International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 2 Issue: 8 2460 – 2465

Dynamic Resource Allocation for Parallel Data Processing in Cloud Computing Environment

Vinayak V. Awasare
PG Student of PCCOE, Department of Computer Engineering, University of Pune-44, Maharashtra, India
vinayak.awasare009@gmail.com

Prof. Sudarshan S. Deshmukh
Assistant Professor, PCCOE, Department of Computer Engineering, University of Pune-44, Maharashtra, India
deshmukh.sudarshan@gmail.com

***Abstract*—** Dynamic resource allocation problem is one of the most challenging problems in the resource management problems. 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. Ad-hoc parallel data processing has emerged to be one of the killer applications for Infrastructure-as-a-Service (IaaS) cloud. 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.

Keywords *—Cloud Computing, Dynamic Resource Allocation, Resource Management, Resource Scheduling.*

*****

I. 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.

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 [3].

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.

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 applications. The objective of our proposed approach is to reduce the execution time, migration time for resources and network latency**.**

II. 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 [5].

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.

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 [4], 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.

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. [1]

III. RESOURCE ALLOCATION STRATEGIES AND ALGORITHMS

Recently many resource allocation strategies have come up in the literature of cloud computing environment as this technology has started maturing. Number of Researcher communities around the world have proposed and implemented several types of resource allocation. Some of the strategies for resource allocation in cloud computing environment are discuss here briefly.

A. *Topology Aware Resource Allocation (TARA) for cloud*

Different types of resource allocation strategies are proposed in cloud. The author mentioned in [2] the architecture for resource allocation in Infrastructure-as-a-Service (IaaS) based cloud systems

An architecture which adopts a "what if" methodology to guide allocation decisions taken by the IaaS is proposed to address this resource allocation problem. The architecture uses a prediction engine with a lightweight simulator to estimate the performance of a given resource allocation and a algorithm to find an optimized solution in the large search space.

Results showed that Topology Aware Resource Allocation reduced the job completion time of these applications by up to 59 percent when compared to application-independent allocation policies

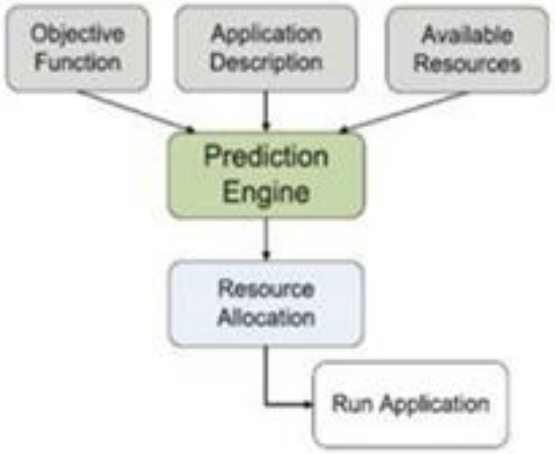

Fig. 1: Basic Architecture of TARA [2]

B. *Linear Scheduling Strategy for Resource Allocation for cloud*

Considering the processing time, resource utilization based on CPU usage, memory usage and throughput, the cloud environment with the service node to control all clients request, could provide maximum service to all clients [3]. A scheduling algorithm named as Linear Scheduling for Tasks and Resources (LSTR) is designed, which performs tasks and resources scheduling respectively. Here, a server node is used to establish the IaaS cloud environment and KVM/Xen virtualization along with LSTR scheduling to allocate resources which maximize the system throughput and resource utilization.

Resource consumption and resource allocation have to be integrated so as to improve the resource utilization. The

scheduling algorithms mainly focus on the distribution of their sources among the requestors that will maximize the selected QoS parameters. The QoS parameter selected in our evaluation is the cost function. The scheduling algorithm is designed considering the tasks and the available virtual machines together and named LSTR scheduling strategy. This is designed to maximize the resource utilization.

C. *Dynamic Resource Allocation for Parallel Data Processing using Nephele*

Dynamic Resource Allocation for Efficient Parallel data processing [2] introduces a new processing framework explicitly designed for cloud environments called Nephele Most notably, Nephele is the first data processing framework to include the possibility of dynamically allocating or deallocating different compute resources from a cloud in its scheduling and during job execution.

Nephele's architecture [1] follows a classic master-worker pattern as illustrated in Fig.2. 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 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. 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).

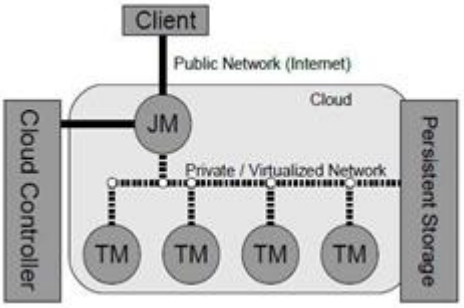

Fig. 2: Basic Design Architecture of Nephele Framework [1]

D. *Comparative Study of Resource Allocation Strategies*

An optimal Resource Allocation Strategy should solve the following problems of resource allocation:

Resource Contention - Resource contention arises when two applications try to access the same resource at the same time.

Scarcity of Resource - Scarcity of resource arises when there are limited resources and the demand for resources is high.

Resource Fragmentation - Resource fragmentation arises when the resources are isolated. There would be enough resources but cannot allocate it to the needed application due to fragmentation into small entities.

Over Provisioning - Over provisioning arises when the application gets surplus resources than the demanded one.

Under Provisioning - Under provisioning of resources occurs when the application is assigned with fewer numbers of resources than it demanded.

Following table shows optimal resource allocation strategy which solves all problems of resource allocation those describe above.

TABLE I.
Comparative Analysis of Resource Allocation Strategies

Sr. No.	Parameter	Linear Scheduling Strategy for Resource Allocation	Topology Aware Resource Allocation	Dynamic Resource Allocation (Nephele)
1]	Resource Contention	This does not Solve	This Solves	This Solves
2]	Scarcity of Resource	This does not Solve	This does not Solve	This Solves
3]	Resource Fragmentation	This does not Solve	This does not Solve	This Solves
4]	Over Provisioning	This does not Solve	This does not Solve	This Solves
5]	Under Provisioning	This does not Solve	This Solves	This Solves

IV. 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.

A. *System Architecture*

Proposed system architecture follows a classic master-worker pattern is illustrated in following Fig. 3. The Job

 ISSN: 2321-8169
2460 – 2465

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.

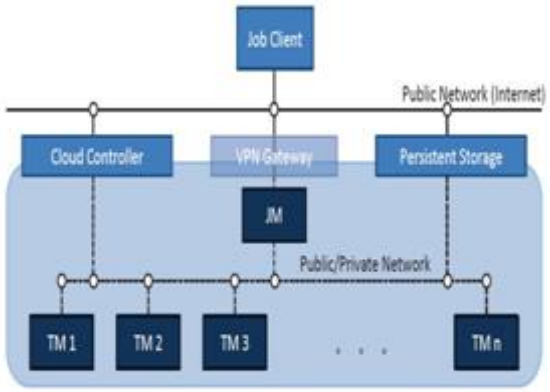

Fig. 3: Structural overview of Proposed Framework running in an Infrastructure-as-a-Service (IaaS) cloud.

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.

B. 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.

Following Fig. shows the flow of Cyclic Job Scheduling Algorithm in detail.

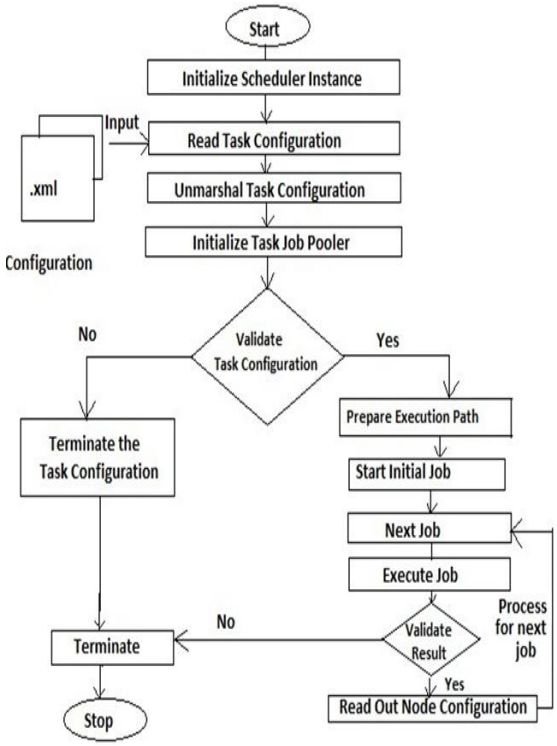

Fig. 5: Flow Diagram of Cyclic Job Scheduling Algorithm

C. 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.

1) LOC:

Lines of code (LOC) is a software metric used to measure the size of a computer program by counting the number of lines in the text of the program's source code. LOC is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate programming productivity or maintainability once the software is produced.

V.RESULT AND DISCUSSIONS

A. Comparison of LOC of Input .XML File

Following tables shows the comparison of existing systems with proposed system in terms of LOC of input .xml file to the system.

TABLE III
Comparison in terms of number of LOC of input .xml file

Number of Taks / *Systems*	20	40	60	80	100
DAG and Nephele	164	247	330	413	496
Proposed Framework	95	95	95	95	95

Following Graph shows the comparison of existing systems with proposed system in terms of LOC of input .xml file to the system.

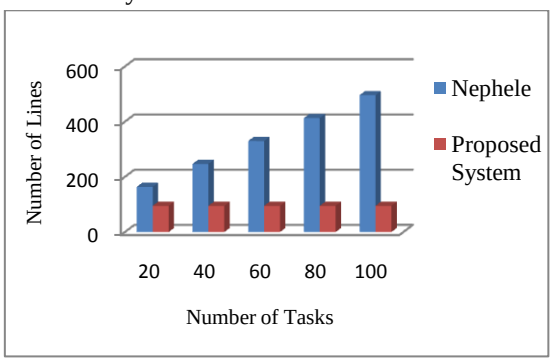

Fig. 7: Comparison of Existing systems with proposed system in terms of LOC of input .xml file to the system.

VI. 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] Ronak Patel, Sanjay Patel ," Survey on Resource Allocation Strategies in Cloud Computing ",International Journal of Engineering Research & Technology (IJERT) , ISSN: 2278-0181, Vol. 2 Issue 2, February- 2013

[3] 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.

[4] Amazon Web Services LLC. Amazon Simple Storage Service. http: //aws.amazon.com/s3/, 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: Simplified Relational Data Processing on Large Clusters. In SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 1029–1040, New York, NY, USA, 2007. ACM.

[7] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang. Maximum Likelihood Network Topology Identification from Edge-Based Unicast Measurements. SIGMETRICS Perform. Eval. Rev., 30(1):11–20, 2002.

[8] R. Davoli. VDE: Virtual Distributed Ethernet. Testbeds and Research Infrastructures for the Development of Networks & Communities, International Conference on, 0:213–220, 2005.

[9] J. Dean and S. Ghemawat. MapReduce: Simplified Data Process-ing 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.

[10] 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.

[11] T. Dornemann, E. Juhnke, and B. Freisleben. On-Demand Re-source 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.

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

[13] 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.

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

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

[16] 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.

[17] 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.

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

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

[20] 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. Workshop on, pages 1–11, Nov. 2008.

[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.

Authors

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 Cloud Computing, Load Balancing and Parallel Computing.

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.