ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 2, Febraury 2014

Dynamic Resource Allocation in Cloud Environment

M.Vanitha, Ar.Sivakumaran, S.Shiny Swarna Sugi

Assistant Professor, Department of Computer Science and Engineering, Karpagam College of Engineering Coimbatore, Tamilnadu, India.

Associate Professor, Department of Computer Science and Engineering, Karpagam College of Engineering, Coimbatore, Tamilnadu, India

Assistant Professor, Department of Computer Science and Engineering, Karpagam College of Engineering Coimbatore, Tamilnadu, India

ABSTRACT: Cloud computing allows business customers to scale up and down their resource usage based on needs. Many of the touted gains in the cloud model come from resource multiplexing through virtualization technology. In this paper, we present a system that uses virtualization technology to allocate data center resources dynamically based on application demands and support green computing by optimizing the number of servers in use. We introduce the concept of "skewness" to measure the unevenness in the multidimensional resource utilization of a server. By minimizing skewness, we can combine different types of workloads nicely and improve the overall utilization of server resources. We develop a set of heuristics that prevent overload in the system effectively while saving energy used. Trace driven simulation and experiment results demonstrate that our algorithm achieves good performance.

KEYWORDS: cloud computing; green computing.

I. INTRODUCTION

Cloud Computing is a marketing term for technologies that provide computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. A parallel to this concept can be drawn with the electricity grid, wherein end-users consume power without needing to understand the component devices or infrastructure required to provide the service. Cloud Computing is now-a-days being used for on demand storage and processing power. It allows the leasing of resources to improve the locally available computational capacity when necessary. When using a cloud, the user accesses computing resources as general utilities that can be leased and released. The ubiquitous access to cloud resources easily enables the simultaneous use of different clouds.

II. CREATION OF CLOUD NETWORK USING CLOUDSIM

A simulation toolkit enables modeling and simulation of Cloud computing systems and application provisioning environments. The CloudSim toolkit supports both system and

behavior modeling of Cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies. It implements generic application provisioning techniques that can be extended with ease and limited effort. Currently, it supports modeling and simulation of Cloud computing environments consisting of both single and inter-networked clouds (federation of clouds). Moreover, it exposes custom interfaces for implementing policies and provisioning techniques for allocation of VMs under inter-networked Cloud computing scenarios. In cloud initialization module, the creation of cloud users, datacenters and cloud virtual machines as per our requirement. Nephe Job manager, Cloud controller also created. The Job Manager receives the client's jobs, is responsible for scheduling

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 2, Febraury 2014

them, and coordinates their execution. It is capable of communicating with the interface the cloud operator provides to control the instantiation of VMs. The interface is called the Cloud Controller. By means of the Cloud Controller the Job Manager can allocate or deallocate VMs according to the current job execution phase. The term instance type will be used to differentiate between VMs with different hardware characteristics.

III.PARALLELIZATION AND SCHEDULING STRATEGIES USING EXECUTION GRAPH

Nephele separates the Execution Graph into one or more so-called Execution Stages. An Execution Stage must contain at least one Group Vertex. Its processing can only start when all the subtasks included in the preceding stages have been successfully processed. Based on this Nephele's scheduler ensures the following three properties for the entire job execution: First, when the processing of a stage begins, all instances required within the stage are allocated. Second, all subtasks included in this stage are set up and ready to receive records. Third, before the processing of a new stage, all intermediate results of its preceding stages are stored in a persistent manner. Hence, Execution Stages can be compared to checkpoints. In case a sufficient number of resources cannot be allocated for the next stage, they allow a running job to be interrupted and later on restored when enough spare resources have become available. Before processing a new Execution Stage, the scheduler collects all Execution Instances from that stage and tries to replace them with matching cloud instances. If all required instances could be allocated the subtasks are distributed among them and set up for execution. In case of task parallelization, when a Group Vertex contains more than one Execution Vertex, the developer of the consuming task can implement an interface which determines how to connect the two different groups of subtasks. Nephele requires all edges of an Execution Graph to be replaced by a channel before processing can begin. The type of the channel determines how records are transported from one subtask to the other. Currently, Nephele features three different types of channels, which all put different constrains on the Execution Graph.

IV.PARALLELIZATION AND SCHEDULING STRATEGIES USING ON-LINE PREEMPTIVE SCHEDULING

In the preemptive scheduling module implements the parallelization and scheduling strategies using on line preemptive scheduling. There are five main parts in the scheduling. They are the preemption checking, feasibility checking, task selecting, scheduling point checking, and critical point checking. When new tasks are added in to ready queue, not matter whether there is preemption or not, the feasibility checking will work to check if the new ready queue is feasible or not. If any task cannot meet the requirement, it will be removed from the ready queue. Scheduling point checking makes sure all the left tasks in the ready queue will have a qualified expected accrued utility density at the expected finish time of current running task. The task selecting has the policy that always select the highest expected accrued utility density task to run when the server is idle. The critical point checking will always monitor the current running task's state to prevent the server wasting time on the non-profitable running task. The preemption checking works when there is a prosperous task wants to preempt the current task. The combination of these parts guarantees to judiciously schedule the tasks for achieving high accumulated total utilities. It is worthy to talk more about the preemption checking part in details; because improper aggressive preemption will worsen the scheduling performance. If a task can be finished successfully before its deadline even in its worst case, the scheduling will protect the current running task from being preempted by any other tasks. Otherwise, if a prosperous task has an expected accrued utility density which is larger than the current running task's conditional expected accrued utility density by at least a value equals to the pre-set preemption threshold, the preemption is permitted.

V.PARALLELIZATION AND SCHEDULING STRATEGIES USING ON-LINE NON PREEMPTIVE SCHEDULING

In non-preemptive scheduling module, the parallelization and scheduling strategies using on line non-pre emptive scheduling. Non-preemptive scheduling algorithm of real-time services based on a task model similar to the profit and penalty model introduced. In addition to the careful choice of the ready task to run, scheduling method judiciously discards pending requests and aborts task executions, and therefore can achieve better performance. on-line non-preemptive scheduling solution to address the problem defined in the previous section. 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

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 2, Febraury 2014

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.

VI.PERFORMANCE EVALUATION

Average resource utilization, Total profit, total utility and total penalty are the performance metrics in performance evaluation module. These performances are evaluated under different thresholds and with various numbers of resources and users. It is interesting to see that the highest utility does not always occur at the point when the threshold equals zero, the highest utility will seldom occur at the point with neither the lowest nor the highest threshold value. The lower the threshold, the more tasks can be accepted to the system and get executed. This helps to improve the value of total profit. However, having more tasks accepted into ready queue may potentially increase the penalty cost. On the contrary, using a higher threshold helps to control the potential penalty but may limit the total profit that can be obtained.

VII.RESULTS AND DISCUSSION

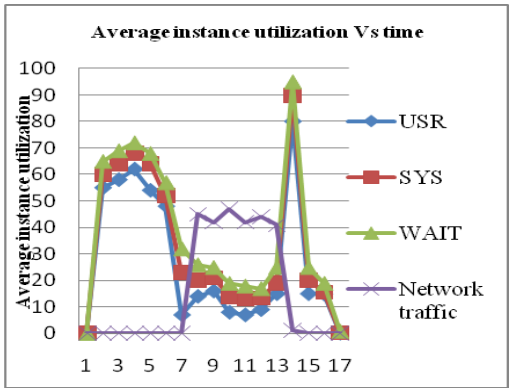

Figure Average Instance Utilization Vs Time

The utilization of each instance has been monitored with the UNIX command top and is broken down into the amount of time the CPU cores spent running the respective data processing frame work (USR), the kernel and its processes (SYS), and the time waiting for I/O to complete (WAIT). Here if the time is increased, the average instance utilization of four methods is varied. The variation depends on time. Initially it increased the utilization then the rate is varied gradually.

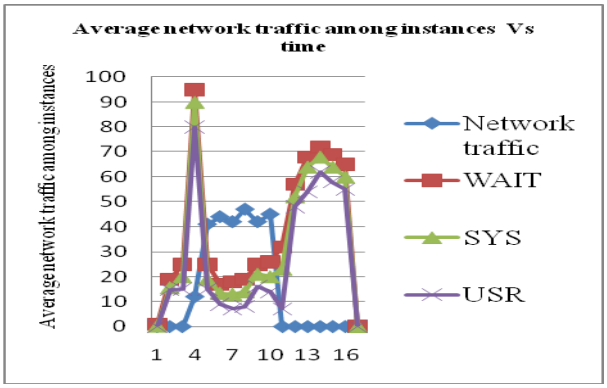

Figure. Average Network Traffic Among Instances Vs Time

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 2, Febraury 2014

The comparison of average network traffic among instances with time is explained in the above graph. The utilization of each instance has been monitored with the UNIX command top and is broken down into the amount of time the CPU cores spent running the respective data processing frame work (USR), the kernel and its processes (SYS), and the time waiting for I/O to complete (WAIT). Here if the time is increased, the average network traffic among instances of four methods is varied. The variation depends on time. Initially it increased the network traffic then the rate is varied gradually.

VIII.CONCLUSION

The challenges and opportunities for efficient parallel data processing in cloud environments and Nephele, the first data processing framework to exploit the dynamic resource provisioning offered by today's IaaS clouds are discussed. The Nephele's basic architecture and performance comparison to the well-established data processing framework Hadoop are implemented. 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.

REFERENCES

[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] 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. Very Large Database Endowment, vol. 1, no. 2, pp. 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," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2007.
[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 Performance Evaluation Rev., vol. 30, no. 1, pp. 11-20, 2002.
[8] R. Davoli, "VDE: Virtual Distributed Ethernet," Proc. Testbeds and Research Infrastructures for the Development of Networks and Communities, Int'l Conf., pp. 213-220, 2005.
[9] 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.
[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," Scientific Programming, vol. 13, no. 3, pp. 219-237, 2005.
[11] T. Dornemann, E. Juhnke, and B. Freisleben, "On-Demand Resource Provisioning for BPEL Workflows Using Amazon's Elastic Compute Cloud," Proc. Ninth IEEE/ACM Int'l Symp. Cluster Computing and the Grid (CCGRID '09), pp. 140-147, 2009.
[12] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," Int'l J. Supercomputer Applications, vol. 11, no. 2, pp. 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, vol. 5, no. 3, pp. 237-246, 2002.
[14] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," Proc. Second ACM SIGOPS/EuroSys European Conf. Computer Systems (EuroSys '07), pp. 59-72, 2007.
[15] A. Kivity, "Kvm: The Linux Virtual Machine Monitor," Proc. Ottawa Linux Symp. (OLS '07), pp. 225-230, July 2007.

BIOGRAPHY

1. **Prof.M.VANITHA** received her M.E degree in Computer Science and Engineering in V.L.B. Janakiammal College of Engineering and Technology, Coimbatore, Tamilnadu. . She is at present working as a Assistant Professor in department of Computer Science and Engineering in Karpagam College of Engineering, Coimbatore, Her area of interest is Cloud Computing , Data Mining.

2.**Prof. AR. Sivakumaran** received his M.C.A degree from R.V.S.College of Engineering, Dindigul and M.Tech. degree in computer science and engineering from Motilal Nehru National Institute of Technology, Allahabad. He is at present working as a Associate Professor in department of Computer Science and Engineering in Karpagam College of Engineering, Coimbatore, Tamil Nadu, India. His field of interest is Data Mining, Cloud Computig and Bio Informatics.