CcssQoS: A load balancing mechanism for cloud serving systems

W. K. Hsieh
General Education Center, De Lin Institute of Technology, New Taipei, Taiwan

W. H. Hsieh
Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan

J. L. Chen & P. J. Yang
Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan

ABSTRACT: This study presents a novel QoS-aware load balancing mechanism called CcssQoS system for scheduling Virtual Machines (VMs) to leverage the capability of the Eucalyptus platform and improves the overall performance of cloud computing systems. Additionally, the Hadoop platform is integrated into the CcssQoS system to improve its ability to accommodate real-time services. The system performance is assessed in terms of CPU load, memory load and system throughput. Analytical results indicate that the proposed CcssQoS system can reduce both CPU load by 12.92% and memory load by 5.99%, eventually improving system throughput by 6.94% over those of the original Eucalyptus platform with Hadoop mechanisms.

Keywords: Virtual Machine; Eucalyptus; open cloud; Cloud Serving System; Quality of Service (CcssQoS); load balancing system

1 INTRODUCTION

Cloud computing enables developers to deploy applications automatically during task allocation and storage distribution by using distributed computing technologies in numerous servers (Ahlgren et al., 2011). The cloud computing architecture is divided into three layers. The bottom layer, infrastructure as a Service (IaaS), has a service-oriented architecture (e.g., Amazon EC2/S3). The middle layer, Platform as a Service (PaaS), is a service platform that allows developers to deploy their own applications (e.g., Amazon Web Services and Google App Engine). The top layer, Software as a Service (SaaS), enables users to access services based on their requirements (e.g., Microsoft’s online update service).

By using virtual devices, cloud computing reduces server power costs and minimizes hardware costs (Xiao et al., 2013). Also, the enormous data scale enables users to access multiple data storage. Developers must design mechanisms that optimize the use of architectural and deployment paradigms to gain the maximum benefit from cloud computing (Jin et al., 2011).

Eucalyptus (Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems), an IaaS platform, is an open-source framework. The operations of VMs across heterogeneous physical resources can be executed and coordinated using this platform. However, the data transfer rate obviously decreases when VMs calculate large amounts of data on the Eucalyptus cloud computing platform, resulting in an unsatisfactory network QoS. Consequently, VMs encounter difficulty in exchanging data, causing overloading (Moshanee et al., 2013 & Wang et al., 2013). To solve the above problem, this study develops a novel load balancing system, capable of improving the performance of VMs. This Cloud Serving System, called CcssQoS, is devised by improving the Eucalyptus platform. In the CcssQoS system, agents are created to monitor the load balancing mechanism. Figure 1 schematically depicts the CcssQoS system architecture.

According to Figure 1, the lower level of the Eucalyptus platform comprises numerous VMs that contain dispersive data (Sempolinski et al., 2010). The competing capability of VM decreases when a large number of users access these services. To handle this problem, the proposed load balancing mechanism leverages the features of Hadoop frameworks. A Hadoop framework consists mainly of: the Hadoop Distributed File System (HDFS),

269
INTRODUCTION

Cloud computing enables developers to deploy applications automatically during task allocation and storage distribution by using distributed computing technologies in numerous servers (Ahlgren et al. 2011). The cloud computing architecture is divided into three layers. The bottom layer, Infrastructure as a Service (IaaS), has a service-oriented architecture (i.e., Amazon EC2/S3). The middle layer, Platform as a Service (PaaS), is a service platform that allows developers to deploy their own applications (i.e., Amazon Web Services and Google App Engine). The top layer, Software as a Service (SaaS), enables users to access services based on their requirements (i.e., Microsoft’s online update service).

By using virtual devices, cloud computing reduces server power costs and minimizes hardware costs (Xiao et al. 2013). Also, the enormous data scale enables users to access multiple data storage. Developers must design mechanisms that optimize the use of architectural and deployment paradigms to gain the maximum benefit from cloud computing (Jin et al. 2011).

Eucalyptus (Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems), an IaaS platform, is an open-source framework. The operations of VMs across heterogeneous physical resources can be executed and coordinated using this platform. However, the data transfer rate obviously decreases when VMs calculate large amounts of data on the Eucalyptus cloud computing platform, resulting in an unsatisfactory network QoS. Consequently, VMs encounter difficulty in exchanging data, causing overloading (Monhamed et al. 2013 & Wang et al. 2013). To solve the above problem, this study develops a novel load balancing system, capable of improving the performance of VMs. This Cloud Serving System, called CssQoS, is devised by improving the Eucalyptus platform.
mechanism leverages the features of Hadoop frameworks. A Hadoop framework consists mainly of: the Hadoop Distributed File System (HDFS), MapReduce and Hadoop Database (HBase) (Fagui et al. 2010). HDFS consists of a master system and slave systems. MapReduce is a simple distributed system for parallel-oriented computing. Both HDFS and MapReduce can handle large amounts of data.

2 BACKGROUND KNOWLEDGE

2.1 Cloud Computing Systems

The proposed CssQoS system is implemented using two cloud open platforms, Eucalyptus and Hadoop. These platforms are described as follows.

2.1.1 Eucalyptus Platform

The Eucalyptus platform architecture supports VMs that operate on the top of Xen hypervisor layer, as shown in Figure 2.

- Cloud Controller (CLC): CLC is responsible for managing underlying virtualized resources such as servers, storages and networks.
- Cluster Controller (CC): CC runs on the cluster front-end machine to collect VMs information, scheduling VMs execution on a specific Node Controller and managing the configuration of VMs’ network.
- Node Controller (NC): NC is a component executed on the bottom of cloud computing to perform the start-up, shutdown, cleanup, inspection and management of VMs.
- Storage Controller (Walrus): Walrus is responsible for putting, getting, deleting services, as well as setting access control policies.

2.2 Load Balancing

As a load balancing solution for Linux systems (Kansal et al. 2012), Linux Virtual Server (LVS) attempts to provide a basic framework to construct a high-performance, scalability and reliability server. Additionally, LVS offers advanced IP load balancing software (IPVS), application-level load balancing software (KTCPVS) and cluster management components.

2.2.1 Linux Virtual Server via Network Address Translation (LVS/NAT)

NAT function maps one group IP addresses to another set of local IP addresses. On the LVS/NAT workflow, when a user accesses the service in the server cluster, the request packet is destined for the external virtual IP address for the load balancer.

2.2.2 Linux Virtual Server via IP Tunneling (LVS/TUN)

While encapsulating IP packets within an IP datagram, IP tunneling can be built on the virtual server where the load balancer can tunnel the request packets to the different servers.

2.2.3 Linux Virtual Server via Direct Routing (LVS/DR)

The load balancer has an interface for configuration along with a virtual IP address, where the interface can route the packets to the chosen server directly.

3CssQoS SYSTEM

To improve the VMs services in Eucalyptus cloud computing, Figure 3 illustrates the proposed CssQoS system architecture, including Hadoop, a load balancer and an agent-based monitor. Hadoop MapReduce technology processes vast amounts of data in hardware clusters in a fault-tolerant, simple and reliable manner.

![Figure 3. Proposed CssQoS system architecture](image3.png)

Additionally, Figure 4 illustrates the proposed CssQoS system framework, including MapReduce Module, Distributed File System (DFS) Module, HBase Module and Load Balancing Module. This framework combines Hadoop technology to support large amounts of data operations and a load balancer to schedule VM resources.

![Figure 4. CssQoS system framework](image4.png)
3.1 Map Reduce Module

Figure 5 illustrates the MapReduce operation procedure. The client node provides MapReduce actions to the JobTracker node. The JobTracker node is responsible for reducing the action to optimize the scheduling tasks. The TaskTracker node performs the map/reduce operation, while the HDFS stores the events of map/reduce operation. By including JobTracker and TaskTracker functions that can analyze a large amount of data, the MapReduce Module provides optimal services to users.

![Figure 5. MapReduce operation procedure](image)

3.2 DFS Module

The proposed DFS Module is derived from Hadoop HDFS with Eucalyptus cloud computing. DFS consists of a NameNode and many DataNodes. NameNode manages namespace operations like closing, opening and renaming files, as well as determines the mapping of blocks to DataNodes. DataNode manages the storages attached to the node and provides read/write requests from users of the file system. DataNode also performs block creation and replication from NameNode.

3.3 HBase Module

This module combines the Hadoop HBase with Eucalyptus cloud computing, which consists mainly of HMaster and HRegion functions.

3.3.1 HMaster Server

In addition to assign regions to HRegion, HMaster manages HRegion loading. If a HRegion server fails, HMaster reassigns the work to another HRegion.

3.3.2 HRegion Server

HRegion handles users’ read/write requests, forms a list of regions from HMaster and transmits heartbeat messages to HMaster, which checks that HRegion is still alive. The HRegion server includes many HRegions, which consist of HStores. HStores consist of MemStore and StoreFile, which includes HFile. MemStore is a sorted memory buffer that stores the client data. When full, MemStore flushes out a StoreFile.

3.4 Load Balancing Module

This study proposes a novel load balancing module to distribute the system workload of VMs in Eucalyptus cloud computing. Figure 6 illustrates the functions of the proposed load balancing module. The module consists of two components: a load balancer and agent-based monitor. The load balancer supports three capabilities that are incorporated into three mechanisms: balance triggering, CssQoS scheduling and VM controlling. The balance triggering mechanism receives data of VMs from agent-based monitor. The CssQoS scheduling mechanism performs a load balancing method by analyzing the data of VMs. The VM control mechanism provides the balancing control of the VMs’ load based on the actions taken by the CssQoS scheduling.

![Figure 6. Functions of Load Balancing Module](image)

3.4.1 Load Balancer

The CssQoS scheduling capability captures the parameters and computes the parameters where the results are stored in the weighted table. The VM control module then distributes the loading of VMs, according to the value stored in the weighted table.

1) Balance Trigger: State parameters of VMs are obtained by the agent-based monitor. If unable to capture the data, the balance trigger module interprets that VM is dead. The balance trigger module then sends messages about the VM state to the CssQoS scheduling module. Parameter W is defined as the VM state. A situation in which W=0, implies that VM is running and its state is currently alive. A situation in which W=1 implies that the VM state is dead. Meanwhile, a situation in which W=2 implies that the VM state remains the same as the previous state.

2) CssQoS Scheduling: The CssQoS scheduling module uses parameters from the balance trigger module to schedule the VM loading. The load balancer captures VM data when the service begins. The CssQoS scheduling capability then generates the weighted table according to the VMs’ state that consists of six parameters. Table 1 lists the notations of the parameters. Figure 7 shows the flow chart of theCssQoS scheduling. Operations of the CssQoS scheduling are listed as follows.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mfe_cpu</td>
<td>%</td>
<td>The rate of the physical CPU time that the VM not used</td>
</tr>
<tr>
<td>Mmax_mem</td>
<td>%</td>
<td>The rate of memory that VM used</td>
</tr>
<tr>
<td>Mmax_msg/s</td>
<td>KB/s</td>
<td>The average rate of successful message delivery about receiving action</td>
</tr>
<tr>
<td>Mmax_msg/t</td>
<td>KB/s</td>
<td>The average rate of successful message delivery about transmission</td>
</tr>
<tr>
<td>Mpe</td>
<td>%</td>
<td>The packet loss ratio about transmission</td>
</tr>
</tbody>
</table>

Table 1. Attributes of weighted table

![Figure 7. Flow chart of CssQoS scheduling](image)
Step 1: The balance trigger module receives the agent-based monitor message containing W value. A situation in which W=1 is established implies that the VM current state as dead. Therefore, the load balancer does not need to retrieve data from the agent-based monitor.

Step 2: The balance trigger capability receives data from the agent-based monitor. The data is classified as Mcpu_idle, Mnmem_used, Mth_rx, Mth_tx, Mpl_rx and Mpl_tx.

Step 3: The CssQoS scheduling capability receives the parameters from the balance trigger and builds the VM table. The attributes of VM table are {VM- Hostname, Mcpu_idle, Mnmem_used, Mth_rx, Mth_tx, Mpl_rx, Mpl_tx}. The VMs hostname is allocated from the Eucalyptus Cluster Controller.

Step 4: The CssQoS scheduling capability sorts the weighted VMs according to the performance of the VMs state.

Step 5: The CssQoS scheduling capability generates the weighted table. The attributes of weighted table are {VM- Hostname, icpu_idle, immem_used, ith_rx, ith_tx, ipl_rx, ipl_tx}.

Step 6: The VM control capability receives the weighted table from the CssQoS scheduling capability and, then distributes the task to the VMs according to the weighted value.

3) VM Control: The tasks are distributed using weighted round-robin method. Figure 7 describes the scheduling of the VM control capability.

4 PERFORMANCE ANALYSIS

The proposed Eucalyptus QoS Mechanism (CssQoS system) is compared with Eucalyptus with Hadoop and, then, with Eucalyptus with Hadoop and the balancer module.

In this study, the performance of cloud computing services was evaluated using data sizes of 3.78MB (1 Chunk), 7.56MB (2 Chunks) and 11.34MB (3 Chunks) starting at 20 seconds. The performance of Eucalyptus was then compared with that of Hadoop built, Eucalyptus with Hadoop and the original balancer built in, and the CssQoS system by using CPU, memory and throughput. All experiments were repeated 100 times. Table 2 lists the average execution times of the three cases.

Table 2. Average execution time

<table>
<thead>
<tr>
<th>Case</th>
<th>Average Execution Time (Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Eucalyptus And Hadoop</td>
<td>118.23</td>
</tr>
<tr>
<td>With Original Hadoop Balancer (Improved Rate)</td>
<td>117.56 (+0.57%)</td>
</tr>
<tr>
<td>CssQoS System (Improved Rate)</td>
<td>116.32 (+1.62%)</td>
</tr>
</tbody>
</table>

Figure 8 shows the CPU loading. With an increasing amount of data, the CPU loading on CssQoS system resembles that performed on the other two systems. Table 3 shows the average CPU loading in these three cases. The CssQoS system increases the CPU ratios in that the loading gradually diminishes as the data size increases.

Table 3. Average CPU load

<table>
<thead>
<tr>
<th>Case</th>
<th>Average CPU Loading (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Eucalyptus And Hadoop</td>
<td>33.33 (+9.57%)</td>
</tr>
<tr>
<td>With Original Hadoop Balancer (Improved Rate)</td>
<td>32.1 (+12.92%)</td>
</tr>
<tr>
<td>CssQoS System (Improved Rate)</td>
<td>31.66 (+8.97%)</td>
</tr>
</tbody>
</table>

4) Step 1: The balance trigger module receives the agent-based monitor message containing W value. A situation in which W=1 is established implies that the VM current state as dead. Therefore, the load balancer does not need to retrieve data from the agent-based monitor.

2) Step 2: The balance trigger capability receives data from the agent-based monitor. The data is classified as Mcpu_idle, Mnmem_used, Mth_rx, Mth_tx, Mpl_rx and Mpl_tx.

3) Step 3: The CssQoS scheduling capability receives the parameters from the balance trigger and builds the VM table. The attributes of VM table are {VM- Hostname, Mcpu_idle, Mnmem_used, Mth_rx, Mth_tx, Mpl_rx, Mpl_tx}. The VMs hostname is allocated from the Eucalyptus Cluster Controller.

4) Step 4: The CssQoS scheduling capability sorts the weighted VMs according to the performance of the VMs state.

5) Step 5: The CssQoS scheduling capability generates the weighted table. The attributes of weighted table are {VM- Hostname, icpu_idle, immem_used, ith_rx, ith_tx, ipl_rx, ipl_tx}.

6) Step 6: The VM control capability receives the weighted table from the CssQoS scheduling capability and, then distributes the task to the VMs according to the weighted value.

Figure 7. Flowchart of CssQoS scheduling

3) VM Control: The tasks are distributed using weighted round-robin method. Figure 7 describes the scheduling of the VM control capability.

3.4.2 Agent-based Monitor

The agent-based monitor module consists of data collection, data analysis and data transmission modules. The operation of the module; its capabilities are described as follows.

1) Data Collection: The information of VMs states is collected, including their CPU, memory, disk space and bandwidth.

2) Data Analysis: The data analysis mechanism analyzes the collected data and updates the value of W, indicating whether the data differs from the previous data.

3) Data Transmission: The data transmission mechanism is responsible for transmitting the data of VMs to the load balancer.

The operation of the load balancing module is assigned to the load balancer when users want to access services from the Eucalyptus platform. The balance trigger mechanism of the load balancer then sends messages to the agent-based monitor to obtain the data of VMs. The data collection mechanism gathers the parameters and passes the data to the data analysis mechanism. The data analysis mechanism analyzes the data of VMs that has been transmitted to the data transmission mechanism of the agent-based monitor.

Next, the load balancer receives the data of VMs. If W does not equal 1, the CssQoS scheduling mechanism obtains the VM data and generates the weighted table to the VM control mechanism, which balances the VM’s loading, according to the values of the weighted table. The HDFS and HBase modules store the data created when the VMs perform tasks requested by users. Finally, the MapReduce module handles the tasks based on the weighted table and transmits the results of the tasks to the users.
Figure 8. CPU Load: (a) 3.78 MB; (b) 11.34 MB

Figure 9 illustrates the memory loading performance, while Table 4 shows the average memory loading in all three cases. With an increasing amount of data, memory loading with the CssQoS system is nearly the same as that with the other systems. This phenomenon may be caused by the memory size of VMs, which is limited to 256MB.

Table 4. Average memory load

<table>
<thead>
<tr>
<th>Case</th>
<th>Original Eucalyptus and Hadoop</th>
<th>With Original Hadoop Balancer (Improved Rate)</th>
<th>CssQoS System (Improved Rate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Chunk</td>
<td>195.04</td>
<td>186.85 (+4.19%)</td>
<td>182.25 (+5.09%)</td>
</tr>
<tr>
<td>2 Chunks</td>
<td>210.13</td>
<td>205.35 (+2.27%)</td>
<td>196.20 (+3.48%)</td>
</tr>
<tr>
<td>3 Chunks</td>
<td>223.12</td>
<td>218.22 (+2.19%)</td>
<td>215.67 (+3.33%)</td>
</tr>
</tbody>
</table>

Figure 9. Memory Load: (a) 3.78 MB; (b) 11.34 MB

Figure 10 illustrates the system throughput performance, while Table 5 shows the average system throughput in all three cases. With an increasing data size, the master node should allocate many data blocks to slave nodes. If the loading of slave nodes are too large, the load balancer uses the round-robin method, causing the CssQoS system throughput performance to become nearly the same as that of the other two systems.

Table 5. Average system throughput

<table>
<thead>
<tr>
<th>Case</th>
<th>Original Eucalyptus and Hadoop</th>
<th>With Original Hadoop Balancer (Improved Rate)</th>
<th>CssQoS System (Improved Rate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Chunk</td>
<td>65.4</td>
<td>67.23 (+2.79%)</td>
<td>69.95 (+6.94%)</td>
</tr>
<tr>
<td>2 Chunks</td>
<td>115.33</td>
<td>117.83 (+2.16%)</td>
<td>120.26 (+4.27%)</td>
</tr>
<tr>
<td>3 Chunks</td>
<td>142.25</td>
<td>144.02 (+1.24%)</td>
<td>145.47 (+2.26%)</td>
</tr>
</tbody>
</table>

5 CONCLUSIONS

This study has developed the CssQoS system, which incorporates a Eucalyptus cloud computing infrastructure, Hadoop HDFS, Hadoop MapReduce, Hadoop HBase, load balancer and agent-based monitor technologies. In the proposed system, load balancer computing parameters are retrieved, and the loads are distributed to Eucalyptus cloud computing nodes using the IPVS toolkit. Emulation results indicate that the CPU loading, memory loading and system throughput performance comparable to that of the original Eucalyptus system with Hadoop technology and the original Eucalyptus system with the original built-in Hadoop load balancer. Analytical results demonstrate that the proposedCssQoS system can reduce the loading of virtual machines and improve the performance of cloud computing systems.

REFERENCES


