ABSTRACT

Building a server cluster is one of the methods to provide high availability web server. However, not all of the servers in the cluster are utilized continuously. This condition leads to wasted energy consumed by unused servers. Therefore, to prevent excessive power consumption we need to optimize energy usage. This paper proposes an automated power optimization in high availability server cluster based on queuing model theory. The power is managed by allocating optimal power to each member in the cluster based on received workload. The proposed method could save power up to 70 Watt per checking period (25 seconds).

Keywords: power optimization, high availability, server cluster, queuing theoretic model.

1 INTRODUCTION

A server cluster consists of a set of computers that work together as an integrated computing resource [1]. Not only able to perform complex computing, server cluster also provides high availability services or applications. It means they have maximum availability level. In addition, the system can also cope with failures that are caused by planned or unplanned factors [2]. Cluster servers are widely used in industry or research field.

The example software that provides high availability services are Heartbeat [3] and Pacemaker [4]. Heartbeat is a cluster messaging services and Pacemaker is a cluster resource management. Heartbeat makes one server and the other one is able to detect each other presence. Cluster resource management gives us fail over capability. If there is one server dies, then resources can be diverted to other servers that are still alive.

Nevertheless, the performance of the server cluster that provides high availability has not been matched with the optimization of consumed power. In 2011, growth of clusters and data centers energy consumption costs in the United States is estimated at 7.4 billion dollars per year [5]. We would require power management to save money on a cluster.

Research in the field of power optimization begins in mobile devices environment to conserve battery [6]. However, it needs an adjustment on the power saving methods when we implement them on a cluster server. In [7], energy optimizations techniques are discussed with inter server consolidation and power level turnover on the server. The only resource considered in the study is storage device. Other power optimization techniques are the scheduling of high-performance computing applications between geographically distributed data centers [8]. In data centers that are not geographically distributed, queuing model theory is used to optimize the allocation of power on each server in a data center such that the achieved average response time is minimum [9]. This model proved to give good results in experiments with different scenarios and varying workloads. However, this model still utilizes a combination of hardware and expensive software in providing power allocation.

Before performing the optimization of power, we must calculate the power consumed by a server. Since the power measurement requires expensive type of server or a physical measuring instrument, we choose to use the power estimation techniques. Various researches to estimate power has been proposed. Power estimation in the entire system using performance counters on microprocessors is discussed in [10]. Access to the chipset, processor, memory, disk, and input/output (I/O) device recorded by the microprocessor and used to estimate power. Mantis is a power modeling in blade environment and Itanium servers [11]. Mantis calculate central processing unit (CPU) usage, access to memory, as well as the level of I/O on the hard disk and network interface. The collected data is then calculated by linear program to generate the power estimation formula.

Computing power without using physical measurement tool has also been done on a virtual machine environment [12]. The parameters considered are also still the same as the Mantis. Different from the research described previously, we utilize power estimation that takes into account
virtual memory utilization on the Linux operating system (swap). The calculation does not use a linear program but multiple linear regression method. This mechanism has been discussed in our previous work [13].

Another aspect in power optimization is the power restrictions or commonly stated as power capping. Methods of limiting the power use by embedding hardware on the server have been developed [14] and achieve precision measurements at 1 Watt power. This method was improved on [15] by increasing the precision to 0.1 Watt. Furthermore, technical limitations by using an embedded controller power is summarized, reviewed its relationship with the shift in power, and discussed in more detail in [16].

In principle, power capping can be done by setting the state on the components of a server that supports state regulation [17]. We prefer to cap the power by manipulating dynamic voltage and frequency scaling (DVFS) to run power restrictions in a more adaptive environment. This feature has been supported by almost all vendors of processors. DVFS has proven to achieve sufficiently low power consumption without sacrificing performance. The basic idea of DVFS techniques is to dynamically change the supply of voltage level to the processor while providing sufficient processor speed (frequency) to process the workload such that the computing time and limits of performance are still optimum [18].

Clusters of servers that provide services with high availability is pretty much researched and implemented. Large Hadron Collider (LHC) project, the highest energy particle accelerator and the largest one in the world, has a cluster consisting of more than 200 servers using Heartbeat and Pacemaker to ensure availability of services [19]. Power management, which coordinates several processes that already available on almost any type of processor.

2 RESEARCH METHOD

There are three main steps to optimize power in a high availability server cluster. They are power estimation, power capping, and power optimization as the main step of the proposed method. Every step will be described in detail below.

2.1 Power Estimation

We utilize power estimation technique that not only consider main parameters of utilization i.e. processor, memory, hard disk, and data transfer rate on network interface but also adds virtual memory usage [13]. This technique has successfully been implemented in the blade server environment and we employ it in the non-blade one. This method is described briefly as follows.

A daemon (background process) to record the utilization of each server component is run first. Furthermore, make sure clamp meter as voltage measuring tool certainly has been running well and can monitor server power consumption. Then, the benchmark is run to give the load to the processor, memory, hard disk, swap, and I/O network. Various benchmarks provided are intended to overload the server component in accordance with the real work of the server that processes various tasks.

Data from the daemon log file and clamp meter are synchronized so that we obtain component utilization and power consumed per minute. This data is used as input for multiple linear regression method. Multiple linear regression is used because it can represents several independent variables to be predictors in the better way [21]. This method generates equation that can be used to estimate the power from utilization of server components as follows:

\[
y = 59.0625 + 0.366407x_1 + 0.405598x_2 - 0.638004x_3 + 1.5529x_4 - 0.954271x_5
\]

whereas \(x_1, x_2, x_3, x_4, \) and \(x_5\) are utilization of processor, memory, hard disk, swap, and data transfer rate on the network, respectively.

2.2 Power Capping and Power-Frequency Relationship

Power capping is implemented by manipulating DVFS or so-called P-state. This feature has proven to achieve sufficiently low power consumption without sacrificing performance. For example, on
an Intel Core i3 there are 14 states in MHz unit: 2.926, 2.793, 2.660, 2.527, 2.394, 2.261, 2.128, 1.995, 1.862, 1.729, 1.596, 1.463, 1.330, and 1.197.

There are several modes in P-state setting such as performance, ondemand, conservative, powersave, and userspace. Performance mode puts the processor frequency at the maximum rate. The ondemand will dynamically adjust the rise and fall frequency based on the given workload. Conservative is similar to ondemand, but changes made in more subtle frequencies. As opposed to performance, powersave sets the frequency of the processor run at minimum levels. We will use userspace mode where the processor frequency can be set manually. All of this manipulation can be done with root privileges on the operating system.

Restrictions on the minimum power can be performed by adjusting the frequency of the processor at the lowest state. Whereas the maximum power limitation is implemented by adjusting the frequency of the processor at the highest state. Power capping technique proposed in this study can be run in the more adaptive environment because the features DVFS has been supported by almost all vendors of processors.

The main parameter for power optimization on the cluster is a relation between power and processor frequency. In this context, the maximum power allocation on each server is limited. To determine the relationship between power and frequency, we give workload on the cluster. One type of workload that is used is LINPACK [22].

LINPACK is given to the server continuously without any interval of time between the arrival of the first and subsequent loads. This ensures that the server get peak workload performance in accordance with the allocation of power delivered. The relation between power and processor frequency is expressed in a linear equation as follows:

\[ s = s_0 + \alpha(P - b), \]  

whereas \( s \) is the speed or server processor frequency (GHz), \( s_0 \) is fully loaded server processor frequency at power \( b \) (Watt), \( \alpha \) is coefficient of slope in power-frequency relationship (GHz/Watt), \( P \) is power allocation allocated (Watt), \( b \) is minimum power consumed by fully loaded server at allowed processor frequency range (Watt).

### 2.3 Power Optimization

Server cluster used consist of \( k \) servers. Workload, \( q \), is given to the server with the speed \( \lambda \). Each server receives the workload of \( q_k \). Processor carries out workload by Processor-Sharing (PS) scheduling algorithm. This algorithm is identical to the Round Robin algorithm with quantum or time slices that close to zero. Round Robin is a load sharing algorithms that are time-sharing. Furthermore, each server has a processor speed of \( s_b \) with given power allocation \( P_b \). Distribution of the load is carried by load balancer (LB). In the high availability architecture, LB should have a maximum availability. Therefore, we add another LB to handle workload distribution when there is a failure on the active LB. Our proposed server cluster is illustrated in Figure 1.

Queueing model theory will minimize the average response time with a certain power allocation given to the cluster. The workload is assumed to come into clusters based on the Poisson distribution. This study uses open-loop configuration. In this configuration, the load coming from outside the system and leave the system after the process is complete. For this type of configuration, queuing theory model used is shown in the Formula 3 [9].

\[ \text{if } \frac{2k}{b} \leq \alpha: \quad \lambda^{*}_{2m+1,n} = \beta \cdot \lambda^{*}_{2m+1,n+1,k} = 0 \]

\[ \text{if } \frac{2k}{b} > \alpha: \quad \left\{ \begin{array}{l}
\lambda^{*}_{12,...,n} = \alpha \cdot \lambda^{*}_{12,...,n} \\
\lambda^{*}_{12,...,n} = \beta \cdot \lambda^{*}_{12,...,n+1,m+1,k} \end{array} \right. \text{if } \lambda \leq \lambda_{low} \]

(3)

whereas \( c \) is maximum power consumed by fully loaded server at allowed processor frequency range (Watt), \( P^* \) is optimum power allocation, \( n \) is number of active server at power \( c \), \( m \) is number of active server at power \( b \), \( k \) is total number of server in the cluster, \( \lambda \) is workload arrival rate in server cluster, and \( \lambda_{low} = \alpha P^* \).

Power is measured with the Equation (1) that has been generated from the power estimation step. After getting input from the server processor speed and power consumed, the ratio of these two parameters were compared with coefficient \( \alpha \). We continue optimization process by looking at workload arrival rate (\( \lambda \)). Then, the system decides allocation of power to the server whether it is maximum power (\( c \)) or the minimum one (\( b \)).

Passwordless ssh (secure shell) is used to turn off the remote server. With passwordless ssh, ssh command can be directly followed by the halt command to shut down the server. As for turning on the server, we use WakeOnLan features that have been supported by all latest versions of motherboards. Passwordless ssh and WakeOnLan (WOL) must be configured first. Therefore, this power optimization daemon (background process) can run well. We employ daemon to automate all tasks implemented by the system. Power optimization daemon is installed on the load
balancer as well as decision-makers in turn on or off a server. Power optimization feature is run on the background of the operating system once every 25 seconds. Time of 25 seconds was chosen because it takes 25 seconds to turn on or off the server from booting on average.

For ease of calculation in implementation, the minimum power \( (\alpha) \) rounded up to 170 watts and maximum power \( (c) \) rounded up to 210 Watt. With 840 Watt power allocation, it can be calculated that \( n = 840/210 = 4 \) and \( m = 840/170 = 5 \). In other words, there are four servers running with maximum power and five servers are operating at minimum power. Allocation of total power on a server cluster based on queuing theory model should satisfy the following equation: \( P = m \times \alpha = n \times c \). This experiment is trying to find the proper allocation of resources for clusters with five servers in it. The constant \( \alpha \) is a threshold for decision making whether a server can be switched on or off.

### Table 1. Power consumed by each state

<table>
<thead>
<tr>
<th>State (MHz)</th>
<th>Test 1 (Watt)</th>
<th>Test 2 (Watt)</th>
<th>Test 3 (Watt)</th>
<th>Mean (Watt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.197</td>
<td>166.15</td>
<td>168.07</td>
<td>167.05</td>
<td>167.09</td>
</tr>
<tr>
<td>1.330</td>
<td>189.33</td>
<td>187.45</td>
<td>188.09</td>
<td>188.29</td>
</tr>
<tr>
<td>1.463</td>
<td>187.41</td>
<td>187.75</td>
<td>187.97</td>
<td>187.71</td>
</tr>
<tr>
<td>1.596</td>
<td>179.60</td>
<td>178.50</td>
<td>180.70</td>
<td>179.60</td>
</tr>
<tr>
<td>1.729</td>
<td>180.75</td>
<td>180.25</td>
<td>178.22</td>
<td>179.74</td>
</tr>
<tr>
<td>1.862</td>
<td>190.33</td>
<td>190.32</td>
<td>190.65</td>
<td>190.43</td>
</tr>
<tr>
<td>1.995</td>
<td>196.15</td>
<td>196.83</td>
<td>197.99</td>
<td>196.99</td>
</tr>
<tr>
<td>2.128</td>
<td>200.17</td>
<td>198.95</td>
<td>202.78</td>
<td>201.63</td>
</tr>
<tr>
<td>2.261</td>
<td>188.75</td>
<td>189.55</td>
<td>190.98</td>
<td>189.76</td>
</tr>
<tr>
<td>2.394</td>
<td>190.87</td>
<td>191.89</td>
<td>192.97</td>
<td>191.91</td>
</tr>
<tr>
<td>2.527</td>
<td>195.76</td>
<td>190.76</td>
<td>190.95</td>
<td>192.49</td>
</tr>
<tr>
<td>2.660</td>
<td>195.35</td>
<td>193.00</td>
<td>195.00</td>
<td>195.45</td>
</tr>
<tr>
<td>2.793</td>
<td>195.67</td>
<td>195.9</td>
<td>195.90</td>
<td>195.82</td>
</tr>
<tr>
<td>2.926</td>
<td>210.30</td>
<td>205.5</td>
<td>207.00</td>
<td>207.60</td>
</tr>
</tbody>
</table>
Power consumed on each server when processing workloads range from 50-70 Watts. With the optimization of this power, active server is adapted to accepted workload. From the experiments performed with httperf application, fifth server is turned on when each server in the cluster receives the workload of 7000 requests/second or more. With this mechanism, a cluster that contains five servers is able to save about 50 to 70 Watts per checking period (25 seconds). High availability features that have been configured on the load balancer can be checked by using the command crm_mon (cluster resource manager monitors) owned by Pacemaker.

Screenshot when one load balancer is active or inactive is illustrated in Figure 3. When a load balancer fails, its role will be directly handled by the second one. Fail over takes about 23 seconds. If a cluster server has been tested for one week, then the level of availability clusters, $A$, can be calculated as follows:

\[
A = \frac{MTBF}{MTBF + MTTR} \times 100
\]

\[
A = \frac{(7 \times 24 \times 60 \times 60)}{(7 \times 24 \times 60 \times 60 + 21)} \times 100 = 99.996197 \%.
\] (5)

whereas $A$ is availability, $MTBF$ is mean time between failure, and $MTTR$ is mean time to resolve.

<table>
<thead>
<tr>
<th>Workload (request/second)</th>
<th>Active server</th>
</tr>
</thead>
<tbody>
<tr>
<td>6969</td>
<td>4</td>
</tr>
<tr>
<td>6818</td>
<td>4</td>
</tr>
<tr>
<td>7048</td>
<td>5</td>
</tr>
<tr>
<td>6948</td>
<td>4</td>
</tr>
<tr>
<td>6963</td>
<td>4</td>
</tr>
<tr>
<td>7004</td>
<td>5</td>
</tr>
<tr>
<td>7122</td>
<td>5</td>
</tr>
<tr>
<td>7007</td>
<td>5</td>
</tr>
<tr>
<td>7026</td>
<td>5</td>
</tr>
<tr>
<td>6911</td>
<td>4</td>
</tr>
</tbody>
</table>

4 CONCLUSION

From the all steps of study that has been done, we can draw conclusion as follows. Power capping in each server in the cluster do not need to use expensive software or hardware because it can be implemented with state manipulation of processor providing features Dynamic Voltage and Frequency Scaling (DVFS). Power optimization in a high availability server cluster based on queuing theory models can be implemented in automated ways. Based on our experiment, the proposed method can save power up to 70 Watt per checking period (25 seconds).

There are several improvements that can be developed to increase the performance of proposed method in this paper. More accurate power capping should consider the hard disk state, memory, and network interface. Optimization of power is not only implemented on the processing server but also to load balancer server.

REFERENCE


