IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

Load Balancing using Amazon Cloud Services

Prof. Vidya Chitre[1], Kaustubh Katkar[1], Shriraj Bamne[1], Vishwanath Prasad[1]

Department of Information Technology, Vidylankar Institute of Technology, Mumbai[1]

Abstract: One of the most appealing factors about any distributed task execution system is its processing speed to process the large amounts of data presented by incoming inputs. With increasing scope and popularity of Cloud services in Industrial applications it becomes important to achieve speeds over the Cloud which should be equal to or, if possible, faster than the existing traditional computing structure. Today, most of the existing frameworks utilizing Cloud services are based on Master/Slave architecture. The limitations associated with these Master/Slave architectures include lack of scalability and a single point of failure, the point of vulnerability being the Master. Our proposed system, LoBUCS, will provide better scalability and fast processing of HPC [2] and MTC [2] workloads using a fully parallel Client-Worker based structure eliminating the single point of failure. By utilization of Amazon Web Services (Amazon EC2 [5], DynamoDB [7] and SQS [6]) we will be substituting the complexity of existing distributed job management systems while maintaining good utilization.

Keywords: Cloud Services; parallel processing; distributed scheduling; LoBUCS.

I. INTRODUCTION

With LoBUCS we aim at achieving efficient job distribution driven by Client-Worker environment. This would be done by balancing the workload and using parallel processing for utilizing the computation power of workers.

To establish this efficiency, we will dynamically generate workers depending on the requirement of the workload [1]. Similarly, the framework will dynamically discard workers created for a previous larger job that would not be required for smaller jobs, thus maintaining efficiency in resource utilization. With the framework as proposed we will develop an application to hosted on the Amazon Cloud Servers that would accept images, videos and music clips as inputs and create a video slideshow as output on identifying these inputs. Then we will conduct a corresponding comparison with speeds in existing frameworks and determine the difference in performance with respect to the speed and management of incoming workload.

II. LITERATURE SURVEY

Traditionally, load balancing is being used to handle and control execution of user tasks and load i.e. users present online. With increase in load, organizations began to increase the number of servers required to manage it. But this increase proves costly.

Also, this increase in servers does not change the fact that the controller of the system or the main component that are these servers, leading to a single point of failure. Such a system will be costly to setup for an extreme level load, also to overcome single failure points, huge space would also be needed for backups of the data. One alternative was that, instead of buying the server components, an organization could setup a cloud, but it would need to keep an employee for its maintenance and handling. Instead of running such systems individually, we are proposing to use the complete job execution system over a public cloud which is highly optimized on cloud environment [1].

Our proposed system would end any dependency over a single server. Our idea is to create a complete system for load balancing over cloud and develop separate components for detection, allocation and clearance of user jobs. The aim of our system is to develop a lightweight distributed task execution framework that runs on instances created through Amazon Elastic Compute Cloud (EC2) [5] with the help of the building blocks of a distributed system such as Simple Queuing Service (SQS) [6] and the Amazon distributed NoSQL key/value store (DynamoDB) [7]. Through this we will achieve the goal of creating an application running on a compact and lightweight task execution framework [3] featuring fast processing speed and scalability. Load balancing and task execution will be achieved through SQS. Instead of assigning of jobs by the administrator component, such as a Master in Master/Slave architecture, we will implement pulling or work stealing algorithms on the worker nodes.

The proposed system will make complete use of cloud and virtual services which will help in cutting the cost massively as the hardware costs are significantly minimized.

The proposed system will let us achieve the following benefits:

1. HTC (High Throughput Computing) [2] applications can be executed with enhanced efficiency.
2. Processing of extremely large amounts of data without any interruption.
3. Any type of heavy job submissions can be handled by this framework.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

The proposed framework will then be used to host a job processing application. This application will take high quality images, video clips and music as an input and process them quickly to give a video slideshow with background music as an output.

III. TRADITIONAL SYSTEM

The traditional systems employed by the organizations have the following limitations:

1. Larger number of servers required.
2. Manpower required for system maintenance.
3. System upgrade is time consuming and costly.
4. Small and Medium businesses find buying and maintaining their own servers and systems too expensive.

IV. PROPOSED SYSTEM

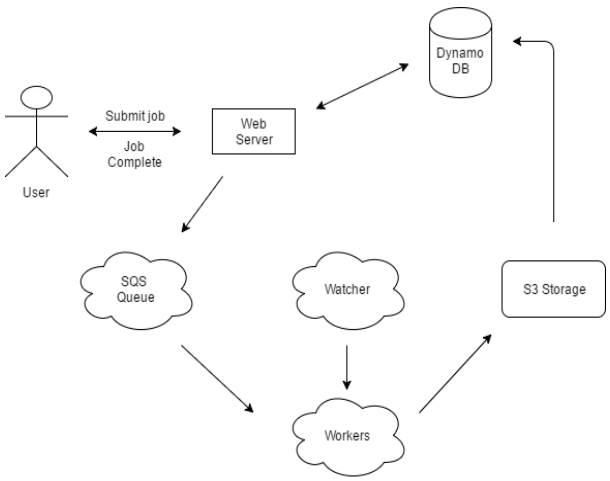

Figure 1: LoBUCS Architecture

1. Global Request Queue— The proposed system will accept user jobs which will then get queued in the Global Request Queue. This Global Request Queue behaves as a big pool for tasks to be executed. Client submits their jobs here and workers pull jobs from this pool.

2. Dynamic Provisioner— Input job would be allocated resources and its processing would be done on a virtual system i.e. on cloud. The resources required are provided by the workers with their computational power. Workers are created dynamically by Dynamic Provisioner making the system scalable.

3. Client Response Queue— To achieve parallelism, Client program requires is made to be multithreaded so that it can submit multiple tasks in parallel. The Client creates a response queue for itself before submitting any tasks. The tasks submitted by the client will carry the address of the client response queue. To reduce overhead in communication the client will have the option for task bundling.

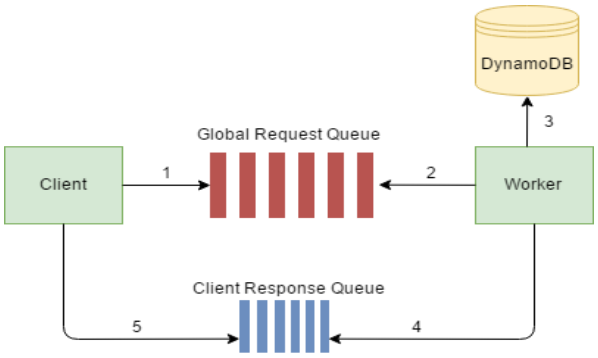

Figure 2: Communication Cost

4. DynamoDB— The status of the system is maintained by DynamoDB. This helps the workers to identify which tasks have been completed and which are yet to be executed. Utilization of each worker is reported periodically by a monitoring thread.

5. Distributing job entry, resource allocation, job pulling and processing will remove the single failure point by removing the dependency from the central component.

V. ADVANTAGES

1. To replace traditional master slave architecture.
2. To shift complete load balancing on cloud.
3. To remove the responsibility of maintaining servers.
4. To increase the efficiency of the work (MTC or HPC jobs).

VI. LIMITATIONS

1. Efficiency decreases when watcher fails or no worker is available.
2. System cannot work without Internet Service.
3. One of the major limitations is the inability of SQS to identify duplicates in input jobs [1].

VII. SOLUTIONS

1. Secondary watcher component will be developed so that it is able to replace the watcher in case of failure. Workers will be created dynamically by Dynamic Provisioner component.
2. A stable internet connection will be maintained during the implementation process.
3. We will make use of DynamoDB for verification of duplication. The worker thread will verify whether the is new or a duplicate. The worker thread then makes a conditional write to DynamoDB table which results in adding a unique identifier for the task which is a combination of Task ID and the Client ID.

VIII.CONCLUSION

Our system will allocate resources dynamically, therefore wastage of power is reduced. Also, the speed of this system

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

will be higher as compared to traditional master slave architecture based systems. Users will experience the faster processing of the system upon execution of heavy jobs.

REFERENCES

[1] Iman Sadooghi, Sandeep Palur, Ajay Anthony, Isha Kapur, Karthik Belagodu, Pankaj Purandare et al, "Achieving Efficient Distributed Scheduling with Message Queues in the Cloud for Many-Task Computing and High-Performance Computing," Illinois Institute of Technology, Chicago IL, USA, 2014.
[2] A. Rajendran, Ioan Raicu. "MATRIX: Many-Task Computing Execution Fabric for Extreme Scales," Department of Computer Science, Illinois Institute of Technology, MS Thesis, 2013.
[3] I. Raicu, et. al. "Falkon: A Fast and Light-weight tasK executiON Framework," IEEE/ACM SC 2007.
[4] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn. "Case study for running HPC applications in public clouds" In Proc. of ACM Symposium on High Performance Distributed Computing, 2010.
[5] Amazon Elastic Compute Cloud (Amazon EC2), Amazon Web Services, [online] 2016, http://aws.amazon.com/ec2/
[6] Amazon SQS, [online] 2016, http://aws.amazon.com/sqs/
[7] Amazon DynamoDB, Amazon Web Services, [online] 2016, http://aws.amazon.com/dynamodb.