ISSN:0975-9646

Resource Allocation Method using Scheduling methods for Parallel Data Processing in Cloud

Sowmya Koneru[1], V N Rajesh Uddandi[1], Satheesh Kavuri[2]

[1]*Depatment of Computer Science & Engineering,*
Sri Sunflower College of Engineering & Technology,
Lankapally, Krishna Dist,
[2]*Depatment of Computer Science & Engineering,*
Dhanekula Institute of Engineering & Technology,
Vijayawada, Krishna Dist,

Abstract- Infrastructure as a Service (IaaS) clouds have emerged as a promising new platform for massively parallel data processing. By eliminating the need for large upfront capital expenses, operators of IaaS clouds offer their customers the unprecedented possibility to acquire access to a highly scalable pool of computing resources on a short-term basis and enable them to execute data analysis applications at a scale which has been traditionally reserved to large Internet companies and research facilities. However, despite the growing popularity of these kinds of distributed applications, the current parallel data processing frameworks, which support the creation and execution of large-scale data analysis jobs, still stem from the era of dedicated, static compute clusters and have disregarded the particular characteristics of IaaS platforms so far. 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 processing a job can be assigned to different types of virtual machines which are automatically instantiated and terminated during the job execution. However, the current algorithms does not consider the resource overload or underutilization during the job execution. In this paper, we have focused on increasing the efficacy of the scheduling algorithm for the real time Cloud Computing services. Our Algorithm utilizes the Turnaround time Utility efficiently by differentiating it into a gain function and a loss function for a single task. The algorithm also assigns high priority for task of early completion and less priority for abortions issues of real time tasks. The algorithm has been implemented on RR method. The out performs existing utility based scheduling algorithms and also compare its performance.

Keywords: Cloud computing, task scheduling, resource utilization, High-Throughput Computing.

I. INTRODUCTION

Cloud computing refers to the idea of delivering dynamically-scalable IT resources like computing power, storage or higher level platforms and services on demand to external customers over the Internet. The resources are rapidly provisioned and released with minimal management effort or service provider interaction, allowing the cloud customer to quickly grow or shrink the rented infrastructure according to his requirements. The Cloud computing has the potential to dramatically change the landscape of the current IT industry (Armbrust *et al*., 2009; Goldberg, 1989) For companies that only have to process large amounts of data occasionally running their own data center is obviously not an option. Instead, Cloud computing has emerged as a promising approach to rent a large IT infrastructure on a short-term pay-per-usage basis. Operators of so-called IaaS clouds, let their customers allocate, access and control a set of Virtual Machines (VMs) which run inside their data centers and only charge them for the period of time the machines are allocated. The VMs are typically offered in different types, each type with its own characteristics (number of CPU cores, amount of main memory) and cost. Since the VM abstraction of IaaS clouds fits the architectural paradigm assumed by the data processing frameworks described above, projects like Hadoop The Apache Software Foundation, 2011 (White, 2010), a popular open source implementation of Google's MapReduce framework, already have begun to promote using their frameworks in the cloud (White, 2010) Only recently, Amazon has integrated Hadoop as one of its core infrastructure services . However, instead of embracing its dynamic resource allocation, current data processing frameworks rather expect the cloud to imitate the static nature of the cluster environments (Dornemann *et al*., 2009) they were originally designed for, e.g., at the moment the types and number of VMs allocated at the beginning of a compute job cannot be changed in the course of processing, although the tasks the job consists of completely different demands on the environment. As a result, rented resources may be inadequate for big parts of the processing job, which may lower the overall processing performance 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. 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 job can be assigned to different types of virtual machines which are automatically instantiated and terminated during the job execution. While there exist different interpretations and views on cloud computing (Armbrust *et al*., 2009) it is less disputable that being able to effectively exploit the computing resources in the clouds to provide computing service at different quality levels is essential to the success of cloud computing. For real-time applications and services, the

timeliness is a major criterion in judging the quality of service. Due to the nature of real- time applications over the Internet, the timeliness here refers to more than the deadline guarantee as that for hard real-time systems. In this regard, an important performance metric for cloud computing can thus be the sum of certain value or utility that is accrued by processing all real-time service requests. 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 (Li *et al*., 2006). Based on this model, there have been extensive research results published on the topic of UA scheduling. While Jensen's definition of TUF allows the semantics of soft time constraints to be more precisely specified, 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. We believe that, 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. This is especially true for cloud computing in considering the large possibility of migration of a task within the clouds for reasons such as the economy considerations (Casati and Shan, 2001). If a job is deemed to miss its deadline with no positive semantic gain, a better choice should be one that can detect it and discard it as soon as possible. Recently, (Yu *et al*., 2010) proposed a task model that considers both the profit and penalty that a system may incur when executing a task. 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. Mathematically, if there exists such a single function, it would imply that a single value in its domain was mapped to two values in its range, violating that it is a function. Therefore, one utility function cannot accurately represent both the profit and penalty information when executing a task. There are also some other penalty related models proposed in the literature. For example, studied the on-line scheduling problem when penalties have to be paid for rejected jobs. This model, however, does not account for the penalty to drop the task before its deadline. 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 This study includes further details on scheduling strategies and extended experimental results. The study is structured as follows: First it starts with describing the basic concept of cloud and present the architecture of the Nephele and outline how jobs can be described and executed in the cloud. Followed by our scheduling approach in explained in detail. Then we present the experiment setup used for the evaluation and discuss the results. Finally, we conclude the study.

II. SCHEDULING AND LOAD-BALANCING

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 (Zaharia *et al*., 2009). In cloud computing, each applications of users will run on a 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.

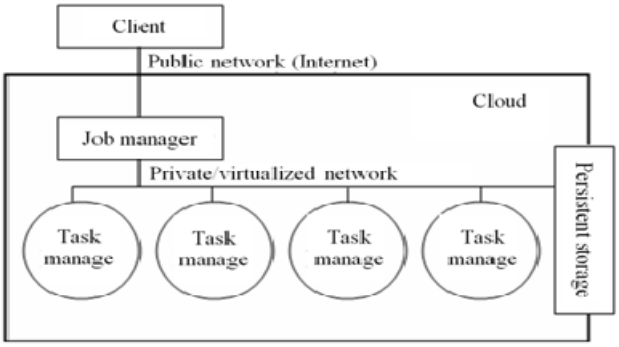

Fig. Nephele's Architecture

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.

Overview of Nephele architecture: Nephele is a new data processing framework (Warneke and Kao, 2009; Ravindran *et al*., 2005) 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. 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. We call this interface the Cloud Controller. By means of the Cloud Controller the Job Manager can allocate or De-allocate VMs according to the current job execution phase. We 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. For example, the instance type "m1.small" could denote VMs with one CPU core, one GB of RAM and a 128 GB disk while the instance type "c1.xlarge" could refer to machines with 8 CPU cores, 18 GB RAM and a 512 GB disk. 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, we 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/de-allocated 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. As a result, we expect the cloud to offer persistent storage (like, e.g., Amazon S3 (Amazon Web Services)). This persistent storage is supposed to store the job's input data and eventually receive its output data. It must be accessible for both the Job Manager as well as for the set of Task Managers, even if they are connected by a private or virtual network.

RR Scheduling: The RR scheduling method which is used to maximize the efficiency gain. Since the execution of a task may gain positive profit or suffer penalty and thus degrade the overall computing performance, judicious decisions must be made with regard to executing a task, dropping or aborting a task and when to drop or abort a task. The rationale of our approach is very intuitive, i.e. a task can be accepted and executed only when it is statistically promising to bring positive gain and discarded or aborted otherwise. Before we introduce the details of our scheduling approach, we first introduce two useful concepts, the expected gain utility and the critical point.

Algorithm For RR scheduling:

Consider K accepted Task in Ready Queue and the Current Time t.

Parameters

1: Accepted Task in the Queue Level. Let { t1 , t2 , ..., tk}Ar be the Arrival Time AT [T= 1 to K]

2: Let Currently Running Task may be at T=0. Show the task with T and the Threshold Value Th AT = A0.

3: Conditions The Current Job is in Critical, Then Abort the execution of T0

4:Otherwise New Task enrolled in the end process.

5: Calculation of efficiency of task and reschedule the task based on the Utility value and load into the ready Queue.

6: Start the Execution from T1. The utility value is less then the Threshold value then remove the process from ready queue else the current process and start its execution

The scheduling algorithm: Our scheduling algorithm works at scheduling points that include: the arrival of a new task, the completion of the current task and the critical point of the current task.

III. RELATED WORK

Dynamically switching between (constrained) local and (plentiful) remote resources, often referred as cyber-foraging, has shed light on many research work [3], [9]. These approaches augment the capability of resource-constrained devices by offloading computing tasks to nearby computing resources, or surrogates. Nephele takes insights and inspirations from these previous systems, and shifts the focus from alleviating memory constraints and provides evaluation on hardware of the time, typically laptops, to more modern smartphones. Furthermore, it enhances computation performance by exploiting parallelism with multiple VM creation on elastic cloud resources and provides a convenient VM management framework for different QoS expectation [10]. The variation between the total utility and the experiment sets. From that we can know, how the values of total utility will increased to the corresponding values of experiment sets. And this graph shows that, preemptive results are having higher total utility than the RR and execution graph of the Nephele.

IV. CONCLUSION

In this paper we discussed the challenges and opportunities for efficient parallel data processing in cloud environments and presented the Internet has grown enormously, which has presented a great opportunity for providing real-time services

over the Internet. We have discussed the challenges and opportunities for efficient parallel data processing (Chaiken *et al*., 2008) in cloud environments and presented Nephele, the first data processing framework to exploit the dynamic resource provisioning offered by today's IaaS clouds. We have described Nephele's basic architecture and presented a performance comparison to the wellestablished data processing framework Hadoop. The performance evaluation gives a first impression on how the ability to assign specific virtual machine types to specific tasks of a processing job, as well as the possibility to automatically allocate/deallocate virtual machines in the course of a job execution, can help to improve the overall resource utilization and, consequently, reduce the processing cost. The on-line real-time service system should be compatible with preemption in respect that it is necessary and befitting for nowadays' service requests. Our experimental results clearly show that our proposed RR scheduling algorithm is effective in this regard. In this study, we present a novel Turnaround time utility scheduling approach which focuses on both the high priority and the low priority takes that arrive for scheduling. This study can be viewed as the extended version of Nephele (Warneke and Kao, 2011). It is also a significant improvement compared to RR scheduling (Liu *et al*., 2010) in which, the RR approaches better than the preemptive counterpart. Our extensive experimental results clearly show that our proposed RR method can outperform the preemptive approach.

REFERENCES

[1] 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.

[2] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. Intl. Journal of Supercomputer Applications, 11(2):115–128, 1997.

[3] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor- G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing, 5(3):237–246, 2002.

[4] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. In EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pages 59–72, New York, NY, USA, 2007. ACM.

[5] A. Kivity. kvm: the Linux Virtual Machine Monitor. In OLS '07: The 2007 Ottawa Linux Symposium, pages 225–230, July 2007.

[6] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems. Technical report, University of California, Santa Barbara, 2008.

[7] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: A Not-So-Foreign Language for Data Processing. In SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pages 1099–1110, New York, NY, USA, 2008. ACM.

[8] O. O'Malley and A. C. Murthy. Winning a 60 Second Dash with a Yellow Elephant. Technical report, Yahoo!, 2009.

[9] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51:107–113, January 2008. [44] Ewa Deelman, Gurmeet Singh, Mei-Hui Su

[10] Jens Dittrich, Jorge-Arnulfo Quian´e-Ruiz, Alekh Jindal, Yagiz Kargin, Vinay Setty, and J¨org Schad. Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). *Proc. of the VLDB Endowment*, 3:515–529, September 2010.

[11] Tim Dornemann, Ernst Juhnke, and Bernd Freisleben. On-demand resource provisioning

for BPEL workflows using Amazon's Elastic Compute Cloud. In *Proc. Of the 9th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, CCGRID '09, pages 140–147, Washington, DC, USA, 2009. IEEE Computer Society.

[12] Alexei Drummond and Korbinian Strimmer. PAL: An object-oriented programming library for molecular evolution and phylogenetics. *Bioinformatics*, 17:662– 663, 2001.

[13] Kenneth J. Duda and David R. Cheriton. Borrowed-virtual-time (BVT) scheduling: Supporting latency-sensitive threads in a general-purpose scheduler. *SIGOPS Operating Systems Review*, 33:261–276, December 1999.

[14] Nick G. Duffield, Joseph Horowitz, Francesco Lo Presti, and Donald Towsley. Multicast topology inference from end-to-end measurements. *Advances in Performance Analysis*, 3(3):207–226, September 2000.

[15] Nick G. Duffield, Joseph Horowitz, Francesco Lo Presti, and Donald Towsley. Multicast topology inference from measured end-to-end loss. *IEEE Transactions on Information Theory*, 48(1):26 –45, January 2002.

[16] Nick G. Duffield, Francesco Lo Presti, Vern Paxson, and Donald Towsley. Network loss tomography using striped unicast probes. *IEEE/ACM Transactions on Networking*, 14:697–710, August 2006.

[17] Bin Fan, Wittawat Tantisiriroj, Lin Xiao, and Garth Gibson. DiskReduce: RAID for data-intensive scalable computing. In *Proc. of the 4th Annual Workshop on Petascale Data Storage*, PDSW '09, pages 6–10, New York, NY, USA, 2009. ACM.