Cloud Computing via Parallel Information System supported by Harmonized Resource Appropriation

P.Sivaprakash

[#]*Computer Science and Engineering Department, RVS College of Engineering and Technology, Coimbatore, Tamil Nadu,India*

[1]sivaprakash04@gmail.com

***Abstract*— Ad hoc parallel data processing has emerged to be one of the modest applications for Infrastructure-as-a-Service (IaaS) clouds. For the large amount of submitting jobs the allocated compute resources may be inadequate and this increases the processing time and cost. This paper gives an overview about the data processing framework, Batch Harmonizing Nephele architecture explicitly exploits the dynamic resource allocation that runs on clouds following IaaS abstraction for task scheduling and execution. Thus here with prospects and challenges for competent parallel data processing in cloud reduces the time and cost. With this framework, we perform extended evaluations of Map Reduce-inspired processing jobs on an IaaS cloud system.**

***Keywords*— IaaS, Nephele, Load balancing and Batch harmonizing .**

I. INTRODUCTION

A revolution is defined as a change in the way people think and behave that is both dramatic in nature and broad in scope. Thus in the present days, cloud computing is indeed a revolution. Cloud computing is creating a fundamental change in computer architecture, software and tools development, and of course, in the way we store, distribute and consume information.

End users access cloud-based applications through a web browser or a light-weight desktop or mobile app while the business software and user's data are stored on servers at a remote location. Cloud computing entrusts services with a user's data, software and computation over a network.

There are three types of cloud computing:

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

Compared to previous paradigms, cloud computing focused on treating computational resources as measurable and billable utilities. From the clients' point of view, cloud computing provides an abstraction of the underlying hardware architecture. This abstraction saves them the costs of design, setup and maintenance of a data centre to host their Application Environments (AE). Whereas for cloud providers, the arrangement yields an opportunity to profit by hosting many AEs. This economy of scale provides benefits to both parties, but leaves the providers in a position where they must have an efficient and cost effective data center.

Here in this paper, Infrastructure as a Service (IaaS) is been discussed deeper. Infrastructure as a Service (IaaS) cloud computing focuses on providing a computing infrastructure that leverages system virtualization to allow multiple Virtual Machines (VM) to be consolidated on one Physical Machine (PM) where VMs often represent components of AEs. VMs are loosely coupled with the PM they are running on; as a result, not only can a VM be started on any PM, but also, it can be migrated to other PMs in the data center. Migrations can either be accomplished by temporarily suspending the VM and transferring it, or by means of a live migration in which the VM is only stopped for a split second. With the current technologies, migrations can be performed on the order of seconds to minutes depending on the size and activity of the VM to be migrated and the network bandwidth between the two. The ability to migrate VMs makes it possible to dynamically adjust data center utilization and tune the resources allocated to AEs. Furthermore, these adjustments can be automated through formally defined strategies in order to continuously manage the resources in a data center with less human intervention.

.

II. RELATED WORKS

In recent years a variety of systems to facilitate MTC has been developed. Although these systems typically share common goals (e.g. to hide issues of parallelism or fault tolerance), they aim at different fields of application. Map Reduce [2] (or the open source version Hadoop [3]) is designed to run data analysis jobs on a large amount of data, which is expected to be stored across a large set of share-nothing commodity servers. Map Reduce is highlighted by its simplicity: Once a user has fit his program into the required

map and reduce pattern, the execution framework takes care of splitting the job into subtasks, distributing and executing them. A single Map Reduce job always consists of a distinct map and reduce program. However, several systems have been introduced to coordinate the execution of a sequence of Map Reduce jobs [4], [5]. Map Reduce has been clearly designed for large static clusters. Although it can deal with sporadic node failures, the available compute resources are essentially considered to be a fixed set of homogeneous machines.

The Pegasus framework by Deelman [6] has been designed for mapping complex scientific workflows onto grid systems. Similar to Nephele, Pegasus lets its users describe their jobs as a DAG with vertices representing the tasks to be processed and edges representing the dependencies between them. The created workflows remain abstract until Pegasus creates the mapping between the given tasks and the concrete compute resources available at runtime. The authors incorporate interesting aspects like the scheduling horizon which determines at what point in time a task of the overall processing job should apply for a compute resource. This is related to the stage concept in Nephele. However, Nephele's stage concept is designed to minimize the number of allocated instances in the cloud and clearly focuses on reducing cost. In contrast, Pegasus' scheduling horizon is used to deal with unexpected changes in the execution environment. Pegasus uses DAGMan and Condor-G [7] as its execution engine. As a result, different task can only exchange data via files.

Thao introduced the Swift [8] 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. Building upon components like CoG Karajan [9], Falkon [10], and Globus [11], the authors present a scripting language which allows to create mappings between logical and physical data structures and to conveniently assign tasks to these. The system our approach probably shares most similarities with is Dryad [12]. Dryad also runs DAG-based jobs and offers to connect the involved tasks through file, network, or in-memory channels. However, it assumes an execution environment which consists of a fixed set of homogeneous worker nodes. The Dryad scheduler is designed to distribute tasks across the available compute nodes in a way that optimizes the throughput of the overall cluster. It does not include the notion of processing cost for particular jobs.

In terms of on-demand resource providing several projects arose recently: Dornemann [13] presented an approach to handle peak-load situations in BPEL workflows using Amazon EC2. Ramakrishnan [14] discussed how to provide a uniform resource abstraction over grid and cloud resources for scientific workflows.

Both projects rather aim at batch-driven workflows than the data-intensive, pipelined workflows Nephele focuses on. The FOS project [15] recently presented an operating system for multicore and clouds which is also capable of on-demand VM allocation.

III. PROBLEM STATEMENT

A cloud computing facility (data center) which is composed of a number of clusters is considered. This cloud computing environment has a central manager that has some information about all clusters as well as the clients. Clusters are characterized by the number and type of computing, data storage, and communication resources that they control.

The IaaS cloud providers integrate the processing framework to reduce the processing time and provide simplicity to the users. Reducing process time leads to reduction in the cost for attracting the users to use their cloud services. Several frameworks have been developed with some specific features (e.g. to reduce cost or increase performance) for cloud which reduce the complexities for the user. However, the existing well known frameworks like Google's Map Reduce, Yahoo's Map Reduce Merge need the job to be written in a distinct map and reduce program by the developer. Map Reduce is very rigid, forcing every computation to be structured as a sequence of map-reduce pairs. Nephele framework introduces some basic issues for Dynamic allocation of instances.

The existing Nephele framework has some difficulties with the resource overload and underutilization problems during job execution. And also Nephele needs more user annotations to execute the tasks.

IV. OVERVIEW OF BATCH HARMONIZING NEPHEL FRAMEWORK

Proposed framework shadows most of the existing Framework Nepheles functionality. This framework differs by including the resource monitor to check the availability of resource to avoid starvation by using a checker and resource manager to schedule the allocation and de-allocation in the Nephele Framework to accomplish load balancing automatically during job execution.

A. Architecture

Batch harmonizing framework follows a standard master-worker pattern to process the given sequential code in the IaaS cloud. The batch harmonizing framework describes the master node (i.e. VM) as Job Manager (JM) which is taking place before submitting the job to execute. The Job Manager, takes the clients' jobs, is responsible for scheduling them and coordinates their execution. It is capable of communicating with the Cloud Controller interface which the cloud operator offers to control the instantiation of VMs. By means of the Cloud Controller the Job Manager can allocate or deal locate VMs rendering to the current job execution phase. VMs are denoted by instances as per the common cloud computing terminology. The term instance type will be used to differentiate between VMs with different hardware characteristics. For example, the instance type "small

instance" could denote VMs with one CPU core, 1.7 GB of RAM, and a 160 GB disk while the instance type "Extra-large" could refer to machines with 8 CPU cores, 16 GB RAM and a 1690 GB disk. The executions of tasks are carried out by a set of in-stances called Task Managers (TM). These worker nodes receive one or more task from the Job Manager at a time, execute them, and after that, inform the Job Manager about their completion or possible errors. JM allocates the subtasks to the TM conferring to the type and size of the job. 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. Resource monitor is used to standardize the amount of sub-tasks being distributed to each instance (VMs) by using the Resource Manager. Resource Manager is the responsible for reallocating the subtasks to the instances according to the execution phase. The starvation checker will check that none of the job waits without the resource. Job execution with take place according to the multilevel feedback queue process by highest priority of jobs in the queue allocated for each resource. The jobs with move in the queue level according to the priority. Thus the job completion will be soon and de-allocation resource is been intimated by resource monitor.

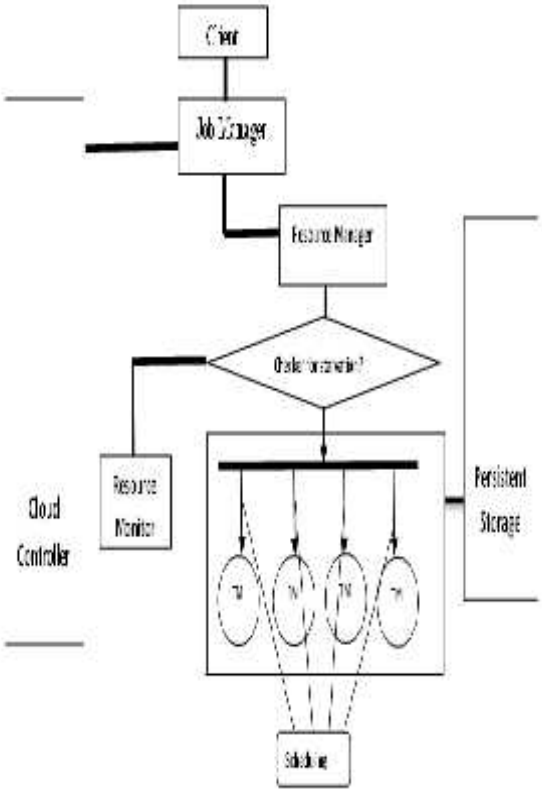

Fig.1 Structural overview of Batch Harmonizing Nephele Framework in an Infrastructure-as-a-Service (IaaS) cloud

B. Job Description

The job of the batch harmonizing Nephele Framework is articulated as a directed acyclic graph (DAG) which agrees tasks to have multiple inputs and output gates. Each vertex in the graph signifies a task of the overall processing job; the graph's edges define the communication flow between these tasks.

Defining a batch harmonizing Nephele job comprises three mandatory steps:

First, the user must write the program code for each task of his processing job or select it from an external library. Second, the task program must be assigned to a vertex. Finally, the vertices must be connected by edges to define the communication paths of the job.

The user has to submit the job as the DAG graph which specifies the tasks as the vertices and communication flow as the edges. This DAG graph is called as the Job Graph in the Extended Nephele framework. Users should be able to describe the tasks and the relationships on the abstract level.

Fig. 2. shows the simplest Job graph which involves one input, one task and one output vertex. For generating the job graph user must have some ideas about traits like number of subtasks, number of subtasks per instances, sharing instances between tasks, channel types and instance types for

job descriptions. Once the Job Graph is specified, the user submits it to the Job Manager together with the authorizations which the user has obtained from the cloud operator. The authorizations are required since the Job Manager must allocate/deallocate instances during the job execution.

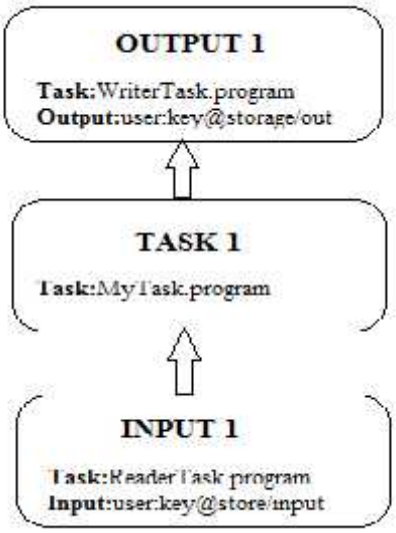

Fig. 2. An example of Job Graph.

C. Job Scheduling and Execution

After receiving the valid Job Graph the JM renovates it into the Execution Graph which is the primary data structure for scheduling and monitoring the execution of the Batch Harmonizing Nephele job. It contains all the existing information required to schedule and execute the tasks in the cloud. Fig. 3. shows the Execution Graph for the given Job Graph (i.e, Fig. 2.). Here Task 1 is, e.g., Split into two parallel subtasks which are both connected to the task Output 1 using file channels and are all scheduled to run on the same instance.

The Basic Execution Graph structure is no longer a pure DAG. It resembles in two different levels of details, an abstract level and a concrete level. On the abstract level, the Execution Graph equals the user's Job Graph. In the concrete level more fine-grained graph defines the mapping of subtasks to instances and the communication channels between them. Execution Graph consists of a Group Vertex for every vertex in Job Graph represent distinct tasks of the overall job. Execution stages are used to avoid the instance type availability problems in the cloud. Subtasks are represented in the Execution graph called Execution vertex which is controlled by its corresponding Group vertex. Each subtask is mapped to an Execution Instance which is defined by an ID and an instance type representing the hardware characteristics of the corresponding VM. After submitting the job to the JM, it divides the job into subtasks and schedules them into a number of Task managers according to the number of subtasks. These subtasks are given to the TM using the any type of channel according to the type of the job. The channel may be network, file or in-memory channel.

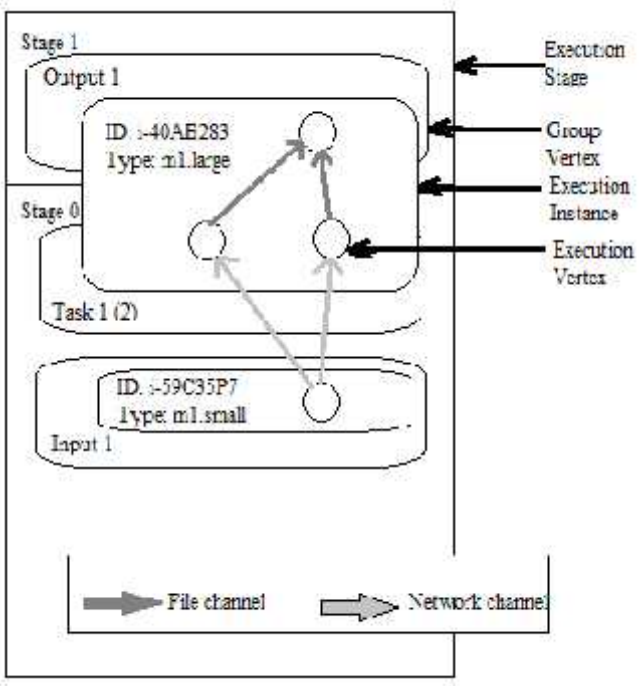

Fig. 3. An Execution Graph created from the original Job Graph.

D. Load Balancing

After the execution starts the TM must be monitored for providing better performance. While executing if any of the instances execute more tasks compared to others it may cause resource overload problem. Or at the same time any of the instances are process below the normal processing range it cause underutilization problem. These two problems take effect in the billing charges. To overcome this inside the IaaS cloud there must be a resource monitor check the TM and has to load balance the resource. This can be done by using user notification or by the job manager has to allocate/deal locate resource according to the memory used in the instances. The framework does a runtime based analysis and monitoring of the utilization of the allocated resources by the task that performs parallel computation with randomized version of multilevel feedback queue scheduling and solves the discrepancies (overutilization of underutilization of the allocated resources) and does the corresponding actions like allocating or de-allocating the needed resource and also avoids starvation of resources and thus saving the cost of computation involved in the task thereby reducing the time consumed and also the capital expenditure consumed.

E. Conclusion

In this paper we proposed a parallel processing frame-work Batch Harmonizing Nephele which avoids the resource overload and underutilization during job execution and performed a word count application based on this framework presented a performance comparison to the Nephele framework and establish out as a significant methodology for using the allocated resources efficiently without wasting the

resource thereby reducing the cost involved in the computation task. A Scheduling algorithm has been implemented to reduce more time.

ACKNOWLEDGMENT

I would like to thank my guide Prof.J.Jayavel working as Assistant Professor in Information Technology Department at Anna University, Regional Centre, Coimbatore, Tamil Nadu for his continuous support and guidance.

REFERENCES

[1]Daniel Warneke and Odej Kao Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud. In ieee transactions on parallel and distributed systems, January 2011.

[2]J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

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

[4]R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the Data: Parallel Analysis with Sawzall. Sci. Program., 13(4):277– 298, 2005.

[5]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.

[6]E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. Sci. Program.,13(3):219–237, 2005.

[7]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.

[8]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. In Services, 2007 IEEE Congress on, pages 199–206, July 2007.

[9]G. von Laszewski, M. Hategan, and D. Kodeboyina. Workflows for e-Science Scientific Workflows for Grids. Springer, 2007.

[10]I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde. Falkon: a Fast and Light-weight tasK executiON framework. In SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, pages 1–12, New York, NY, USA, 2007. ACM.

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

[12]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.

[13]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.

[14]L. Ramakrishnan, C. Koelbel, Y.-S. Kee, R. Wolski, D. Nurmi, D. Gannon, G. Obertelli, A. YarKhan, A. Mandal, T. M. Huang, K. Thyagaraja, and D. Zagorodnov. VGrADS: Enabling e-Science Workflows on Grids and Clouds with Fault Tolerance. In SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, pages 1–12, New York, NY, USA, 2009. ACM.

[15]D. Wentzlaff, C. G. III, N. Beckmann, K. Modzelewski, A. Belay, L. Youseff, J. Miller, and A. Agarwal. An Operating System for Multicore and Clouds: Mechanisms and Implementation. In SoCC '10: Proceedings of the ACM Symposium on Cloud Computing 2010, pages 3–14, New York, NY, USA, 2010. ACM.