Load Forecasting Applied to Soft Real-time Web Clusters

Carlos Santana
Institute of Computing
Fluminense Federal University
Niterói, Brazil
csantana@ic.uff.br

J.C.B. Leite
Institute of Computing
Fluminense Federal University
Niterói, Brazil
julius@ic.uff.br

Daniel Mossé
Dept. of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
mosse@cs.pitt.edu

ABSTRACT
Dynamic configuration techniques such as DVFS (Dynamic Voltage and Frequency Scaling) and turning on/off computers are well known ways to promote energy consumption reduction in web server clusters. This paper demonstrates how the application of forecasting methods improves energy savings in a soft real-time application, and compares it with other energy aware methods. Instead of a synthetic workload, a real traffic pattern was used to make the experiments more realistic. Our system promotes energy reduction while maintaining user’s satisfaction with respect to deadlines being met. The results obtained show that prediction capabilities increase the QoS of the system, while maintaining or improving the energy savings over state-of-the-art power management mechanisms.

Categories and Subject Descriptors
C.5.5 [Servers]: Energy reduction, power management, load forecasting, experimentation

General Terms
Web Server Cluster, QoS, Power Management, Energy Consumption, Dynamic Configuration, Load Forecasting

1. INTRODUCTION
Energy consumption has been growing as a global concern. It is an economic question for the potential social and financial effects to a country’s economy. It is also an environmental issue because the most used energy sources also cause a large amount of pollution. Even with an increasing usage of renewable sources of energy (biomass, fuel cells, solar, wind etc.), fossil fuels remain the most utilized source of energy [5].

The increase in data center costs, energy consumption, and importance to society has been a remarkable fact. These costs account for about 25% of a company’s IT budget [11]. Not only the energy necessary to maintain the machines working is taken into consideration, but also the energy spent with cooling structures and devices. Features embedded in new architectures of processors (e.g., dynamic voltage and frequency scaling mechanisms for multicore chips) and the advent of the Green Data Centers concept started to lead the way to achieve more energy savings [15].

Aiming to improve the results of the existing state-of-the-art energy consumption reduction policies, this paper shows the effects of load prediction applied to power management in a soft real-time web server cluster. To make this effort application-independent and environment-agnostic, the policy described in this article is based on a generic metric, CPU utilization, while maintaining the deadline miss ratio as specified by the user. Thus, the benefits obtained using forecasting techniques in a energy management method could be extended to a large number of applications and environments. It should be clear that the energy economy scheme here proposed is attained while also meeting a probabilistic restriction on the deadlines of the requests sent to the web cluster.

To perform the evaluation of the methodology used in this work, a real cluster was built, and a trace based on a real application was executed for hours, to show the energy and QoS improvements obtained over standard techniques that do not have any forecasting capability.

2. RELATED WORK
Different techniques to reduce energy consumption on clustered environments have been developed over the recent years. Relying on hardware mechanisms such DVFS and dynamic configuration of the cluster mechanisms allows for much energy savings [2, 6]. However, the results obtained by these methods do not consider forecasting capabilities in their original proposals.

Predictive policies have been shown to avoid unnecessary configuration changes due to load fluctuations [8], and thus provide further energy reduction and better quality of service (QoS). Even in pervasive computing domains [13], prediction-based approaches help to improve the resource allocation decisions aimed at enhancing user’s experience.

The work presented in [3] analyzes a real workload, based on Microsoft MSN protocol. Characteristics such as load persistence and patterns (like seasonality and fluctuation) are included in their simulations, through the forecasting method called SPAR (Sparse Periodic Auto-Regression). The main difference between their approach and the one used in this paper consists in the forecast method used. In SPAR, significant assumptions were made on the load pattern, while...
in the work presented here we used a smoothing technique, namely Holt Linear method (HLM), due to its independence from the incoming load distribution.

3. SYSTEM MODEL

The context chosen to apply the proposed methodology is a web server cluster. The application software is based on the TPC-W benchmark [14], where soft real-time deadlines are specified for requests. Our model assumes that the average request execution time is known. The QoS metric adopted is the fraction of requests that complete by their deadlines.

The cluster model in this paper, shown in Figure 1, has four basic entities: the front-end (FE) node, the worker (back-end) nodes, a load generator, and a LabView-based power meter. The load generator issues requests to the FE, and the FE carries out load balancing, delivers commands to workers (e.g., set CPU frequency, turn on/off), collect cluster statistics (e.g., number of computers turned on, current load, utilization level), and applies the forecast method over load samples. The worker node is the entity responsible for request processing. The FE and the workers (heterogeneous) are connected through a separate network, implementing a typical 2-tier web architecture.

Figure 1: Cluster model

The FE architecture is typically adopted in data centers, because of its simple implementation of cluster-wide policies for energy reduction. If the FE uses global information about all the cluster machines, a more efficient state with respect to energy consumption could be reached [4]. In addition this organization favors load balancing decisions. To address the scalability problem, hardware FEs or hierarchical schemes with more than one FE are employed. The FE as a bottleneck is beyond the scope of this paper.

Apache allows for a fair load balancing among back-end servers, for any set of frequencies used in those servers. The nodes that are off are ignored in the load balancing and request distribution.

Although current processors provide a limited set of discrete frequencies, we assume and implemented the continuous version of processor speed setting. This is based on the principle that an intermediary frequency can be achieved by periodically switching between a higher and a lower one [7]. This scheme allows for a finer grain control on power management.

To achieve optimal cluster configuration for any load value, an optimization problem similar to that described in [1] is solved. This is done off-line and a table is generated for each value of load, so that it can be used for on-line control. See Table 1 in Section 4 for an example.

To forecast the future value of the cluster load, the forecasting method used is based on Holt’s Linear Method (HLM or just Holt), as presented in [10]. HLM was chosen because as a smoothing technique it makes no assumptions about the load distribution and because of its ease of implementation. Also, for the time scale load forecasting is done in this work, the traffic pattern encountered in web servers does not present seasonality (seasonality would be typically encountered on a weekly period).

To dynamically obtain the HLM smoothing parameters, we used a software package [16] with low overhead to solve an optimization based on a non-linear least squares algorithm, applied to the load data.

4. METHODOLOGY

Large load variations make it more difficult for any forecasting algorithm to predict the workload. The request pattern for the workload was obtained from the traces of the 1998 Soccer World Cup [9]. Using the period ranging from day 65 (June 29, 11:30 AM) to day 66 (June 30, 00:00 AM), a customized version of the httperf tool [12] reproduces the original request pattern (as shown in Figure 2). Note that scaling of the request rate in the trace is performed, to match the cluster’s maximum capacity. In this work, we focused on CPU-intensive requests, given that we were just interested in studying the effects of forecasting.

The interval chosen from the original workload has three important characteristics. The first one is that it can be issued to the cluster without trace compression, which may hide intermediate load values and states, making the trace unrealistic (in our experiments, we ran traces at 1x rate). The second reason is the abrupt changes observed close to the third, fifth, and ninth hours. Together with the up-and-down of the 4th hour, they impose interesting challenges to the forecasting mechanism. The last characteristic is the behavior of the load in the valley between the fifth and ninth hours, which does not have extremely steep nor flat slopes, while presenting some variations in load that will stress the power management methods proposed.

Figure 2: Workload pattern
While the load distribution is being performed, the FE counts the number of requests over a fixed period (Holt sampling interval). This accumulated value is passed to the Holt forecasting routine in order to calculate the future window load value.

In order to reduce the machines boot time, the suspend to RAM mechanism was employed. To guarantee that a configuration has enough time to stabilize, a minimum interval (settling_time in Table 2) between reconfigurations was adopted; it should be pointed out that HLM is called periodically to track the load, even if a reconfiguration is not performed.

The Holt forecast routine is able to predict more than one time window ahead, but the error in the forecast becomes bigger for each future window. Therefore, to have the smallest prediction error, we will just estimate the future load for the first window ahead.

The estimation of the future load provided by the Holt method (load_forecast) is used to obtain a variable called $H_{base}$ (input load value for the power optimization problem), which is the offered load to the system (in fact, it is a fraction of the maximum attained load by the cluster). In other words, it is obtained as follows: $H_{base} = \frac{\text{load\_forecast}}{\text{load\_cluster\_Max}}$. With this value, the system then searches the pre-computed optimization table for the new configuration to be applied to the cluster. As an example, a sample of the optimization table is showed in Table 1. This table has a granularity of 1000 for the load, and was computed assuming 5 machines (ampere, coulomb, hertz, joule, and ohm).

Because a very loaded system is more likely to have a worse QoS then a lightly loaded one, we heuristically define a parameter $\gamma$ to relate load, cluster utilization, and QoS. The general idea is to use $\gamma$ to force a more powerful configuration then that provided by the optimization method. This optimization is based on the load presented, and the way we relate QoS, utilization, and load is novel. The larger the value of $\gamma$, the more conservative the prediction method is. Our forecasting policy executes the algorithm described in Figure 3 to calculate the value of $\gamma$.

```
if current QoS is below the target QoS then 
  $\gamma = \min (\gamma_{max}, (\text{target qos}/\text{current qos}))$
else
  $\gamma = 1.0$
end if
$\gamma = \gamma / (\text{target cluster utilization})$
```

Figure 3: Gamma factor setting algorithm

The new cluster configuration will be produced consulting the variable table_line obtained from Equation 1 below. This equation takes into account the gamma value obtained from the algorithm described in Figure 3. The variable table_size represents the granularity of the normalized load.

$$\text{table\_line} = \text{ceil}(H_{base} \cdot \text{table\_size} \cdot \gamma)$$ (1)

A line in Table 1 indicates the configuration for each of the cluster machines. For instance, line 600 represents a load of 60%, and shows that the frequency for servers ampere, hertz and ohm should be set to 2000MHz, 1800MHz and 1971MHz, respectively. Note that a zero frequency value means that FE must shut down the corresponding machine.

## 5. RESULTS

This section of the paper presents some results of the proposed energy reduction policy versus three other policies: (i) the Performance Governor (PG), which is a Linux standard, with all machines at their maximum frequency; (ii) the OnDemand Governor (OG), which is a Linux power management policy that changes frequency and voltage of the CPU based on the load in the system (uses a threshold); and (iii) an Optimized On-Off (OOO) power management policy without the forecasting policy (and carrying out frequency and voltage scaling at the same time as on-off reconfigurations).

All the results are for the cluster in Figure 1, with the five server machines indicated in Table 1 and a front-end. Table 2 shows the parameters used for our experiments. Some of them have already been defined above. Additionally, $\text{param\_opt\_time}$ is the period between optimizations of the Holt smoothing parameters, and $\text{holt\_window\_time}$ is the time window used for the Holt forecasting. We experimented with several values for this time window, varying from 5s to 20s, and 5s was the one that the minimized QoS losses. All results shown below are for a time window of 5s.

### Table 2: Experiment variables set

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_cluster_load</td>
<td>highest load</td>
<td>670 req/s</td>
</tr>
<tr>
<td>$\gamma_{max}$</td>
<td>gamma max value</td>
<td>1.4</td>
</tr>
<tr>
<td>settling_time</td>
<td>reconfiguration lag</td>
<td>2 min</td>
</tr>
<tr>
<td>$\text{param_opt_time}$</td>
<td>optimization interval</td>
<td>5 min</td>
</tr>
<tr>
<td>$\text{holt_window_time}$</td>
<td>Holt sampling interval</td>
<td>5 s</td>
</tr>
<tr>
<td>$\text{req_deadline}$</td>
<td>request deadline</td>
<td>4 s</td>
</tr>
</tbody>
</table>

The first comparison is against OG and PG. This was done to show the potential of power-aware techniques against standard techniques. Figure 4 shows the amount of total power consumed by the cluster, as a function of time, for OG, PG, and HLM.

The total energy spent in the experiments was approximately 5,267Wh for PG, 4,730Wh for OG, and 2,168Wh for the Holt (HLM) method. This means an energy consumption reduction of 3,099Wh (from PG to HLM), or approximately 59%; the reduction from OG to HLM was 2,562Wh, or about 54%. Comparing the HLM policy with OOO, the results indicate an energy reduction of about 1.5%, given a total of 2,200Wh was obtained for the latter method. The power curve for OOO is not shown in Figure 4 because it is
too close to HLM in the used scale. As indicated by [1] and other published works, the turning on/off of machines is the greater source of energy savings. However, it should be emphasized that this difference is obtained against an already optimized solution (OOO).

In addition to the energy savings obtained by HLM, our forecasting policy shows much better results than OOO when it comes to trying to maintain the system QoS level closer to a target (95% in the experiment). Figure 5 presents the system QoS levels experienced by HLM and OOO over the entire experiment duration. At the beginning of the experiment (close to hour 2) and at the end (close to hour 10), the heavy losses shown by the OOO policy are explained by an overreaction to short load bursts, as observed from the experiment logs (this can only be observed in a much smaller time scale than that shown for Figure 5); our method, however, performs much better in this situation.

Near the fourth hour, when the load reaches (or is close to) its peak, an overload situation occurs: requests were accumulated by the workers servers for later processing. This queuing effect of delayed requests, and also the load fluctuation, causes several changes in configuration in the OOO scheme. Our method, as described previously, was able to deal with this situation in a much smoother manner, with few QoS losses. Remember that QoS losses are allowed (and usually inevitable) in soft real-time systems. It is worth to note that the PG scheme did not have any QoS misses, but at the expense of over provisioning the system, and spending more than twice as much energy than HLM.

The HLM policy had 30 QoS misses during the whole experiment. Taking into account the enormous number of requests (near 7.28 millions) during the 12 hours of the experiment, this number becomes unimportant. However, the concentration of QoS misses at some particular instants is worth commenting on (near hours 2 and 10). These are moments where, most of the time, the configuration in operation has just one server, that might not cope adequately with the offered load; in these particular points in time, short bursts occur. Our method, with its forecasting scheme, is able to filter out unwanted reconfigurations, and clearly outperforms OOO. This scenario is illustrated in Figure 6. In this figure, in the y axis we represent the actual configuration, through the first letter of the names of the machines in operation (e.g., O represents the ohm server on; O+A represents a configuration with ohm and ampere on). Note, for example, near to hour 2, that the servers in operation change in response to the load variation. The same graphic for OOO has a greater number of configuration changes.

The power consumption curves from Figure 4, when correlated with the configurations in Figure 6, clearly show the impact of dynamic system reconfigurations when there are changes in load. As one machine needs to be turned on, an expected power spike shows up in the curve. To better understand the benefits of our method, we computed the energy expended by each policy during the critical moments, when QoS is not being maintained at the requested user level. This is shown in Table 3. We have already seen the advantages of HLM when compared to the PG and OG Linux policies. In this table, additionally, we see that HLM also behaves better than OOO in those moments where there are steep changes in load.

Finally, Figure 7 shows the aggregate cluster frequency for the OG and HLM policies. The aggregate frequency is the sum of the frequencies of all servers that are on. Given that the consumed dynamic power for a processor is approximately proportional to the square of its frequency, it is interesting to note the impact of our policy for CPU intensive loads.
Table 3: Energy consumed during load surges (Wh)

<table>
<thead>
<tr>
<th>Policy</th>
<th>Period (hours)</th>
<th>1.7-2.4</th>
<th>3.6-5.5</th>
<th>10-10.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG</td>
<td>277</td>
<td>937</td>
<td>198</td>
<td></td>
</tr>
<tr>
<td>OG</td>
<td>241</td>
<td>885</td>
<td>173</td>
<td></td>
</tr>
<tr>
<td>OOO</td>
<td>74</td>
<td>728</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>HLM</td>
<td>70</td>
<td>716</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>HLM / PG gain (%)</td>
<td>74.7</td>
<td>24.1</td>
<td>74.2</td>
<td></td>
</tr>
<tr>
<td>HLM / OG gain (%)</td>
<td>71.0</td>
<td>19.1</td>
<td>70.5</td>
<td></td>
</tr>
<tr>
<td>HLM / OOO gain (%)</td>
<td>5.4</td>
<td>1.6</td>
<td>3.8</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7: HLM vs OG: aggregate cluster frequency

6. CONCLUSIONS

The work described here studies on-off and forecasting mechanisms in an implementation of a web server cluster application with soft real-time characteristics. It is important to emphasize that this is not a simulation work, as in many of the published articles in the area. We managed to relate a soft real-time metric (i.e., meeting deadlines to satisfy a probabilistic QoS contract), with the offered load and the cluster utilization, and thus proposed a power management technique that is application-transparent. This paper also showed the benefits of a load forecasting mechanism on a clustered environment.

As a future work, the testbed developed here will be further tested, introducing new workloads. Also, we plan to test more complex forecasting techniques (e.g., regression methods), to see if its added complexity will bring any benefit. Finally, after the promising results presented in this work, we are starting a thorough parameter sensitivity analysis, to evaluate their impact on our forecasting scheme.

7. ACKNOWLEDGMENTS

This work was supported in Brazil by CAPES and CNPq, and in the USA by NSF through grants ANI-0325353, CCF-0811295, and CCF-0803180.

8. REFERENCES


