ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 11, May 2013

A Study on Task Scheduling Algorithms in Cloud Computing

Vijayalakshmi A. Lepakshi, Dr. Prashanth C S R

Abstract— Cloud computing is a new computing model using which applications, data and IT services are provided over the Internet. Cloud computing shares data and offers services transparently among its users. With the increase in number of users of cloud the tasks to be scheduled increases. The performance of cloud depends on the task scheduling algorithms used in the scheduling components or brokering components. Scheduling parallel applications modeled by Directed Acyclic Graphs onto a network of heterogeneous computers is a NP-Complete problem. A number of algorithms have been proposed in the past to solve the task-scheduling problem for heterogeneous network of computers. However, none of these algorithms can be extended to cloud computing systems which are also heterogeneous computing systems. Since cloud computing systems have a high degree of unpredictability with respect to resource availability and network bandwidth, task scheduling algorithms for cloud computing systems should incorporate the latency caused by unpredictable resource availability. The present study involves surveying the different task scheduling algorithms developed for cloud environment.

Index Terms— **Cloud Computing, Directed Acyclic Graph Scheduling, Infrastructure as a Service (IaaS), Task Scheduling, Virtualization.**

I. INTRODUCTION

In recent years, with the growth of Internet and its services, a new computing model, Cloud Computing has appeared. A Cloud is a Distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically presented as unified computing resources. This model utilizes the vacant resources of computer globally increasing the economic efficiency through improving utilization rate, and decreases the equipment energy consumption. It aims to share data, services and resources among its users. Cloud computing provides application services over the Internet and it provides computing as a utility to its users. It allows its users to lease resources and services in a pay as you use manner and release them when they are no longer useful. Cloud computing provides different services to different users such as SaaS, IaaS, and PaaS [10]. Users can avail these services in a Pay per- Use-On-Demand model, which can access shared IT resources like server, data storage, application, network and so on through internet. IaaS offers storage and compute resources that developers and IT organizations use to deliver custom business solutions. The customer accesses those services with defined interfaces. In cloud computing the underlying infrastructure that provides the service may be very sophisticated indeed. However, the user doesn't necessarily need to understand this infrastructure to use it. The IaaS customer rents computing resources instead of buying and installing them in their own data center. The service may include dynamic scaling so that if the customer winds up needing more resources than expected, he can get them immediately, and release them on completion. Organizations with similar needs for additional computing resources may boost their own data centers by renting the computer hardware — appropriate allocations of servers, networking technology, storage, and data center space — as a service. Instead of laying out the capital expenditure for the maximum amount of resources to cover their highest level of demand, they purchase computing power when they need it. As the number of users of Cloud computing Systems increased, the tasks to be scheduled in Cloud increased proportionally. Therefore, there is a need for better algorithms to schedule tasks on these systems. Algorithms required to schedule tasks are service oriented and differ in different environments.

II. LITERATURE REVIEW

A. *Cloud Architecture*

The two most significant components of cloud computing architecture are known as the front end and the back end. The front end is the part seen by the client, i.e. the computer user. This includes the client's network (or computer) and the applications used to access the cloud via a user interface such as a web browser. The back end of the cloud computing architecture is the 'cloud' itself, comprising various computers, servers and data storage devices. Cloud computing sample architecture is shown in Fig. 1.

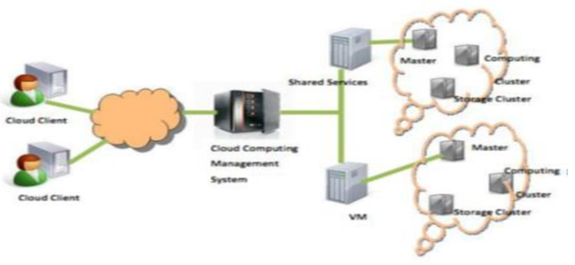

Fig 1 Cloud Computing Architecture

B. *Concept of Cloud Computing*

Based on the observation of what Clouds are promising to be, this paper follows the definition of cloud computing proposed in [5]: "A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 11, May 2013

presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers.”

C. Infrastructure as a Service

Infrastructure as a Service is one of the example of cloud computing. IaaS offers storage and compute resources that developers and IT organizations use to deliver custom business solutions. The customer accesses those services with defined interfaces. These interfaces are, in fact, all that the user ever comes in contact with. In cloud computing the underlying infrastructure that provides the service may be very sophisticated indeed. However, the user doesn't necessarily need to understand this infrastructure to use it. The IaaS customer rents computing resources instead of buying and installing them in their own data center. The service may include dynamic scaling so that if the customer ends up needing more resources than expected, he can get them immediately, and release them on completion.

D. Task Scheduling

As the number of users of Cloud computing Systems increased, the tasks to be scheduled in Cloud increased proportionally. Therefore, there is a need for better algorithms to schedule tasks on these systems. Algorithms required to schedule tasks are service oriented and differ in different environments. Task Scheduling algorithms in cloud computing aim at minimizing the makespan of tasks with minimum resources efficiently. Cloud computing, uses low-power hosts to achieve high usability. The cloud computing refers to a class of systems and applications that employ distributed resources to perform a function in a decentralized manner. Cloud computing is to utilize the computing resources (service nodes) on the network to facilitate the execution of complicated tasks that require large-scale computation. Thus, the selecting nodes for executing a task in the cloud computing must be considered [8]. A task is an activity that uses set of inputs to produce a set of outputs. In Cloud computing, user applications will run on virtual systems where distributed resources are allocated dynamically. Dynamic load-balancing mechanism has to allocate tasks to the processors dynamically as they arrive. Redistribution of tasks has to take place when some processors are overloaded. Every application is completely different in nature and independent where some require more CPU time to compute complex task, and some others may need more memory to store data. Different scheduling algorithms can be used depending on the type of the task to be scheduled. The scheduling algorithms can utilize better executing efficiency and maintain the load balancing of system. The efficiency of the cloud depends on the algorithms used for task scheduling.

E. Directed Acyclic Graph Scheduling

Parallel tasks are often represented by a directed acyclic graph (DAG). The directed acyclic graph (DAG) is a generic model of the task of a parallel program consisting of a set of processes (nodes) among which there are dependencies. In general, a DAG is defined by the tuple G = (V, E), V (lVI = v) is a set of nodes, representing the tasks, and E (lEI = e) is a set of directed edges, representing the communication messages. The weight on a node is called the computation cost of a node ni and is denoted by w (ni). The edges in the DAG, each of which is denoted by (ni, nj), correspond to the communication messages and precedence constraints among the nodes. The weight on an edge is called the communication cost of the edge and is denoted by c (ni, nj). A Critical Path (CP) of a DAG is a set of nodes and edges constituting a path which has the largest length. The length of the CP is the sum of the computation costs and communication costs along the path.

F. Scheduling Algorithms

a. A Priority Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing[13]

In this paper, a Priority Impact Scheduling Algorithm (PISA) [13], is proposed in which user's Priority is taken in to account. The differences between users' priority may base on the fee they paid. First, a prototype algorithm based on the workflow's priority is used to classify the workflow, where the priority is set by the users themselves. The priority may be based on the fee that user could pay for the workflow. Service provider may classify the users as free users, VIP; et al. Higher priority workflow can access the resource faster. If a low priority workflow can't access the resource for a long time, then set a counter value initially to 0 and after each round of scheduling increment it by 1. After many rounds if the value exceeds a fixed value then workflow can access a resource regardless of of its priority in the next round. Secondly, Access Policy Check procedure is used in which, When a workflow requests the cloud computing resource provider for services, the provider first queries the Access Strategy library. We define one policy name as WAPC; the definition of WAPC is as follow: WAPC[(workflow level, int priority), Task, (T_s, T_e, P)]. T_s is the start time of Task T_i, T_e is the end time Task T_i, *P* is a periodic expression. The value of priority determines the highest level of cloud resource it can get. If the request level consistent with the priority value and its request meet the time quota, the application is accepted. If the value is not big enough or the time quota expires, the application is denied. We define Time quota variance T_q, It means the execution time limit of task.

$$T_q = Min(T_i) + AT(T_i)(T_e - T_s) / \sum_{i=0}^{n} T_i$$

Min (T_i) is the minimum time of the task T_i executed.

$\sum_{i=1}^{n} T_i$ Is the time variance sum of all task in the workflow.

AT (T_i) is time variance which the task executed on all the services.

If the workflow is accepted by the system, it then enters the schedule sequence. The workflow composed of many tasks, for a workflow W, it has the priority value W_{pi}.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 11, May 2013

The task T_i's position in the scheduling sequence can calculated as follows: $W_{pi}*a+T_{pi}*b$, where a is called first class impact factor, b is called second class impact factor. Factor a signifies the workflow's priority, b signifies task weight.
The values of *a and b are* calculated as follows:
For a series of N workflows $W_1, W_2, ..., W_n$, a user can specify their relative importance, so the weights of workflows can be $Wp1, Wp2, Wp3, ..., Wpn$, the average weight awp is

$$awp = \left(\sum_{i=1}^{n} w_{pi}\right)/n$$

For workflow W_i, difference degree is W_{pi}-awp, so $a=(W_{pi}awp)/\ awp$.
For a workflow Wi, it includes n tasks, a user may specify the relative importance of n tasks as $Tp1, Tp2, Tp3, ..., Tpn$, Wpi is the sum of n tasks ,so the average weight atp is

$$atp = \left(\sum_{i=1}^{n} T_{pi}\right)/n$$

Every task's difference degree is T_{pi}-atp, so $b=(T_{pi}atp)/atp$.

b. BAR: An Efficient Data Locality Driven Task Scheduling Algorithm for Cloud Computing[1]

In this paper, a heuristic task scheduling algorithm called BAlance-Reduce (BAR)[1], is proposed, in which an initial task allocation will be produced at first, then the job completion time can be reduced gradually by tuning the initial task allocation. By considering the network state, BAR can adjust data locality dynamically. In this system, a set of independent tasks on a homogeneous platform with m tasks and n servers is considered, where each task processes an input block on a server. A job is not completed until all tasks are finished. Cloud computer Clusters workload is considered to design an allocation strategy that minimizes the make span of tasks. BAR called Balance-Reduce is a data locality driven task scheduling algorithm, which finds a good solution in time $O(\max\{m+n, n \log n\}\cdot m)$. BAR is split into two phases, *balance* and *reduce*: In Balance phase, a balanced total allocation is produced where all tasks are allocated to their preferred servers evenly. In Reduce phase, we generate a sequence of total allocations, and reduce the make span iteratively.

c. Adaptive Resource Allocation for Pre-emptable Jobs in Cloud Systems[2]

In this paper, an adaptive resource allocation algorithm for the cloud system with pre-emptable tasks [2] is proposed. These algorithms adjust the resource allocation adaptively based on the updated state of the actual task executions. The experimental results show that these algorithms work significantly in intense resource contention situation. In this paper, an infrastructure-as-a-service (IaaS) cloud system is considered. The computational resources are available in the form of virtual machines (VMs) deployed in a provider's data center. In Cloud Computing, three different modes of renting the computing resources are available, such as, Advance Reservation (AR), Best-effort and Immediate from a cloud provider. To overcome the problems involved in resource utilization, AR and best-effort can be combined. In this paper, it is assumed that a few of jobs submitted in the cloud system are in the AR mode, while the rest of the jobs are in the best-effort mode. And the jobs in AR mode have higher priorities, and are able to preempt the executions of the best-effort jobs. In this paper, two algorithms for the task scheduling: *adaptive list scheduling* (ALS) and *adaptive min-min scheduling* (AMMS)[2] are proposed. Once a job is submitted to a scheduler, it will be partitioned into tasks in the form of DAG. Both ALS and AMMS include a static task scheduling for resource allocation. Then the scheduler will repeatedly re-evaluate the remaining static allocation with a pre-defined frequency, based on the latest information of task execution. To generate the static allocation two greedy algorithms, the Cloud List Scheduling (CLS) and the Cloud Min-Min Scheduling (CMMS) are used. The actual finish time of a task may be different from the estimated, due to the resource contention within the cloud. In this paper, an online adaptive scheduling procedure is proposed to adjust the resource allocation dynamically based on the latest information.

d. Improved Cost-Based Algorithm for Task scheduling in Cloud computing[4]

Cost of each task differs depending on user task schedule. Scheduling of user tasks in cloud is not the same as in traditional scheduling methods. In this paper, an improved cost-based scheduling algorithm [4] for making efficient mapping of tasks to available resources in cloud is proposed. This algorithm measures both resource cost and computation performance for scheduling, and improves the computation/communication ratio by grouping the user tasks according to a particular cloud resource's processing capability and sends the grouped jobs to the resource. In parallel computing environments, in the context of Directed Acyclic Graph (DAG) scheduling [7], grouping of jobs is done to reduce communication dependencies among them as clustering is presented. In this paper, the proposed scheduling strategy focuses on grouping independent jobs with small processing requirements, into suitable jobs with larger processing requirements and schedules them in accordance with in-deterministic network conditions. Traditional way of task scheduling in cloud computing uses the direct tasks of users as the overhead application base. The problem is that there may be no relationship between the overhead application base and the way that different tasks cause overhead costs of resources in cloud systems. For large number of simple tasks this increases the cost and the cost is decreased if we have small number of complex tasks. Activity-based costing measures both the cost of the resources and the computation performance. The cost of every individual resource is different. Tasks are sorted according to their priority, and they are placed in three different lists based

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 11, May 2013

on three levels of priority namely high priority, medium priority and low priority. For computation of tasks, the system can take from high priority list first, then medium and then low. In this paper, the proposed algorithm is applied to group the above lists in order to allocate the task-groups to different available resources. The Improved Activity Based Costing method selects a set of resources to be used for computing. It groups tasks according to the processing capability of resources available. The coarse-grained tasks are processed in the selected resources, so that the Computation-Communication ratio is reduced. The experimental results using a simulator show that the time taken to complete tasks after grouping the tasks is very less when compared with time taken to complete the tasks without grouping the tasks.

e. A Three-Phases Scheduling in a Hierarchical Cloud Computing Network[8]

Cloud computing, uses low-power hosts to achieve high usability. It employs distributed resources to perform a function in a decentralized manner. It selects nodes on the network to facilitate the execution of complicated tasks that require large-scale computation. In this paper, a three-phases scheduling in a hierarchical cloud computing network [8] is proposed. The proposed scheduling can utilize better executing efficiency and maintain the load balancing of Cloud computing system. In this paper, a three-level hierarchical topology is adopted. The first level is the request manager that is used to assign the task to a suitable service manager. The second level is the service manager that is used to divide the task into some logical independent subtasks. The third level is the service node that is used to execute subtask. In this paper, the proposed algorithm carries out the task in three phases. In the first phase, the BTO (Best Task Order) scheduling determines the execution order for each task request. The most suitable scheduling order is obtained in accordance to the job demand and the service priority of each task by using the BTO scheduling. In the second phase, the EOLB (Enhanced Opportunistic Load Balancing) scheduling assigns a suitable service manager for allocation of the service node. EOLB combines a traditional OLB and the service manager threshold, which dispatches the task to the most suitable service manager according to the property of task requiring execution, while maintaining the advantage of traditional OLB to reach the load balance. In the third phase, the EMM (Enhanced Min-Min) scheduling guarantees that a suitable service node will be assigned to execute the task in the minimum execution time. EMM can choose the best service node and then use the threshold of the service node to guarantee that the node carries out the task in the shortest time. Thus, each task will be completed effectively and quickly in the hierarchical cloud computing network. A simulation is used for experiments. The simulation results prove that the task scheduling method combining EOLB with EMM is more effective than other scheduling approaches for reducing the completion time of a task. This scheduling enhances the execution performance of the system and makespan of all the tasks. It has better load balance of nodes.

f. A Community Cloud Oriented Workflow System Framework and its Scheduling Strategy[6]

Cloud computing application models can be divided into several categories: public cloud, private cloud, and community cloud. Community cloud model is proposed recently and has very different characteristics with the other two. Virtualization is a characteristic of Cloud computing, and has created a virtual society. Providing fast collaboration for many related organizations has always been an important research hotpot. As an application model of cloud computing, the Community Cloud Computing [6] is especially prominent in providing fast internet collaboration. In this paper, a novel workflow system framework for the Community Cloud which meets the need of the process-based fast collaboration well is proposed. A community cloud oriented task scheduling strategy called the Unified Scheduling Strategy for Instance-intensive Workflows (USS-I)[5] is proposed to ensure the fast collaboration mechanism to work with high efficiency. The workflow framework[5] contains many virtual domains, in which the community members related to the service are built when the corresponding collaboration service is created. In each community member, the system contains a number of service nodes and at least one management node. The service nodes are maintained and managed independently by each community member, and it’s the final executor of the collaboration tasks. The management node is set up for the task allocation and other management functions. Through this node the computing and storage resources can be connected to the upper layer of the system in order to participate in the execution of collaboration services. In each management node a Workflow Management and Scheduling Engine is deployed, accomplishing all the functions of the management node. The engine contains four main modules: Process Management Module, External Scheduling Module, Inter-member Scheduling Module, QoS Management Module. In this paper, a community cloud oriented task scheduling strategy is proposed for the workflow system framework, which is named the Unified Scheduling Strategy for Instance-intensive Workflows (USS-I) is proposed. E-government and e-business systems will come under instance-intensive workflow systems. A notable characteristic of instance-intensive workflow is the large amount of concurrent relatively simple workflow instances and the fierce competition of resources. The aim of using USS-I is to balance the workload of community members, and use the resources sufficiently, so that the heavy workload caused by large amount of concurrent workflow instances can be well reduced. There are three major steps in USS-I: Global QoS division algorithm QD (QoS Division): it works in the QoS management module of management nodes. In actual situation, users of workflow services are often used to set a global QoS constraint for the whole process, this algorithm provides a method to divide the global QoS constraint into local QoS constraints, so that the cloud resources can execute the tasks appropriately. Inter-member scheduling algorithm QCPCSA (QoS-Constrained Pre-Calculated Scheduling

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 11, May 2013

Algorithm): It is responsible for the allocation of tasks in community members. This algorithm concentrates mainly on the utilization rate of the system so that to reduce the waiting delay of tasks. External scheduling algorithm QCOAL (QoS-Constrained Opposite Average Load): It is responsible for receiving and transmitting tasks or process instances from users or other management nodes. This algorithm has a good resource flexibility, and can provide the system with real-time load balancing, so to maintain a relatively equal user experience and still a high system utilization rate. Thus, the proposed algorithm USS-I, a community cloud oriented task scheduling strategy and through the strategy the framework can support the fast collaboration mechanism with high efficiency.

g. Aggregated-DAG Scheduling for Job Flow Maximization in Heterogeneous Cloud Computing[14]

Cloud computing is a Heterogeneous computing platform in which resource management is fundamentally critical. Under the assumption of jobs comprised of subtasks forming DAG jobs, we focus on how to increase utilization and achieve near-optimal throughput performance on heterogeneous platforms. Aggregated–DAG scheduling[14] algorithm establishes that, by aggregating multiple jobs using *good* scheduling, a near optimal throughput can be achieved. Consequently, its limit is asymptotically converging to a certain value and can be written in the form of the service time of subtasks. In this paper, a simple super-job scheduling is proposed whose performance in terms of throughput is better than the well-known Heterogeneous Earliest-Finish-Time (HEFT) algorithm. Super-Job static scheduling algorithm is used for allocating the subtasks of N batch jobs to the available resources, so that we minimize the makespan. The total makespan or scheduling length of a super-job comprising of N jobs is defined as $TS * N$. We note that the static version of the problem (scheduling subtasks of a DAG with unit service requirements to a finite number of processors) is NP-hard. Our strategy is to decompose the main problem into multiple sub-problems through a divide-and-conquer strategy. This implies that we can aggregate these unrelated subtasks of the N jobs into a batch and efficiently allocate them to proper resources to minimize $TS * N$. The question is which subtasks we should select to aggregate in order to minimize the makespan. Indeed, every DAG graph can be divided into different depth-levels. For instance, let all entry nodes be in the first depth-level, its children be in the second depth-level, its grandchildren be in the third depth-level, and so forth. Since nodes are only dependent on the parent nodes, then, to maximize the degree of parallelism, it can be seen that nodes in the same depth level can be aggregated into a batch and scheduled at the same time. After all aggregated subtasks in each depth-level are scheduled; the descendent aggregated subtasks can be scheduled afterwards without a dependency concern, because all parent nodes of the current subtasks have finished execution. Hence, our first step is to categorize the subtasks from the N super-job into multiple depth-level and aggregate all subtasks in the same depth-level together into a list corresponding to its depth-level. After collecting subtasks from the same depth-level together, the makespan in every depth-level can be minimized.

h. Addressing Resource Management in Grids Through Network-Aware Meta-Scheduling In Advance[14]

In a real Grid, reservations may not be always feasible, for instance, there are resources, such as bandwidth, which lack a global management entity, making their reservation rather difficult. Thus, our work is based on meta-scheduling in advance rather than reservations in advance. This algorithm can be defined as the first step of the reservations in advance algorithm, where resources and time periods to execute the jobs are selected but making no physical reservation. This paper presents a module to perform rescheduling of tasks by sorting the jobs scheduled in a time interval by using its start time instead of its incoming time, as if they all were a Bag of Tasks (BoT). Thus, the allocation process has more information about jobs to be scheduled, and the rescheduling of those tasks reduces resource fragmentation, improving both the scheduled job rate and the resource usage. Furthermore, a technique is developed to decide when the rescheduling process is needed and over which resources and time intervals has to be applied. Meta-scheduling used in advance process behaves as follows [14]:

1) A user sends a request to his local administrative domain meta-scheduler. Every request must provide information on the application and the input QoS parameters, which are just specified by the start time and the deadline.

2) The meta-scheduler communicates with the Gap Management entity to obtain both the resource and the time interval for executing the job. The heuristics presented here take into account the predicted state of the resource (interconnection networks inclusive), the jobs that have already been scheduled and the QoS requirements of the job.

3) If it is not possible to fulfill the QoS requirements using the local resources, communication with meta-schedulers of other domains starts. To this end, techniques based on P2P systems can be used [15], [16].

4) If it is still not possible to fulfill the QoS requirements, a renegotiation process with the user is started to define new QoS requirements. This renegotiation, as well as the overall interaction with users, can be conducted by means of Service Level Agreements (SLA). This support for scheduling in advance is implemented on top of the GridWay meta-scheduler [17], called Scheduler in Advance Layer (SA-layer), as Figure 1 depicts. The usage of the resources is divided into time intervals, named slots. Then, the SA-layer has to schedule the future usage of resources by allocating the jobs into the resources taking one or more time slots. Due to the fact that duration of jobs into the different resources is unknown, algorithms for estimating job durations are needed. Predictions (both for computing and for network transfer times) are computed on the basis of job and resource features and data from previous executions, by applying an

exponential smoothing function. These predictions are only calculated when a suitable gap has been found.

III. JOB REPLANNING: BAG OF TASK RESCHEDULING

Using this model, there could be several reasons for rejecting jobs. One of them is the fragmentation (free parts of the resources are shattered in space and time) which could be solved by compacting the reservations in order to form a large free block. Another reason is due to unfavorable previous decisions (a previously scheduled job uses the only blocks which fit the new incoming job). These problems could be solved by changing the reservations of already admitted jobs. Thus, our system performs what we have named "Bag of Task Rescheduling (BoT-R)" from time to time. The BoT-R is regularly applied to reduce job allocation failures. With the aim of not generating network delays and to make it scalable, this process is made just at intra-domain level and at intervals. Hence, only the jobs whose executions are whithin the studied interval will take part into the re-planning process. Furthermore, there is not preemption of jobs. The BoT-R algorithm is separated into two steps, namely, estimating if there is any necessity of rescheduling tasks, and performing the BoT rescheduling for the selected resources, tasks and intervals. To perform the BoT rescheduling just when needed, resource fragmentation is measured since it is a forecast of how likely future allocations are to fail. To this end, several state variables are checked for each subinterval, such as the load of the system and the status of the Gap Management (number of gaps and its average size). Once the system has estimated which subintervals need task re-planning, the BoT re-planning process is triggered. First, a filtering process is applied over the resources to filter out the resources without usable fragmentation. Then, jobs belonging to those resources whose full execution is within the defined period are sorted by job start time restriction. Finally, for each resource selected to be defragmented, the sorted list of jobs is scanned in order with the aim of allocating as many jobs as possible in each resource. This way of mapping jobs to resources reduces the generated fragmentation, since, whenever possible, the jobs are allocated without leaving free time intervals in between allocations. When it is not possible to allocate any more jobs, the next resource is used for allocating the jobs which are not allocated yet, and so on.

IV. CONCLUSION

Scheduling parallel applications modeled by Directed Acyclic Graphs onto a network of heterogeneous computers is a NP-Complete problem. The efficiency of the cloud depends on the algorithms used for task scheduling. Different scheduling algorithms can be used depending on the type of the task to be scheduled. PISA: (priority impact scheduling algorithm) is based on user's priority. The differences between user's priorities may be based on the fee they paid. The workflow is composed of many tasks. When a workflow requests the cloud computing resource provider for services, the provider first queries the Access Strategy library. The users themselves should set the priority based on the fee they could pay for their workflow. The value of priority determines the highest level of cloud resource it can get. Balance-Reduce algorithm is a data locality driven task scheduling algorithm, which finds a good solution in time O(max{m+n, n log n}•m. Adaptive Resource Allocation for Pre-emptable Jobs in Cloud Systems algorithm adjusts the resource allocation adaptively based on the updated state of the actual task executions. The experimental results show that these algorithms work significantly in intense resource contention situation. The experimental results of Improved cost-based algorithm for task scheduling in Cloud computing show that the time taken to complete tasks after grouping the tasks is very less when compared with time taken to complete the tasks without grouping the tasks. The simulation results of A Three-Phases Scheduling in a Hierarchical Cloud Computing Network prove that the task scheduling method combining EOLB with EMM is more effective than other scheduling approaches for reducing the completion time of a task. This scheduling enhances the execution performance of the system and makespan of all the tasks. It has better load balance of nodes. A Community Cloud Oriented Workflow System Framework and its Scheduling Strategy can support the fast collaboration mechanism with high efficiency. Aggregated–DAG scheduling for job flow maximization in Heterogeneous Cloud Computing algorithm minimizes makespan, by aggregating multiple jobs using good scheduling, and a near optimal throughput can be achieved. Addressing Resource Management in Grids through Network-Aware Meta-Scheduling In Advance algorithm lets the system make rescheduling of tasks already scheduled in the same way as a BoT. To do that, the jobs are rescheduled by its start time instead of by its arrival time. Hence, the reallocation of those tasks will create fewer fragmentations into resources. Based on the above study it can be concluded that makespan can be reduced by grouping the tasks. Since cloud computing systems have a high degree of unpredictability with respect to resource availability in future as the cloud size increases, there is a need for better task scheduling algorithms. A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

REFERENCES

[1] Jiahui Jin, Junzhou Luo, Aibo Song, Fang Dong and Runqun Xiong, "BAR: An Efficient Data Locality Driven Task Scheduling Algorithm for Cloud Computing", 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

[2] Jiayin Li, Meikang Qiu, Jian-Wei Niu, Yu Chen, Zhong Ming "Adaptive Resource Allocation for Pre-emptable Jobs in Cloud Systems" 2010 IEEE

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 11, May 2013

[3] Ching-Hsien Hsu, Tai-Lung Chen "Adaptive Scheduling based on Quality of Service in Heterogeneous Environments" 2010 IEEE

[4] Mrs.S.Selvarani, Dr.G.Sudha Sadhasivam, "Improved Cost-Based Algorithm For Task Scheduling In Cloud Computing", 978-1-4244-5967-4/10/$26.00 ©2010 IEEE

[5] LI Wenhao "A Community Cloud Oriented Workflow System Framework and its Scheduling Strategy" 2010 IEEE

[6] A Marinos and G Briscoe. "Community Cloud Computing", First International Conference on Cloud Computing. ACM Press 2009.

[7] Gerasoulis, A. and Yang, T "A comparison of clustering heuristics for scheduling directed graphs multiprocessors" Journal of Parallel and Distributed Computing, 16(4):276-291.

[8] Shu-Ching, Wang Kuo-Qin, Yan*(Corresponding author) Shun-Sheng, Wang Ching-Wei, Chen, "A Three-Phases Scheduling in a Hierarchical Cloud Computing Network", 2011 Third International Conference on Communications and Mobile Computing, 978-0-7695-4357-4/11 © 2011 IEEE DOI 10.1109/CMC.2011.28

[9] "Building Grep The Web in the Cloud, Part1: Cloud Architectures". Developer.amazonwebservices.com. Retrieved 2010-08-22.

[10] K. Keahey and T. Freeman, "Science Clouds: Early Experiences in Cloud Computing for Scientific Applications," in proceedings of Cloud Computing and Its Applications 2008, Chicago, IL. 2008.

[11] "What is Cloud Computing?," http://www.zeus.comlcloud_computinglc1oud.html, January 2010.

[12] R.Buyya, C.S.Yeo, S.Venugopal. "Market-Oriented Cloud Computing: Vison, Hype, and Reality for Delivering IT Services as Computing Utilites," The 10th IEEE International Conference on High Performance Computing and Communications.

[13] Hu Wu, Zhuo Tang, Renfa Li, "A Priority constrained scheduling strategy of Multiple Workflows for Cloud Computing", ISBN 978-89-5519-163-9 ICACT 2012

[14] Boonyarith Saovapakhiran, George Michailidis, Michael Devetsikiotis, "Aggregated-DAG Scheduling for Job Flow Maximization in Heterogeneous Cloud Computing", IEEE Globecom 2011 proceedings L. Tomas and et al., "Network-aware meta-scheduling in advance with autonomous self-tuning system," Future Generation Computer Systems, vol. 27, no. 5, pp. 486 – 497, 2011.

[15] A. Caminero and et al., "Network-aware heuristics for inter-domain met scheduling in Grids," Journal of Computing and System Sciences, vol. 77, no. 2, pp. 262 – 281, 2011.

[16] A. D. Stefano and et al., "A P2P strategy for QoS discovery and SLA negotiation in Grid environment," Future Generation Computer Systems, vol. 25, no. 8, pp. 862 – 875, 2009.

[17] E. Huedo and et al., "A modular meta-scheduling architecture for interfacing with pre-WS and WS Grid resource management services," Future Generation Computing Systems, vol. 23, no. 2, pp. 252–261, 2007.

AUTHOR'S PROFILE

Mrs. Vijayalakshmi A. Lepakshi is currently Lecturer, Department of Computer Science, at Maharani Lakshmi Ammanni College for Women, Malleswaram, and Bangalore, India. She obtained her MSc in Computer Science from Sri Krishnadevaraya University, Anantapur, A.P, and M.Phil in Computer Science from Global Open University, Nagaland.

Dr. Prashanth C.S.R is currently the Professor and Head of the Department of Computer Science and Engineering, New Horizon College of Engineering, Bangalore-India. He obtained his B.E in Computer Science from Bangalore University, M.S in Computer Science from the University of Texas at Dallas, and Ph.D in Computer Science from Auburn University, USA. Dr. Prashanth has published extensively in the areas of High Performance Computing and Cloud Computing. He is also on the advisory board of several reputed international conferences and journals.