications in Grid Environments, in Proc. of the 9th Heterogeneous Computing Workshop (HCW'00), pp. 349-363, Cancun, Mexico, May 2000.

[21] Mu'alem, A.W., Feitelson, D.G.: Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. IEEE Trans. Parallel Distrib. Syst. 12(6), 529–543 (2001)

[22] Lifka, D.A.: The ANL/IBM SP scheduling system. In: Workshop on Job Scheduling Strategies for Parallel Processing (IPPS'95), London, UK, 1995, pp. 295–303. Springer, Berlin (1995)

[23] Srinivasan, S., Kettimuthu, R., Subramani, V., Sadayappan, P.: Selective Reservation strategies for backfill job scheduling. In: 8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP '02), London, UK, 2002. LNCS, vol. 2537, pp.55–71. Springer, Berlin/Heidelberg (2002)

[24] S. Song, Y. Kwok, and K. Hwang, Security-Driven Heuristics and A Fast Genetic Algorithm for Trusted Grid Job Scheduling, in Proc. of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), pp.65-74, Denver, Colorado USA, April 2005.

[25] Y. Liu, Survey on Grid Scheduling (for Ph.D Qualifying Exam), Department of Computer Science, University of Iowa, http://www.cs.uiowa.edu/~yanliu/, April 2004.

[26] M. M. Shoukat, M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems". Journal of Parallel and Distributed Computing, 59:107–131, 1999.

[27] Haluk Topcuoglo, Salim Hariri, Min-You Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Transactions on parallel and distributed systems, No.3, March 2002.

[28] J. Yu, R. Buyya, and C. K. Tham, "Cost-based scheduling of scientific workflow applications on utility grids," in International Conference on e-Science and Grid Computing, Jul. 2005, pp. 140–147.

[29] S. Abrishami, M. Naghibzadeh, and D. Epema, "Cost-driven scheduling of grid workflows using partial critical paths," in 11th IEEE/ACM International Conference on Grid Computing (GRID), Oct. 2010, pp. 81 –88.

[30] S. Pandey, L. Wu, S. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud Computing environments," in 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), Apr. 2010, pp. 400 –407.

[31] L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," Journal of Internet Services and Applications, vol. 2, no. 3, Dec 2011, pp. 207–227.

[32] Tianchi Ma and Rajkumar Buyya, Critical-Path and Priority based Algorithms for Scheduling Workflows with Parameter Sweep Tasks on Global Grids, in Proc. of the 17th International Symposium on Computer Architecture and High Performance Computing, Rio de Janeiro, Brazil, October 2005.

[33] M. Iverson and F. Ozguner, Dynamic, Competitive Scheduling of Multiple DAGs in a Distributed Heterogeneous Environment, in Proc. of Seventh Heterogeneous Computing Workshop, pp. 70-78, Orlando, Florida USA, March 1998.