Webpage: www.ijaret.org

Volume 2, Issue IX Sep 2014
ISSN 2320-6802

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY

WINGS TO YOUR THOUGHTS…..

PAST SERVICE PERFORMANCE BASED RESOURCE SCHEDULING IN CLOUD COMPUTING

Rishu

Maharishi Markandeshwar Engineering College
Maharishi Markandeshwar University
Mullana, Ambala, Haryana, India
rishugulati13@gmail.com

***Abstract**: In the field of cloud computing decision making for allocation of dynamic resources is the most important and valuable process. Brokers are used to allocate the requests and to generate the profit by implementing efficient scheduling algorithms. In this work we have evaluated the past performance of the resources for future queries.*

***Keywords:** Service, Scheduling, Broker.*

1. INTRODUCTION

Nowadays, many applications are developed on a cloud environment. However, in order to make them scalable and fully elastic, we need to adopt new design patterns in queue management. The event-driven architecture (EDA) is probably the most important architectural pattern used in the cloud for making applications scale. Indeed, this pattern enables asynchronous communication between distributed application instances and fits perfectly to the dynamic and elastic nature of the cloud. Going further, the Infrastructure as a Service (IaaS) management components, such as the decision layer used for auto-scaling decisions, the monitoring system, and other components of the cloud infrastructure all require adapted messaging middlewares in order to efficiently transport the tremendous number of control messages they receive. These messaging middle wares would have to be highly scalable and should scale elastically. Originating from the field of physics and economics, the term elasticity is nowadays heavily used in the context of cloud computing. In this context, elasticity is commonly understood as the ability of a system to automatically provision and deprovision computing resources on demand as workloads change. However, elasticity still lacks a precise definition as well as representative metrics coupled with a benchmarking methodology to enable comparability of systems. Existing definitions of elasticity are largely inconsistent and unspecific, which leads to confusion in the use of the term and its differentiation from related terms such as scalability and efficiency; the proposed measurement methodologies do not provide means to quantify elasticity without mixing it with efficiency or scalability aspects.

To underline this observation, we cite five definitions of elasticity demonstrating the inconsistent use and understanding of the term:

1. ODCA, Compute Infrastructure-as-a-Service defines elasticity as the configurability and expandability of the solution Centrally, it is the ability to scale up and scale down capacity based on subscriber workload.”
2. NIST Definition of Cloud Computing ”Rapid elasticity: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.”
3. IBM, Thoughts on Cloud, Edwin Schouten, ”Elasticity is basically a ’rename’ of scalability and removes any manual labor needed to increase or reduce capacity.”
4. Rich Wolski, CTO, Eucalyptus, 2011 ”Elasticity measures the ability of the cloud to map a single user request to different resources.”
5. Reuven Cohen, Elasticity is ”the quantifiable ability to manage, measure, predict and adapt responsiveness of an application based on real time demands placed on an infrastructure using a combination of local and remote computing resources.”

Therefore in elastic queue *distributed applications often need to exchange data among processes. Coordinating the execution of a workflow and exchanging results of a computation are just two examples. Message queues are often used for this*

Webpage: www.ijaret.org
Volume 2, Issue IX Sep 2014
ISSN 2320-6802

kind of data exchange. we might think **exchanging simple messages** is simple enough that you do not need **middleware** for it. Like so many operations that sound simple at first, message passing can quickly get complicated. This work outlines tips on when to use message queues with the event of elasticity. In its most basic form, a **message queue is a service** that receives data from a sending process and delivers it to a receiving process. This helps in designing the system components which can operate independently of each other. This is a critical property of systems that are designed to scale.

2. RELATED STUDY

There have been several efforts in studying the deployment of scientific and high performance applications on various cloud computing platforms [1], [2], [3]. Our work differs by showing the deployment of a high performance application on multiple cloud computing platforms through a cloud computing framework. We also establish guidelines for the design, implementation, and identification of cloud computing frameworks in this work. There have also been several efforts in building and migrating bio-molecular applications to distributed computing environments [4], [5], [6]. The work in [4] present a framework that provides fault-tolerance and failure recovery for running replica-exchange simulations on distributed systems. This is achieved through check pointing and an external interface that monitors the execution of distributed applications. Work Queue differs by offering these functionalities inherently without overheads. The authors in [5] describe their experiences in running a replica-exchange simulation software, NAMD, on the Condor grid. They add a dedicated set of resources to speedup slow replicas executing on the grid and notice improvement in the efficient usage of available resources. The authors go on to present a database architecture for storing and retrieving bio-molecular simulation results.The work in [6] describes experiences in using Legion, an operating system that provides abstractions for managing and utilizing grid resources, to run replica-exchange simulations built using MPI. This work provides good insights on the effectiveness of abstractions in providing a seamless transition for users porting applications to run on grids. The system in [7] provides shorter time-to-result of a simulation of large protein systems. It uses a combination of distributed and parallel computing techniques to achieve this.

3. SYSTEM MODEL

In the system model we can observe that broker is handling a number of requests and is backed by past performance database and caching techniques. In caching technique the most frequent request is processed and the past performance database is responsible to tackle all the requests coming during peak hours. Peak hour requests management is one of the prime goal in cloud computing and is responsible for major Service level agreement violations. Therefore in this paper we have tried to minimize the violations and increase the response time by utilizing the past performance and put the obtained values in to cache.

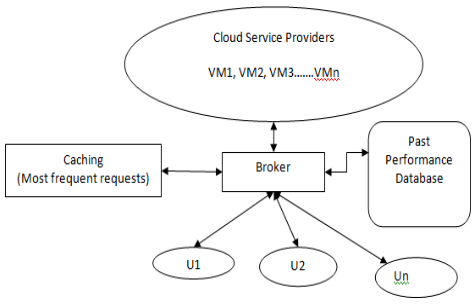
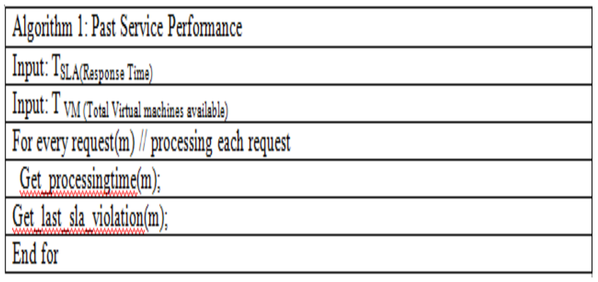


Figure 1: System Model



```
Algorithm 1: Past Service Performance
Input: $T_{SLA(Response\ Time)}$
Input: $T_{VM\ (Total\ Virtual\ machines\ available)}$
For every request(m) // processing each request
 Get_processingtime(m);
Get_last_sla_violation(m);
End for
```

Figure 2: Algorithm for past service performance

4. EXPERIMENTAL SETUP AND RESULTS

In this experimental set we have implemented 5 datacenter, 20 virtual machines of different sizes. Further we assume each data center consists of 2 hosts each. In the experiment we can evaluate the performance of each algorithm with a set of random requests generated by random numbers.

We have implemented five virtual machines and assumes that each virtual machine is holding a service i.e. each service has full virtual machine resources. Further each service 1 to service 3 has implemented generic algorithm which results in high SLA violations.

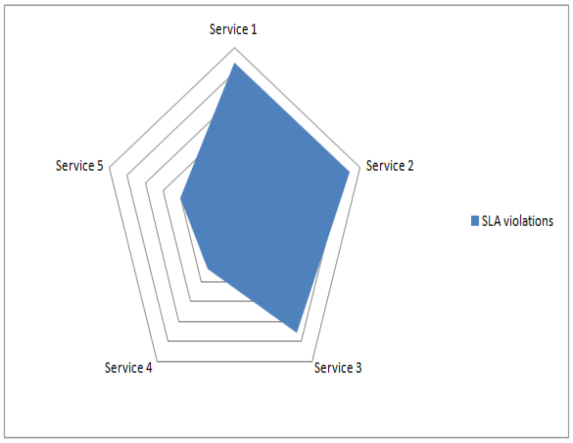


Figure 3: SLA Violation Tendencies

Also the response time of the services which has implemented the cache and in use of past service performance results in decrease in response time.

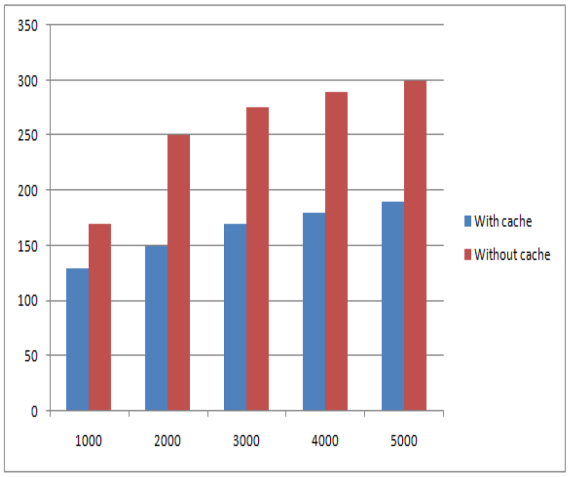


Figure 4: Processing Time Comparison

5. CONCLUSION

In this paper we have implemented past performance and cache the frequent requests and observe that it cause a huge impact to reduce the processing time of the system. In future work we will try to improve the response time by implementing dual caching.

REFERENCES

[1] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa, “Science clouds: Early experiences in cloud computing for scientific applications,” 2008.

[2] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. Berman, and P. Maechling, “Scientific workflow applications on amazon ec2,” in *E-Science Workshops, 2009 5th IEEE International Conference on*, December 2009, pp. 59–66.

[3] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, “On the use of cloud computing for scientific workflows,” in *eScience, 2008. eScience ’08. IEEE Fourth International Conference on*, 2008, pp. 640 –645.

[4] A. Luckow and et al., “Distributed replica-exchange simulations on production environments using saga and migol,” in *IEEE Fourth International Conference on eScience, 2008*, December 2008, pp. 253–260.

[5] C. J. Woods and et al., “Grid computing and biomolecular simulation,” *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 363, pp. 2017–2035, 2009.

[6] A. Natrajan, M. Crowley, N. Wilkins-Diehr, M. Humphrey, A. Fox, A. Grimshaw, and I. Brooks, C.L., “Studying protein folding on the grid: Experiences using charmm on npaci resources under legion,” in *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, 2001, pp. 14–21.

[7] S. Pronk and et al., “Copernicus: A new paradigm for parallel adaptive molecular dynamics,” in *Proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, November 2011.