Cloud Resource Allocation as Preemptive Scheduling Approach

Suhas Yuvraj Badgujar
PG Student (CE)
SKN-SITS, Lonavala

Anand Bone
PG Guide (CE)
SKN-SITS, Lonavala

ABSTRACT
The increased degree of connectivity and the increased amount of data has led many providers to provide cloud services. Infrastructure as a Service (IaaS) is one of the Cloud Services it provides greater potential for a highly scalability of computing resources for demand in various applications like Parallel Data processing. The resources offered in the cloud are extremely dynamic and probably heterogeneous due to this dynamic load balancing, access balancing and scheduling of job is required. To achieve this many scheme are proposed, Nephele is one of the data processing framework which exploits the dynamic resource allocation offered by IaaS clouds for both task scheduling and execution. Specific tasks of processing a job can be allotted to different types of virtual machines which are automatically instantiated and terminated during the job execution. However the current algorithms are homogeneous and they do not consider the resource overload or underutilization during the job execution this increase task completion time. This paper introduces a new Approach for increasing the efficiency of the scheduling algorithm for the real time Cloud Computing services. Proposed method utilizes the Turnaround time Utility efficiently by discerning it into a gain function and a loss function for a single task based on their priorities. Algorithm has been executed on both preemptive and Non-preemptive methods. The experimental results show that it overtakes the existing utility based scheduling algorithms and also compare its performance with both preemptive and Non-preemptive scheduling approaches. Hence, Turnaround time utility scheduling approach which focuses on both high and the low priority jobs that arrives for scheduling is proposed.

General Terms
Task scheduling, Resource utilization, Cloud Computing, make-span, slope index

Keywords
Gain utility, Critical Point

1. INTRODUCTION
Today a increasing number of companies have to process data in a cost efficient way. Cloud computing is the use of computing resources such as hardware, software as well as service that are delivered as a service over a network(typically the Internet). Cloud computing is deliver services on user's data, software and computation. Cloud computing allows companies to keep away from infrastructure costs and focus on projects that separate their work instead of infrastructure for large amount of data.

Cloud computing is a alternative for distributed computing over a network which means that it has ability to run a program on many connected computers at the same time but cloud computing served up by virtual hardware, simulated by software system running on one or a number of real machines. Such virtual servers are physically does not exist. These virtual servers serve the incoming requests and perform action according to user's request. Cloud computing is having following types: 1) Paas -Platform as a Service 2) Iaas - Infrastructure as a Service, 3) Saas - Software as a Service.

Cloud computing has a promising approach to rent a large infrastructure on pay per usage basis. This is known as IaaS cloud. IaaS cloud's feature is the providing computing resources on demand for customer. Customer request for resources, provider allocates access and provides control a set of Virtual Machines which run at cloud provider.

Dynamic resource allocation is in implementation so these frameworks are processed on cluster environments. These are designed for Virtual machines are allocated at computing the job. Because of nature of framework is static the resources and computing environment cannot change until execution of program. Execution task may vary their requirement during the processing. So as a impact of static resource allocation may be insufficient in terms of processing job, which may decrease performance of task and increase the cost.

One of an IaaS cloud's key feature is the provisioning of compute resources on demand. The computer resources available in the cloud are highly dynamic and possibly heterogeneous. Nephele is the first data processing framework to explicitly exploit the dynamic resource allocation offered by today's IaaS clouds for both task scheduling and execution. Particular tasks of a processing a job can be assigned to different types of virtual machines which are automatically instantiated and terminated during the job execution.

To improve the performance of cloud computing, one approach is to employ the traditional Utility Accrual (UA) approach first proposed to associate each task with a Time Utility Function (TUF), which indicates the task's importance. Specifically, the TUF describes the value or utility accrued by a system at the time when a task is completed to improve the performance of cloud computing, it is important to not only measure the profit when completing a job in time, but also account for the penalty when a job is aborted or discarded. Note that, before a task is aborted or discarded, it consumes system sources including network bandwidth, storage space and processing power and thus can directly or indirectly affect the system performance.

However Nephele does not consider resource overload or underutilization during the job execution automatically. In this study, a novel Turnaround time utility algorithm is proposed for scheduling the real time cloud computing services. The most unique characteristics of this approach is that, different from traditional utility accrual approach that works under one single Time Utility Function (TUF), which have two different functions called a Gain and a loss Functions associated with

each task at the same time, to model the real-time applications for cloud computing. To compare the performance of cloud computing, the traditional Utility approach is deployed in both Non-Preemptive and Preemptive scheduling.

2. LITERATURE

In modern techniques systems are classified as high throughput computing (HTC) [1] or many task computing (MTC).Programming models share some similar objectives. Generally all programs is in execution are written consecutively in cloud area. The process framework takes caution of program from assigned nodes and executes every program on the execution instance. This framework is execute the job by allocating resources which are ignores the underutilization and overutilization during the processing of job.

The Pegasus framework by Deelman [2] has been designed for mapping complex workflows onto distributed resources such as the Grid. Pegasus which stands for Planning for Execution in grids. Jobs in this framework is represented as DAG (directed acyclic graph) with vertices representing the tasks to be processed and edges representing the dependencies between them. The created workflows remain hidden until Pegasus creates the mapping between the given tasks and the computed resources available at processing the job. It deals with DAGMan and Condor-G as its execution engine [3]. As a result, different jobs can only exchange data via files.

Thao [4] introduced the Swift system to reduce the management issues which occur when a job involving numerous tasks has to be executed on a large, possibly unstructured set of data. Swift mainly focuses on scientific applications that process heterogeneous data formats with applications and can manages schedule of computations in a location independent way.

Isard [5] proposed Dryad, which is designed to scale from powerful multi-core single computers through small clusters of computers. A data center with thousands of computers operates for processing of unstructured and heterogeneous data. Current data processing frameworks like Google's MapReduce or Microsoft's Dryad engine have been designed for cluster environments. This is reflected in a number of assumptions which are not necessarily valid in cloud environments.

IoanRaicu [6] proposed Falkon, which is a Fast and Lightweight task execution framework and it is designed to enable the efficient execution of many small jobs. Dornemann [7] presented an approach to handle peak situations of load in BPEL workflows using Amazon EC2. Kao, presented research project Nephele. Nephele is the data processing framework to explicitly exploit the dynamic resource allocation for both task scheduling and execution.

To improve the performance of cloud computing, one approach is to employ the traditional Utility Accrual (UA) approach first proposed to associate each task with a Time Utility Function (TUF), which indicates the task's importance. Specifically, the TUF describes the value or utility accrued by a system at the time when a task is completed.

While Jensen's definition of TUF allows the semantics of soft time constraints to be more precisely specified [8], all these variations of UA-aware scheduling algorithms imply that utility is accrued only when a task is successfully completed and the aborted tasks neither increase nor decrease the accrued value or utility of the system.

Yu proposed a task model that considers both the profit and penalty that a system may incur when executing a task [9]. According to this model, a task is associated with two different TUFs, a profit TUF and a penalty TUF. The system takes a profit (determined by its profit TUF) if the task completes by its deadline and suffers a penalty (determined by its penalty TUF), if it misses its deadline or is dropped before its deadline. It is tempting to use negative values for the penalties and thus combine both TUFs into one single TUF. However, a task can be completed or aborted and hence can produce either a profit value or a penalty value.

3. IMPLEMENTATION

Task scheduling and load-balancing technique: A task is a (sequential) activity that uses a set of inputs to produce a set of outputs. Processes in fixed set are statically assigned to processors, either at compile-time or at start-up (i.e., partitioning) avoids overhead of load balancing using these load-balancing algorithms. The Grid computing algorithms can be broadly categorized as centralized or decentralized, dynamic or static or the hybrid policies in latest trend. A centralized load balancing approach can support larger system. Hadoop system takes the centralized scheduler architecture. In static load balancing, all information is known in advance and tasks are allocated according to the prior knowledge and will not be affected by the state of the system. 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 become overloaded [10].

In cloud computing, each application of users will run on virtual operating systems, the cloud systems distributed resources among these virtual systems. Every application is completely different and is independent and has no link between each other whatsoever, For example, some require more CPU time to compute complex task and some others may need more memory to store data. Resources are sacrificed on activities performed on each individual unit of service.

In order to measure direct costs of applications, every individual use of resources (like CPU cost, memory cost, I/O cost) must be measured. When the direct data of each individual resources cost has been measured, more accurate cost and profit analysis.

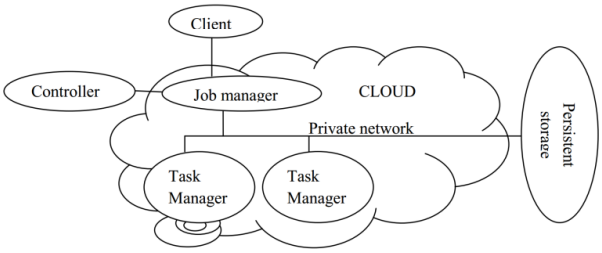

Fig 1: Nephele Architecture

Nephele architecture: Nephele [11] is a new data processing framework for cloud environment that takes up many ideas of previous processing frameworks but refines them to better match the dynamic and opaque nature of a cloud. Nephele's architecture follows a classic master-worker pattern as illustrated in Fig. 1.

Before submitting a Nephele compute job, a user must start a VM in the cloud which runs the so called Job Manager (JM).

The Job Manager which receives the client's jobs is responsible for scheduling them and coordinates their execution. It is capable of communicating with the interface the cloud operator provides to control the instantiation of VMs. It call this interface the Cloud Controller. By means of the Cloud Controller the Job Manager can allocate or deallocate VMs according to the current job execution phase. It will comply with common Cloud computing terminology and refer to these VMs as instances for the remainder of this study. The term instance type will be used to differentiate between VMs with different hardware characteristics. The actual execution of tasks which a Nephele job consists of is carried out by a set of instances. Each instance runs a so-called Task Manager (TM). A Task Manager receives one or more tasks from the Job Manager at a time, executes them and after that informs the Job Manager about their completion or possible errors. Unless a job is submitted to the Job Manager, It expect the set of instances (and hence the set of Task Managers) to be empty. Upon job reception the Job Manager then decides, depending on the job's particular tasks, how many and what type of instances the job should be executed on and when the respective instances must be allocated/deallocated to ensure a continuous but cost-efficient processing. The newly allocated instances boot up with a previously compiled VM image. The image is configured to automatically start a Task Manager and register it with the Job Manager. Once all the necessary Task Managers have successfully contacted the Job Manager, it triggers the execution of the scheduled job. Initially, the VM images used to boot up the Task Managers are blank and do not contain any of the data the Nephele job is supposed to operate on.

The expected gain utility and the critical point

Since the task execution time is not known deterministically, It do not know if executing the task will lead to positive gain or loss. To solve this problem, It can employ a metric, i.e., the expected gain utility, to help us make the decision. Given a task T with arrival time of ati, let its predicted starting time be ti. Then the potential Gain Pi (ti) to execute T can be represented as the integration of the summation of gain over time ti and the difference of the starting time of the process and the arrival time of the process:

$$Pi(ti) = max \int_{max}^{Ci-(ti-ati)} Pi\big(t + (ti - ati)\big) fi(t) dt$$

Similarly, the potential loss (*Li (T)*) to execute Ti can be represented as:

$$Li(T) = Li(D) \int_{Ci-(ti-ati)}^{max} fi(t) dt$$

Therefore, the expected increased efficiency η (T) to execute Ti can be represented as:

$$\eta(T) = Pi(ti) - Li(ti)$$

A task can be accepted or chosen for execution when η(T)>0, which means that the probability of to obtain positive gain is no smaller than that to incur a loss. It can further limit the task acceptance by imposing a threshold (δ) to the expected accrued utility, i.e. a task is accepted or can be chosen for execution if: *Pi (T)* $\geq \mu$

Furthermore, since the task execution time is not known a prior, it needs to decide whether to continue or abort the execution of a task. The longer it executes the task, the closer to the completion point of the task. At the same time, however, the longer the task executes the higher penalty the system has to endure if the task cannot meet its deadline. To determine the appropriate time to abort a task, It employ another metric, i.e., the critical point. Let task Ti starts its execution at t1, then the potential profit Ti >t (i.e., η(T)) can be represented as the integration of the maximum gain the difference of the completion of the task. The Potential loss can be calculated by the integration of its completion time to the max time. Hence, the expected efficiency η is the difference believes the gain of a task and the loss of a task. If It substitute η to be equal to 0, It can see that the gains & loss are found to be equal in executing a task. As time increases, the η decrease and after a critical point at deadline more loss incurs then gains.

Preemptive scheduling: The Preemptive scheduling algorithm belongs to a new family of real-time service oriented scheduling problems. As the complementarily of previous non-preemptive algorithm, real time tasks are scheduled preemptively with the objective of maximizing the total utility time. The preemptive scheduling heuristics is to judiciously accept, schedule and cancel real-time services when necessary to maximize the efficiency. The new scheduling algorithm has much better performance than an earlier scheduling approach based on a similar model does.

Algorithm for Preemptive scheduling:

1. Input: Let {T1, T2,...,Tk } be the accepted tasks in the ready queue and let ei be the expected execution time of Ti. Let current time be t and let T0 be the task currently being executed. Let the expected utility density threshold be μ.

2. if a new task, i.e. T_p arrives then

3. Check if T_p should preempt the current task or not;

4. if Preemption allowed then

5. T_p preempts the current task and starts being executed;

6. End if

7. If Preemption not allowed then

8. Accept T_p if $\frac{U_p(e_0)}{e_0} \geq \mu_e$

9. Reject T_p if $\frac{U_p(e_0)}{e_0} \geq \frac{\mu e_0}{e_0}$

10. End if

11. Remove Tj in the ready queue if $\frac{U_p(e_0)}{e_j} \geq \mu$

12. End if

13. If at preemption check point then

14. PREEMPTION CHECKING;

15. End if

16. If T_0 is completed then

17. Choose the highest expected utility density task T_i to run.

18. Remove Tj in the ready queue if $\frac{U_j(e_i)}{e_j} \geq \mu$

19. End if

20. If t = the critical time of p_0 then

21. Abort p_0 immediately

22. Choose the highest expected utility density task p_i to run.

23. Remove p_j in the ready queue

24. End if

The details of scheduling are described in above algorithm. There are five main parts in the scheduling. They are the preemption checking, feasibility checking, task selecting, scheduling point checking and critical point checking. When new tasks are added in to ready queue, no matter whether there is preemption or not, the feasibility checking will work to check if the new ready queue is feasible or not. If any task cannot meet the requirement, it will be removed from the ready queue. Scheduling point checking makes sure all the left tasks in the expected accrued utility density task to run when the server is idle. The critical point checking will always monitor the current running task's state to prevent the server wasting time on the non-profitable running task. The preemption checking works when there is a prosperous task wants to preempt the current task. The combination of these parts guarantees to judiciously schedule the tasks for achieving high accumulated total utilities. It is worthy to talk more about the preemption checking part in details, because improper aggressive preemption will worsen the scheduling performance. From Algorithm It can see that if a task can be finished successfully before its deadline even in its worst case, the scheduling will protect the current running task from being preempted by any other tasks. Otherwise, if a prosperous task has an expected accrued utility density which is larger than the current running task's conditional expected utility density by at least a value equals to the pre-set preemption threshold, the preemption is permitted.

Algorithm for Verification of Preemptive method

1. Input: Let T_0be the task currently being executed and T_p be the task wants to preempt T_0, current time t, U (T_0, t) be the conditional expected utility density ofT_0at time t, $e_0$is the remaining expected time of T_0. Up (t) is the expected utility density of T_p;

2. If the expected density is greater, then

3. Check what T0's worst case finish time is;

4. If T_0can be finished before its deadline even in the worst case then

5. Preemption is not allowed;

6. End if

7. If T_0's worst case will miss as its deadline then

8. Preemption allowed;

9. End if

10. End if

The feasibility check is one more part deserves detail description. In this part, scheduling simulates the real execution sequence for the left tasks in readyqueue and check following this sequence, if all of them can satisfy the requirement or not. The thing needs to be discussed is how to determine the sequence of the left tasks. From equation (1), (2) and (3), It can clearly see that the expected utility of running a task depends heavily on variable T, i.e., the time when the task can start. If It know the execution order and thus the expected starting time for tasks in the ready queue, It will be able to quantify the expected utility density of each task more accurately. In algorithm.5, It show utility metric based on a speculated execution order of the tasks in the ready queue. The general idea to generate the speculated execution order is as follows. It first calculates the expected utility density for each task in the readyqueue based on the expected finishing time to the current running task. Then the task with the largest one is assumed to be the first task that will be executed after the current task is finished. Based on this assumption, It then calculate the expected utilities for the rest of the tasks in the ready queue and select the next task. This process continues until all tasks in the readyqueue are put in order. When completed, It essentially generate a speculated execution order for the tasks in the ready queue and, at the same time, calculate the corresponding expected utility density for each task.

4. ACKNOWLEDGMENTS

I am grateful to Prof. V. D. Thombre, HOD, Department of Computer Engineering, SKN-SITS, Lonavala, for his constant motivation, encouragement and guidance. I would also like to thank my guide, Prof. Anand,Bone for his constant support and guidance. Special thanks to the experts who have contributed towards development of this paper work.

5. CONCLUSION

In structure of Nephele scheduler, presented preemptive scheduling as new approach to Nephele framework. The real-time service system should be compatible with preemption in respect that it is necessary for nowadays' service requests. In this approach automatically schedules flow steps of tasks to underutilize and over utilized nodes using Cloud computing infrastructures in resource allocation and preemptive scheduling algorithm is effective in this regard. Also It present a Turnaround time utility scheduling approach which focuses on scheduling. The speed can be improved in the proposed system from the existing Nephele framework. By using virtual machines, It can improve the overall resource utilization also reduce the processing cost.

Data dependency, failure handling and recovery approach that takes advantage of the presented solution to dynamically provide resources is another interesting area of further research.

6. REFERENCES

[1] I. Raicu, I. Foster, and Y. Zhao, "Many-Task Computing for Grids and Supercomputers," Proc. Workshop Many-Task Computing on Grids and Supercomputers, pp. 1-11, Nov. 2008.

[2] E. Deelman, G. Singh, M. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. Jacob, and D. Katz, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," Scientific Programming Journal, vol.13(3), pp.219–237, 2005.

[3] J. Frey, T. Tannenbaum, M. Livny, I. Foster and S. Tuecke, "Condor-G: a computation management agent for multi-institutional grids," journal of Cluster Computing, vol.5 (3), pp.237–246, 2002.

[4] T. White, Hadoop: The Definitive Guide. O'Reilly Media, 2009

[5] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data parallel programs from sequential building blocks," in proceedings of the second

ACM SIGOPS/EuroSys European Conference on Computer Systems, New York, USA, pp. 59–72, 2007.

[6] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde, "Falkon: a fast and light weight task execution framework," *proceedings of the ACM/IEEE conference on Supercomputing*, New York, USA, pp.1–12, 2007.

[7] T. Dornemann, E. Juhnke, and B. Freisleben. "On-Demand Resource Provisioning for BPEL Workflows Using Amazon‟s Elastic Compute Cloud." In CCGRID ‟09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 140–147, Washington, DC,USA,2009.IEEE Computer Society.

[8] Li, P, H. Wu, B. Ravindran and E. D. Jensen, April 2006. A utility accrual scheduling algorithm for real-time activities with mutual exclusion resource constraints. IEEE Trans. Comput., 55: 454-469.

[9] Yu, Y., S. Ren, N. Chen and X. Wang, 2010. Profit and penalty aware (PP-aware) scheduling for tasks with variable task execution time. Proceedings of the 2010 ACM Symposium on Applied Computing, Mar. 22-26, ACM, Sierre, Switzerland, pp: 334-339.

[10] Zaharia, M., D. Borthakur, J.S. Sarma, K. Elmeleegy and S. Shenker et al., 2009. Job scheduling for multi-user mapreduce clusters. EECS Department, University of California, Berkeley.

[11] D. Warneke and O. Kao, "Exploiting dynamic resource allocation for efficient parallel data processing in the cloud," IEEE transactions on parallel and distributed systems, vol. 22, no. 6, June 2011