Efficient Parallel Data Processing for Dynamic Resource Allocation in the Cloud

Anand Prabu P[1],Dhanasekar P[2] ,Sairamprabhu S G[3]
[1]*Department of Computer Science and Enginnering,RVS College of Engineering and Technolgy,Anna University.*
[3]*Department of Computer Science and Enginnering,RVS College of Engineering and Technolgy,Anna University.*
[1]a.prabu7@gmail.com
[3] sairambit@gmail.com
[2]*Department of Computer Science and Enginnering,RVS College of Engineering and Technolgy,Anna University.*
[2] write2sekar@gmail.com

***Abstract*—In recent years ad hoc parallel data processing is one of the emerging applications for Infrastructure-as-a-Service (IaaS) cloud environment. The current processing frameworks has been designed for homogenous cloud setup, which consequently leads to increased processing time and cost. In this paper we present Nephele, a first data processing framework for exploiting dynamic resource allocation in an cloud environment. 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. Based on this framework, we perform Map Reduce-inspired processing jobs on an IaaS cloud system and compare the results to the popular data processing framework Hadoop.**

I.INTRODUCTION

In recent times, number of companies have to process large amounts of data in a cost-efficient manner. Classic examples for these companies are Google, Yahoo, or Microsoft. The huge amount of data they work with has made traditional database solutions expensive . Instead, these companies has gone for large number of commodity servers. To simplify the development of distributed applications , many of these companies have also built data processing frameworks. Google's MapReduce , or Yahoo!'s Map-Reduce-Merge are examples of the above. They are classified into terms like high-throughput computing (HTC) or many-task computing (MTC), depending on the amount of data and the number of tasks involved in the computation . Though these two systems differ in design,but their programming models share similar objectives. The framework takes care of distributing the program among the nodes and executes each instance of the program on the appropriate fragment of data. Instead, Cloud computing has emerged as an great approach to rent a large IT infrastructure Amazon EC2 is an operator which allow their clients to access, allocate, and control a set of virtual machines (VMs) which run inside their data centers and charge them for the period of time the machines were used. The VMs are typically offered in different types, based on characteristics and cost. Since projects like Hadoop an exsisting open source implementation of Google's MapReduce framework, already promoted their frameworks in the cloud . However, instead of embracing its dynamic resource allocation, current data processing frameworks expect the cloud to produce the static nature of the cluster environments they were originally designed .As a result, rented resources may be not sufficient for processing a job, which may lower the overall performance and increase the cost. In this paper, we want to present Nephele, a new processing framework designed for cloud environments. Nephele is the first and best data processing framework to perform dynamic allocation/deallocation of resources from an cloud during job execution.This paper includes details on scheduling strategies and results. The paper is structured as follows: In Section 3, we present basic Nephele architecture and describe how jobs are executed in the cloud. Section 4 provides somedetails on Nephele's performance and

optimizations .Finally, paper is concluded by related work.

II.CHALLENGES

In this section, we briefly discuss the challenges in efficient data parallel processing.The major challenge is cloud opaqueness.Current data processing techniques attempt to schedule the computed nodes with knowledge about network and thus avoid bottlenecks.In cloud ,information about the topology is completely hidden from users,so this may create congestion in the network. So therefore the system must become aware of the cloud environment and the jobs executed.Also the paradigm used should be powerful to depict the dependencies.The system should be aware of when to allocate/deallocate the VM's. Finally, the scheduler of such a processing framework must be able to determine which task of a job should be executed on which type of VM. To ensure locality between tasks of a processing is to execute these job's task on the same VM . This may help in allocating fewer,but powerful VMs with multiple CPU cores.Scheduling the task in VM with multiple cores than single core machines ensures data locality.

III.DESIGN:

Based on the challenges,we design Nephele,first data processing framework for cloud.

A.ARCHITECTURE

Nephele's architecture follows a pattern as illustrated in Fig. 1.

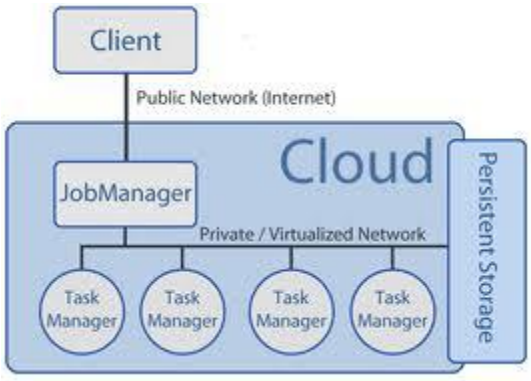

Fig 1.Nephele Architecture

Before submitting a Nephele's job, a VM must be started by the user in the cloud which runs the Job Manager (JM). The Job Manager receives the jobs, schedules them, and coordinates execution. It can communicate with the interface the cloud operator to provide the control to instantiate the VMs and this is termed as the Cloud Controller. Using the Cloud Controller the JM can allocate/deallocate VMs based on the current job execution. The term instance type is used to show difference between VMs with multiple differnt hardware characteristics.The execution of Nephele job tasks is carried out by a set of instances. Every instance runs a so-called Task Manager (TM). A Task Manager
receives the tasks from the Job Manager ,executes them, and t informs the Job Manager
about their completion or possible errors. Unless the Job Manager gets ajob, we assume the set of instances as empty. Upon receivable of job the Job Manager then decides, how many and what type of instances the job should be executed and when the corresponding instances must be allocated/deallocated to ensure a cost-efficient processing.

B. SCHEDULING STRATEGIES:

The Basic idea to refine the scheduling strategy
for recurring jobs is to use feedback data. We develop a system for Nephele which continuously monitor's running tasks and the instances. Based on the Java Management Extensions (JMX) the system is capable of breaking down its processing time that a task spends processing user code and the time it waits for data. With the collected data Nephele is able to detect computational and I/O bottlenecks.The computational bottlenecks suggests that higher degree of parallelization for the tasks, I/O bottlenecks provides hints to switch to faster channel types and reconsider the instance.Then Nephele generates a cryptographic signature for every task and recurring tasks can be identified and already recorded data can be exploited.Now, we use the profiling data to detect the bottlenecks and provide the user to choose annotations for the job A user can use the feedback to improve the job's annotations. In advanced versions of Nephele, the system can automatically adjust to detected bottlenecks between continuous executions of the same job or at job's execution at runtime. The allocation time of cloud instances is determined by the start times of the

subtasks, there are different strategies for deallocation. Nephele can track the instances' allocation times. An instance of a each type in the execution stage is not immediately deallocated if same instance type is required in an upcoming execution Stage.So, Nephele retains the instance allocated till the end of current lease period. If the preceding execution phase has begun before the end of the previous period, it is reassigned to an execution of precding stage, else it deallocates early enough not to cause any additional cost.

IV EVALUTION

In this section, we present the first performance results of Nephele and compare it to the Hadoop's data processing framework. We have chosen Hadoop , because it is an open source software and enjoys high popularity in the data processing.Hadoop has been designed
to run on a very large number of nodes (i.e., thousands of nodes) in current IaaS clouds**.**The general Map Reduce technique has been chosen to run on both the framework.In the preceding section A general Map Reduce on Hadoop and section B describes the general Map Reduce on Nephele framework.

A. MAPREDUCE AND HADOOP:

In order to execute the task with Hadoop, we created different MapReduce programs which were executed consecutively.The MapReduce job reads the input data set, counts number of occurences of each and writes them back to Hadoop's HDFS file system. Since the MapReduce engine is internally designed to count the incoming data words between the map and the reduce phase. Instead, we simply used the word count code, which is well suited for these kinds of tasks. The result of this MapReduce job was a file with count of words in the input file.

B. MAPREDUCE AND NEPHELE:

For Nephele, we used the same MapReduce program we wrote for the previously described Hadoop experiment and execute them on top of Nephele In order to do , we develop a set of wrapper classes providing interface compatibility with Hadoop and required functionality. These classes allowes us
to run the unmodified Hadoop MapReduce programs with Nephele. As a result, the data flow will be controlled by the executed MapReduce programs while Nephele is able to manage the allocation/deallocation of tasks to instances during the experiment.This experiment highlights the effect of thedynamic resource allocation/deallocation while still comparing to Hadoop. Nephele is used to create the communication paths that match the MapReduce processing pattern.

C.RESULTS:

Fig2 shows the performance comparison of the two experiment. The plot illustrates the average execution time based on the size of the datebase used in the experiment.

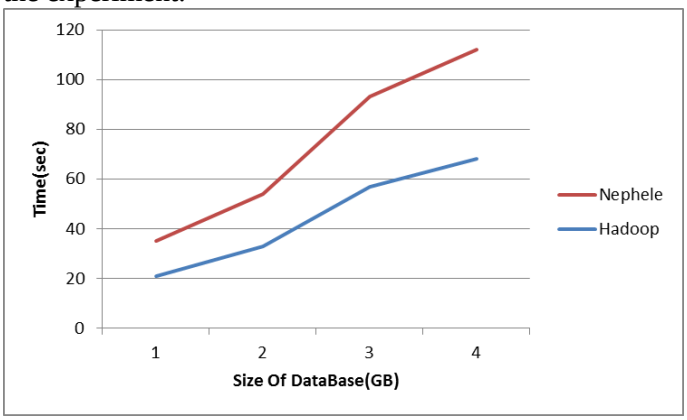

Fig2 . Performance Comparison between Hadoop and Nephele.

The plot shows that the average time taken to complete the MapReduce job takes much longer time in Hadoop when compared to Nephele framework.

V CONCLUSION:

This paper describes that the proposed nephele framework improves the performance of the system in efficient parallel data processing for dynamic resource allocation in cloud. Nephele Framework improves the performance of the system in-terms of time and cost.So we conclude the work by saying that proposed system will improve the performance of the system by improving the time taken to execute a job when compared to Hadoop and this ensures that the reliability of the system is also achieved in terms of cost,because the cloud storage space is rented on the basis of time.

REFERNCES

[1] Amazon Web Services LLC, "Amazon Elastic Compute Cloud (Amazon EC2)," http://aws.amazon.com/ec2/, 2009.

[2] Amazon Web Services LLC, "Amazon Elastic MapReduce," http://aws.amazon.com/elasticmapreduce/, 2009.

[3] Amazon Web Services LLC, "Amazon Simple Storage Service,"http://aws.amazon.com/s3/, 2009.

[4] D. Battre´, S. Ewen, F. Hueske, O. Kao, V. Markl, and D.Warneke,"Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing," Proc. ACM Symp.Cloud Computing (SoCC '10), pp. 119-130, 2010.

.[5] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proc. Sixth Conf. Symp. Opearting Systems Design and Implementation (OSDI '04), p. 10, 2004.

[6] A. Kivity, "Kvm: The Linux Virtual Machine Monitor," Proc. Ottawa Linux Symp. (OLS '07), pp. 225-230, July 2007.

[7] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L.Youseff, and D. Zagorodnov, "Eucalyptus: A Technical Report onan Elastic Utility Computing Architecture Linking Your Programsto Useful Systems," technical report,Univ. of California, SantaBarbara, 2008.

[8] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan, "Interpreting the Data: Parallel Analysis with Sawzall," Scientific Programming,vol. 13, no. 4, pp. 277-298, 2005.

[9] 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.

[10] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde, "Falkon:A Fast and Light-Weight TasK ExecutiON Framework," Proc.ACM/IEEE Conf. Supercomputing (SC '07), pp. 1-12, 2007.

[11] M. Stillger, G.M. Lohman, V. Markl, and M. Kandil, "LEO—DB2's LEarning Optimizer," Proc. 27th Int'l Conf. Very Large Data Bases (VLDB '01), pp. 19-28, 2001.

[12] The Apache Software Foundation "Welcome to Hadoop!" http://hadoop.apache.org/, 2009.

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

[14] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, and M.Wilde, "Swift: Fast, Reliable, Loosely Coupled Parallel Computation," Proc. Services, '07 IEEE Congress On, pp. 199-206, July 2007.

[15] D. Warneke and O. Kao, "Nephele: Efficient Parallel Data Processing in the Cloud," Proc. Second Workshop Many-Task Computing on Grids and Supercomputers (MTAGS '09), pp. 1-10,2009.