Exploiting Dynamic Resource Allocation for Parallel Data Processing in Cloud Computing Environment

'Vinayak V. Awasare', 'PG Student of PCCOE, University of Pune Maharashtra India' ,

'Sudarshan Deshmukh', 'Assistant Professor, PCCOE, University of Pune, Maharashtra, India'

Abstract—The dynamic resource allocation in cloud computing has attracted attention of the research community in the last few years. Many researchers around the world have come up with new ways of facing this challenge. Number of Cloud provider companies has started to include frameworks for parallel data processing in their product which making it easy for customers to access these services and to deploy their programs. The processing frameworks which are currently used have been designed for static and homogeneous cluster setups. So the allocated resources may be inadequate for large parts of the submitted tasks and unnecessarily increase processing cost and time. Again due to opaque nature of cloud, static allocation of resources is possible, but vice-versa in dynamic situations.

The proposed new Generic data processing framework is intended to explicitly exploit the dynamic resource allocation in cloud for task scheduling and execution. Experimental result shows that our approach outperforms existing scenario. The performance gain over existing system i.e. Nephele by proposed approach is found average 45% in terms of execution time of number of tasks.

Index Terms —Cloud Computing, Cyclic Job Execution, Dynamic Resource Allocation, DAG, Resource Management, Resource Scheduling, Nephele.

1. INTRODUCTION

Cloud Computing is an essential ingredient of modern computing systems. Computing concepts, technology and architectures have been developed and consolidated in the last decades; many aspects are subject to technological evolution and revolution. Cloud Computing is an computing technology that is rapidly consolidating itself as the next step in the development and deployment of increasing number of distributed application.

Cloud computing is nothing but a specific style of computing where everything from computing power to infrastructure, business apps are provided as a service. It's a computing service rather than a product. In cloud, shared resources, software and information is provided as s metered service over the network. When the end user accesses some service is cloud, he is not aware of where that service is coming from or what is platform being used or where it is being stored [2].

Currently a number of companies have to handle large amounts of data in a cost-efficient manner. These companies are operators of Internet search engines, like Yahoo, Google or Microsoft. The huge amount of data or datasets they have to process every day has made traditional database solutions prohibitively expensive. So these numbers of growing companies have popularized an architectural paradigm based on a huge number of commodity servers. Problems like regenerating a web index or processing crawled documents are split into several independent subtasks, distributed among the available nodes, and computed in parallel.

The cloud computing paradigm makes the resource as a single point of access to the number of clients and is implemented as pay per use basis. Though there are number of advantages of cloud computing such as virtualized environment, equipped with dynamic infrastructure, pay per consume, totally free of software and hardware installations, prescribed infrastructure and the major concern is the order in which the requests are satisfied which evolves the scheduling of the resources. Allocation of resources has been made efficiently that maximizes the system utilization and overall performance. Cloud computing is mainly sold on demand on the basis of time constrains basically specified in hours or minutes. So the scheduling has to be done in such a way that the resource utilization has need done efficiently.

Nephele is the first data processing framework used for dynamic resource allocation offered by today's Infrastructure-as-a-Service (IaaS)clouds for both, task scheduling and task execution. Some tasks of a particular processing job can be assigned to different types of

virtual machines (VMs) which are automatically started and terminated during the job execution [1].

We proposed a new Generic Framework which dynamically allocates resources to the data processing Framework. The objective of our proposed approach is to reduce the execution time, migration time for resources and network latency. The performance gain over existing system i.e. Nephele by proposed approach is found average 45% in terms of execution time of number of tasks.

2. RELATED WORK

Dynamic resource allocation is one of the most challenging problems in the resource scheduling problems. The dynamic resource allocation in cloud infrastructure has capture attention of the number of research community in the last decade. Many researchers around the world have given number of solution for this challenging problem i.e. dynamic resource allocation in cloud infrastructure.

Now a day's number of growing companies has popularized an architectural paradigm based on a huge number of commodity servers. Problems like regenerating a web index or processing crawled documents are split into several independent subtasks, distributed among the available nodes, and computed in parallel. Simplify the development of such number of distributed applications on top of the architectures; some of the cloud provider companies have also built customized data processing frameworks. Examples are Google's MapReduce [7], Yahoo's Map-Reduce-Merge [6] or Microsoft's Dryad [8]. They can be classified by terms like or many-task computing (MTC) or high throughput computing (HTC) depending on the available amount of data and the number of tasks of a number of jobs involved in the computation [9]. These systems are not same in design but their execution models share similar objectives, fault tolerance, and execution optimizations from the number of developer. Software Developers can continue to write number of sequential programs and processing framework then distribute these programs among the available resources and executes each instance of these programs on the appropriate and available fragment of data.

For companies that only have to process huge amounts of datasets running their own data center is obviously not an option every time but now Cloud computing has emerged as a promising approach to rent a large IT infrastructure on a short-term pay-per-consume basis. Operators of so-called Infrastructure-as-a-Service (IaaS) clouds, like Amazon EC2 [3], let their customers control, allocate and access 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 dynamic. The VMs are typically expressed in different types, each type with its own characteristics such as amount of main memory, number of CPU cores, etc.

VM abstraction of Infrastructure as a Service (IaaS) clouds fits the architectural paradigm assumed by the data processing frameworks like Hadoop [10], a popular open source implementation of Google's MapReduce framework, already have begun to promote using their frameworks in the cloud [11]. Amazon EC2 cloud has integrated Hadoop as one of its core infrastructure services in its infrastructure [4]. However, instead of embracing its dynamic resource allocation, now available data processing frameworks can expect the cloud to use the static nature of the cluster environments they were originally designed for. E.g., at the moment the number and types of VMs allocated at the starting of a processing job cannot be changed in the time of processing, although the job consists of might have completely different demands on the environment.

For on-demand resource provisioning several approaches has been arose recently: Author has presented an approach to handle peak-load situations in BPEL workflows using Amazon EC2[11] and Author has given a solution how to provide a resource abstraction over grid computing and cloud resources for scientific workflows[21]. Both approaches rather point at batch-driven workflows than the pipelined, data-intensive workflows which Nephele focuses on. The FOS project [22] recently presented an operating system for multi core and clouds which is also capable of on-demand VM allocation.

Nephele is the first data processing framework used for dynamic resource allocation offered by today's Infrastructure-as-a-Service (IaaS) clouds for both, task scheduling and task execution. Some tasks of a particular processing job can be assigned to different types of virtual machines (VMs) which are automatically started and terminated during the job execution [1].

3. PROPOSED WORK

Proposed generic framework is a parallel and distributed programming framework written in core Java. Traditionally such frameworks are heavy as well as

complex in nature. We started with the concept that, we wanted to have been a simple model, very light weight and programmer intuitive. We also wanted to make sure that we should have a extremely scalable and very high performance framework in place.

3.1 System Architecture

Proposed system architecture follows a classic master-worker pattern is illustrated in following Fig. 3. The Job Manager receives the client's jobs, is responsible allocation them to available task manager, and coordinates their execution by communicating with task manager. Job Manager (JM) is capable of communicating with the interface the cloud operator provides to control the instantiation of VMs i.e. Cloud Controller. So both Job manager and Cloud Controller is responsible for allocate or deallocate VMs according to the current job execution phase.

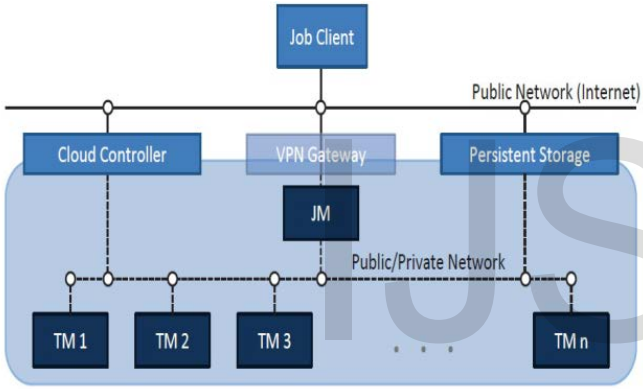

Fig. 1: Structural overview of Proposed Framework running in an Infrastructure-as-a-Service (IaaS) cloud.

The actual execution of tasks is carried out by a Task Manager. A Task Manager receives one or more tasks from the JM at a time, process them, and after that inform the JM about their completion or possible errors. Whenever jobs are received by JM then JM decides, depending on the particular tasks, what type and how many of instances the job should be executed on so according to that the respective instances must be allocated/deallocated to ensure a continuous but cost-efficient processing.

Here a Sample Scenario has been shown in Fig. 3 of how proposed generic framework works in cloud environment. Processes are Replicated and Deployed on three task managers which are managed by a Job Manager. Thus, whenever a request is received by a Cloud controller it passes it to the job manager. The job manager chooses the best available task manager for processing the request. The request is either processed entirely on a task manager or partially in parts by different task manager and response is sent back to the cloud controller.

3.2 Cyclic Job Scheduling Algorithm

Cyclic Job Scheduling:

Cyclic job scheduling is the method by which threads, processes or data flows are given access to system resources in cyclic manner. This is usually done to load balance and share system resources effectively or achieve a target quality of service. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking.

Algorithm

1: Initialize scheduler instance on cloud by starting project execution.
2: Read task configuration form task_configuration.xml file which contains number of tasks to be gets executed and flow of those task.
3: Perform Unmarshalling on input task.configuration.xml file. Unmarshelling process converts .xml file into number of Java Objects.
4: Initialize task job Pooler for queuing of given number of tasks.
5: Validate task_configuration.java file. If task_configuration.java file contains NULL values then go to Step 10 else go to Step 6.
6: Prepare execution path for bunch of available tasks waiting for execution in task pool.
7: Execute Initial task and validate result. Read out node configuration of completed task to find next task to execute.
8: Execute next task and validate results.
9: Read out node configuration of completed task to find next task to execute. If there is next task to execute then go to step 8 else go to step 10.
10: Terminate Execution and Stop

Fig. 2: Cyclic Job scheduling Algorithm

3.3 Mathematical Equations

Two major parameters we used to compare our proposed system with existing system. Those two parameters are execution time of number of tasks and line of code of input .xml file to the framework.

- **Execution Time (ET):**

The execution time or CPU time of a given task is defined as the time spent by the system executing that task, including the time spent executing run-time or system services on its behalf. The mechanism used to measure execution time is implementation defined. It is implementation defined which task, if any, is charged the execution time that is consumed by interrupt handlers and run-time services on behalf of the system.

We used following formula to calculate Execution time of number of tasks.

Request Task Set = { T1, T2, T3,…..Tn }

Resource Set = { R1,R2,R3,…….Rn }

Ri is resource allocated for request task Ti

Ti serves by Ri

ET = Finish time of Ti – Starting time of Ti

Where ET is Execution Time

3.4 General Hardware Setup

All experiments will be conducted on local IaaS cloud of commodity servers. Each server is equipped with two Xeon 2 which having 66 GHz CPUs (8 CPU cores) and a total main memory of 32GB. All servers will be connected through regular 1 GBit/s Ethernet links. The host operating system will be Linux (kernel version 2.6.30) with KVM [15] (version 88-r1) using virtio[23] to provide virtual I/O access.

To manage the cloud and provision VMs on request of Generic proposed framework, we set up Eucalyptus [16] which is much similar to Amazon EC2, Eucalyptus offers a predefined set of instance types a user can choose from. During our experiments we will consider two different instance types: The first type of instance will bex1.small which corresponds to an instance with one GB of RAM, one CPU core, and a 128GB disk. The second type of instance will be y1.xlarge, represents an instance which will having18 GBRAM, 8 CPU cores and a 512 GB disk.

4.RESULT AND DISCUSSIONS

Following tables shows the comparison of existing systems with proposed system in terms of execution time in minutes per number of tasks allocated to particular system.

TABLE I
Comparison in terms of Execution time

Systems \ Number of Tasks	5	20	40	60	80
MapReduce and Hadoop	4.768	19.767	38.12	74.892	152.9
MapReduce and Nephele	6.718	23.767	47.4	97.21	188.21
DAG and Nephele	1.968	5.767	11.43	23.99	44.12
Proposed Framework	0.238	2.75	4.1	11.21	23.12

Following graph shows the comparison of existing systems with proposed system in terms of execution time in minutes per number of tasks allocated to particular system.

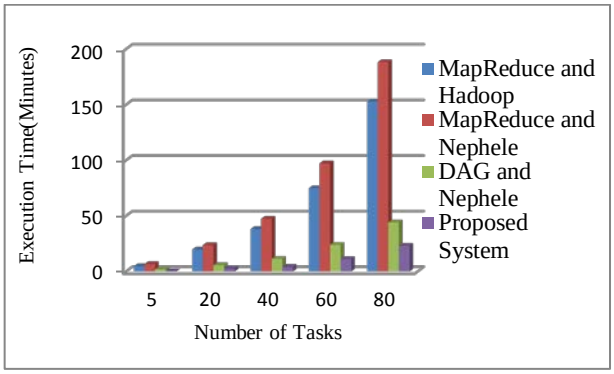

Fig. 3: Comparison of Existing systems with proposed system in terms of Executing time (Minutes) per number of tasks

5. CONCLUSION

In particular, we are interested in improving Framework's ability to adapt to resource overload or underutilization during the job execution automatically. We proposed a new Generic Framework which dynamically allocates resources and which uses cyclic job execution algorithm. Proposed System eliminates the drawback of existing frameworks that is acyclic job execution and improper resource utilization. Proposed Framework avoids rewriting of input file source code which reduces the overall cost of system.

The proposed new Generic data processing framework is intended to explicitly exploit the dynamic resource allocation in cloud for task scheduling and execution. Experimental result shows that our approach outperforms existing scenario. The performance gain over existing system i.e. Nephele by proposed approach is found average 45% in terms of execution time of number of tasks.

REFERENCES

[1] Daniel Warneke and Odej Kao, "Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, JANUARY 2011.

[2] Zhen Xiao, Senior Member, IEEE, Weijia Song, and Qi Chen, "Dynamic Resource AllocationUsing Virtual Machines for Cloud

Computing Environment",IEEE TRANSACTIONSON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 6, JUNE 2013.

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

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

[5] R. Chaiken, B. Jenkins, P.-A.Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou.SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets. Proc. VLDB Endow.,1(2):1265 1276, 2008.

[6] H. chih Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-Reduce-Merge: SimplifiedRelational Data Processing on Large Clusters. In SIGMOD 07: Proceedings of the 2007ACM SIGMOD international conference on Management of data, pages 10291040, NewYork, NY, USA, 2007. ACM.

[7] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. InOSDI04: Proceedings of the 6th conference on Symposium on Opearting Systems Designand Implementation, pages 1010, Berkeley, CA, USA, 2004. USENIX Association.

[8] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed Data-ParallelPrograms from Sequential Building Blocks. In EuroSys 07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pages 5972, NewYork, NY, USA, 2007. ACM.

[9] I. Raicu, I. Foster, and Y. Zhao. Many-Task Computing for Grids and Supercomputers.In Many-Task Computing on Grids and Supercomputers, 2008. MTAGS 2008.Workshopon, pages 111, Nov. 2008.

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

[11] T. White. Hadoop: The Definitive Guide. OReilly Media, 2009.

[12] D. Battre, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke. Nephele/PACTs: AProgramming Model and Execution Framework for Web-Scale Analytical Processing. InSoCC 10: Proceedings of the ACM Symposium on Cloud Computing 2010, pages 119130, New York, NY, USA, 2010. ACM.

[13] M. Armbrust et al., Above the Clouds: A Berkeley View of Cloud Computing, technicalreport, Univ. of California, Berkeley, Feb. 2009.

[14] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R Neugebauer, I. Pratt,and A. Warfield, Xen and the Art of Virtualization, Proc. ACM Symp. Operating SystemsPrinciples (SOSP 03), Oct. 2003.

[15] V. Vinothina, Dr. R. Shridaran, and Dr. PadmavathiGanpathi, A survey on resource allocation
strategies in cloud computing, International Journal of Advanced Computer Scienceand Applications, 3(6):97–104, 2012.

[16] Gunho Lee, NirajTolia, ParthasarathyRanganathan, and Randy H. Katz, Topology awareresorce allocation for data-intensive workloads, ACM SIGCOMM Computer CommunicationReview, 41(1):120–124, 2011.

[17] Abirami S.P. and ShaliniRamanathan, Linear scheduling strategy for resource allocationin cloud environment, International Journal on Cloud Computing: Services and Architecture(IJCCSA), 2(1):9–17, 2012.

[18] DanielWarneke and Odej Kao, Exploiting dynamic resource allocation for efficient parralleldata processing in the cloud, IEEE Transactions On Parallel And Distributed Systems,2011.

[19] Atsuo Inomata, TaikiMorikawa, Minoru Ikebe and Md. MizanurRahman, Proposal andEvaluation of Dynamic Resource Allocation Method Based on the Load Of VMs on IaaS,IEEE, 2010.

[20] N. Roy, A. Dubey, A. Gokhale, and L. Dowdy, A Capacity Planning Process forPerformance Assurance of Component-based Distributed Systems, in Proceedings ofthe 2nd ACM/SPEC International Conference on Performance Engineering (ICPE2011). Karlsruhe, Germany: ACM/SPEC, Mar. 2011, pp. 259270.

[21] 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.

[22] D. Wentzlaff, C. G. III, N. Beckmann, K. Modzelewski, A. Belay,L. Youseff, J. Miller, and A. Agarwal. An Operating System forMulticore 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.

Vinayak V. Awasare graduated in Computer Engineering from the University of Pune (India) in 2012. He is pursuing his Master of engineering in Computer Engineering from the University of Pune (India).He is currently research scholar student in Computer Department, Pimpri Chichwad College of Engineering; Pune (India).His research interests include wireless networking, Load Balancing and Game Theory.

Sudarshan S. Deshmukh graduated in Computer Engineering from the University of Shivaji (India) in 2004. He received his Masters in Computer Engineering from the Bharati University in 2009. He is currently working as an assistant professor, Computer Engg, at PCCOE, University of Pune since 2009. He is a member of the Technical Committee of Parallel Processing (TCPP), IEEE communication society, IAENG etc. Received Nomination for IEEE Technical Committee on Parallel Processing Outstanding Service Award for 2011.Associate Editor of International Journal of Cloud Applications and Computing, also serving as reviewer to several journals and conferences His research interests include distributed systems, resource sharing, load balancing.