Hybrid Web Server: Traffic Analysis and Prototype

Matthew Olson, Ken Christensen
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida, USA
mdolson@mail.usf.edu, christen@csee.usf.edu

SangHak Lee, Jungmee Yun
RFIDUSN Convergence Research Center
Korea Electronics Technology Institute
Seongnam, Republic of Korea
{shlee, yunjm}@keti.re.kr

***Abstract*—Web servers in small and medium enterprises (SME) consume a significant amount of energy. We consider how a hybrid SME web server based on two co-located platforms (one high performance and high power and the second low performance and low power) can be architected to appear as a single system image to clients. A prototype based on a Dell AMD Athlon x64-based PC and a Sheeva ARM-based plug computer is described and evaluated using the Apache ab benchmark. Using traffic traces from the KETI corporate web server, we show that for the majority of the time a low-performance platform can very likely meet the performance demands (during which time the high-power platform can sleep). We explore prediction of server load using network traffic analysis as a means to achieve a selection (or switching) policy between the two platforms. Using simple prediction methods, we show that we can achieve between 41% to 67% energy savings with minimal performance impact.**

Keywords-web; hybrid server; energy efficiency; traffic analysis

I. INTRODUCTION

Energy efficiency is a first-class design criterion for ICT equipment. We seek energy efficiency in servers, clients, and network equipment to reduce both operational costs and CO_2 emissions that harm the environment. It has been estimated that in 2006 the servers and data centers in the U.S. consumed about 61 TWh [8]. This was 1.5 percent of the total U.S. electricity consumption for a cost of about $4.5 billion (at a commercial electricity rate of $0.074 per kWh). This electricity use was estimated to be twice that of in 2000. The operational cost of a data center is dominated by electricity cost. Energy consumption of servers in data centers has been a subject of intense research. Previous work extends back to the early 2000s where powering up and down servers in clusters in response to demand was investigated [3]. In the past few years, virtualization has become the key technique for reducing energy use of servers in data centers through consolidation of services. Virtualization allows physical servers (in clusters) to be consolidated into fewer server computers in response to demand while maintaining a single image to clients. What has not been addressed is the energy use of servers in small and medium enterprises (SME) where servers are often single computers with dedicated services where clustering and virtualization methods do not readily apply.

Of the total estimated 61 TWh consumed by servers in 2006 [8], approximately 6% (equal to about $370 million per year) likely came from single-computer servers installed in an SME. We base this estimate on the total energy consumption of mid-range and high-end servers reported for 2006 in [8]. We assume that mid-range and high-end servers are typically found in SME where volume servers (the third category reported in [8]) are almost exclusively found in data centers. Reducing the energy use of these single-computer server installations has not been significantly addressed. In this paper, we address the energy use of these single-computer server installations. We believe that significant energy savings can be achieved in this so far neglected area. We recognize that there is a trend towards cloud computing where enterprises move their server hosting to data centers (the "cloud"), but we also believe that enterprises will continue to install and operate their own server computers for a variety of business and technical reasons.

ICT energy consumption in enterprises comes from clients (such as desktop and laptop PCs), network infrastructure (including access points, switches, and routers), and servers. Controlling PCs to reduce energy use is a mostly "solved problem" as seen by the many commercial offerings (such as Verdiem's Surveyor product) and ongoing work in proxying-based solutions (such as the Ecma 393 standard [9] and research at Microsoft Research [7] and elsewhere). A proxy is an "entity that maintains network presence for a sleeping higher-power host" [9]. The Ecma 393 standard addresses proxying capability for lower layers (including ARP). Research has addressed proxying for specific applications including P2P (for example, in [1]). We propose to reduce the energy use of a web server by developing a hybrid server (first proposed for large-scale clusters in [4]) whereby two co-located platforms (one high performance and high power and the second low performance and low power) can be architected to appear as a single website to an outside client. The low-power platform can be considered as a web server proxy for the high-power platform. The low-power platform is used during the periods of time that the request rate (demand) is low and thus allowing the high-power platform to sleep, hence energy is saved. The two key challenges to be addressed are the *mechanism* for switching between the two platforms and the *policy* for when to switch. An effective policy must address prediction of server demand.

The remainder of this paper is organized as follows. Section 2 contains an analysis of a one-month trace of an SME server at the Korea Electronics Technology Institute (KETI) and shows significant potential for energy savings. Section 3 describes a prototype SME hybrid server and its switching mechanism. Section 4 is a preliminary study of a switching policy and includes energy savings estimates. The final section is a summary and discussion of future work.

978-1-61284-928-7/11/$26.00 ©2011 IEEE

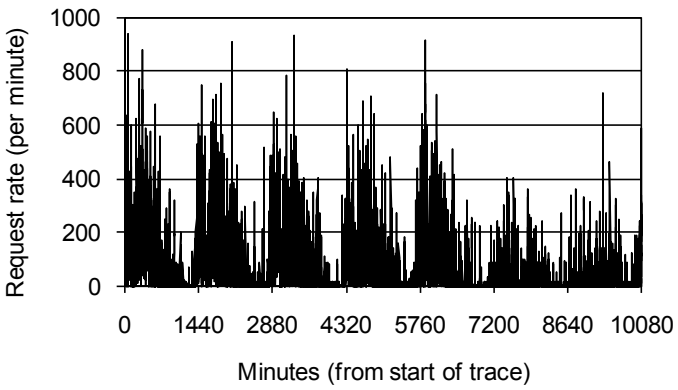

Figure 1. Daily variation in request rate to KETI server for one week

Table 1. Summary statistics for full KETI trace

Parameter	Measure
Mean time between all requests	0.81 sec
Mean time between ASP requests	16.81
Percent of requests that are ASP	4.8%
Mean request rate	74 hits / min
99% request rate	524
95% request rate	321
50% (median) request rate	10
5% request rate	1
1% request rate	1

TABLE 2. Percentage of one-minute intervals below threshold rate

	Request rate in requests per minute					
Trace	10	25	50	100	150	200
First week	53%	59%	63%	75%	82%	87%
Full month	50	56	61	74	82	87

II. ANALYSIS OF KETI TRAFFIC TRACES

It is generally understood that servers (and ICT equipment in general) operate at very low utilization levels most of the time. This is even true for data centers [2]. To gain an understanding of the fraction of time an SME server operates at a low utilization (which is also a low request rate), we analyzed a one month log of HTTP request traces from the KETI main server (found at http://www.keti.re.kr). KETI is a government research laboratory in Korea with two major tasks, "to help small and medium-sized businesses and ventures secure innovative technology, and to develop advanced technologies and create new venture projects" [5]. KETI has about 450 employees. KETI is itself a representative SME.

The one-month trace, taken in November 2010, captured and time-stamped all HTTP requests to the main KETI server. The server PC is running Microsoft IIS and serves both static and active (ASP) pages. Table 1 shows the summary statistics for the type of request, interarrival time between requests, and the request rate per minute for successive one minute intervals for the entire month. It can be seen that 1) active page requests are less than 5% of all requests, and 2) there is great variation in request rate (but, for over half the time the request rate is very low – 10 requests per minute or less). Figure 1 shows the daily variation in request rate for the first week of November, which started on a Monday. A diurnal cycle for demand can clearly be seen and also lower demand during the weekend.

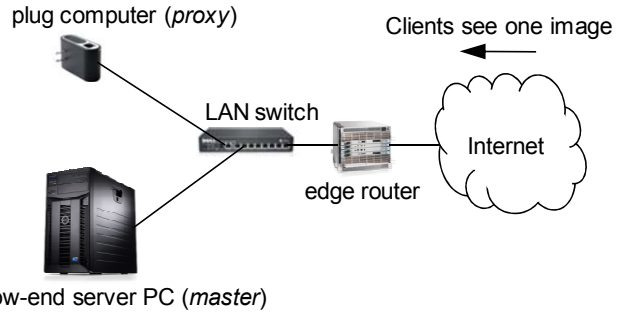

Figure 2. System configuration for hybrid web server

The analysis from the trace clearly shows that there are significant periods of time with low request rates. We assume that a low-power platform can maintain a given request rate lower than that of the high-power platform. Table 2 shows the percentage of one minute intervals for the first week and full month of November that are below a given threshold request rate. We assume that the high-power platform can sleep during these lower than threshold periods (and the low-power platform handles the requests). The percentage of time the high-power platform can sleep increases as the value of the threshold rate increases. We do not know exactly what request rate a low-power platform could support (however, we show some hints of what is possible in Section III of this paper), but we can assume that some lower rate is able to be supported. From Table 2 it can be seen that the total percentage of time for even a modest rate of 50 requests per minute is over 60%. This suggests that a low-power platform could "cover" (or proxy) for the high-power platform for a large percentage of time.

III. PROTOTYPE HYBRID WEB SERVER

The basic idea in our hybrid web server is to have two platforms that support the same web server software (for example, Apache) and are mirrored for content. The high-power platform can be considered to be the *master* and the low-power platform the *proxy*. Figure 2 shows the system configuration for our hybrid web server. The notion is that the proxy can be on for much of the time (while the high-power master platform sleeps) and do so with a sufficient level of performance so that clients do not perceive a decrease in performance (for example, a noticeable increase in response time). To be economical, the energy savings must be such that the additional cost of the proxy computer can be recovered in a "short" period of time (say, within one to two years).

We specifically explore the use of emerging extremely small plug computers for the proxy. For our prototype proxy we used a Sheeva plug computer manufactured by Marvel that cost less than $100 and contained a 1.2 GHz ARM processor, 512 MB SDRAM, 512 MB Flash, and USB and 1 Gb/s Ethernet connectivity (this is as of March 2011) [6]. The computer ran Ubuntu 9.04. The power consumption ranged from 2.3 W to 7.0 W depending on CPU load. Storage was connected via the USB port. We used a 16 GByte Flash drive for storage (at an approximate cost of $20 in March 2011). For our master we used a Dell AMD Athlon x-64 based PC running Ubuntu or Windows.

Step #1: Process requests until time to go to sleep
Step #2: Stop processing requests
Step #3: Change IP addr to different than site IP addr
Step #4: Unicast Magic Packet to partner
Step #5: Broadcast Gratuitous ARP
Step #6: Change IP address back to site IP addr
Step #7: Go to sleep
Step #8: Wait for Magic Packet to trigger wake-up

Figure 3. Script for switching between platforms

A. Dual-agent mechanism for switching between platforms

In our prototype, configured as shown in Figure 2, both platforms executed an identical agent that determined when to switch between the platforms. We call this a dual agent approach. At any given time, one platform is powered-up and serving and the other platform is sleeping (we note that for our prototype, the Sheeva plug computer was unable to sleep, but the Dell PC was able to sleep). Figure 3 shows the script used to switch platforms where the other platform is the "partner platform". Both platforms have the same IP address. A Magic Packet is used to wake-up the sleeping platform (step #4) and then a Gratuitous ARP packet changes the MAC-IP association in the edge router (step #5). To prevent an IP address conflict if both platforms are online at the same time, the script switches IP addresses during the wake-up period (step #3 and #6). Both platforms appeared to the outside as a single system image with a common site IP address. To implement the script of Figure 3 we used commonly available utilities to send a Magic Packet and Gratuitous ARP from a Python script. The decision when to switch platforms – that is, the switching policy – is addressed in Section IV. For our prototype we used a set time period as the switching criterion.

The rsync utility could be used to sync the two platforms. Syncing can occur during predetermined time periods during the day or during the switchover between the two platforms.

B. Handling of active pages

Handling of active pages, such as Microsoft ASP, is a special challenge for the hybrid server. An active page is an HTML page generated "on the fly" by the server processor. Typically, active pages are generated from content stored in a database (which may, or may not, be on the same platform as the web server). There are two options for handling active pages, the proxy platform can:

1) Run the same scripts as the master. That is, have the proxy generate active pages.
2) Serve stale copies of active pages.

Option (1) may not be possible in all cases – the proxy may not have the necessary processing capability and/or storage to generate active pages. We classify web sites as follows:

- Type 1: Realtime news sources that must always serve current content – stale content is unacceptable.
- Type 2: Non-realtime news sources that may on occasion serve emergency notifications – stale content may or may not be acceptable.
- Type 3: Other sites with no realtime content – stale content is very likely to be acceptable.

Table 3. Benchmark results for master (PC) and proxy (plug computer)

	Master		Proxy	
Type	Mean	95%	Mean	95%
Static 100 KB, 1	27 ms	27 ms	10 ms	10 ms
Static 100 KB, 10	11	21	10	10
Dynamic, 1	38	38	157	157
Dynamic, 10	38	59	167	181

For a Type 2 web site, wake-up policies may be possible whereby rare emergency notifications can trigger a wake-up of the master. This is an area for further work. Another area of future work is determining how active pages can best be generated for storage in the proxy (to be served as stale pages).

C. Initial benchmarking results for prototype system

Using the Apache Benchmark (ab) we evaluated the master and proxy for response time for both static and active pages as a function of the number of concurrent clients (1 and 10 clients, respectively). The static page was 100 KB of text, the dynamic page was an ASPX page which contained multiple database lookups, a date-time label, and an image. The master and proxy were both running Linux with Mono installed to support ASPX. Table 3 shows the results. The master and proxy have about the same performance for delivering static pages, but the master is faster for generating and delivering active pages.

IV. Switching Policy using Predicition

The proxy can cover for the master during periods of time when the request rate is below a given threshold. The statistics shown in Table 2 are the maximum possible sleep times (in percent of total time) for the master. What is needed is a means to predict if the next interval will be below a maximum request rate threshold and then initiate a sleep period for the master. A prediction scheme will have two types of errors:

- The scheme predicts a low request rate, but a high request rate occurs instead. The proxy will be overloaded; a performance penalty will have occurred.
- The scheme predicts a high request rate, but a low request rate occurs instead. The master will be awake when it could have been sleeping and energy is wasted.

As an initial experiment we used exponential smoothing as a means of predicting the request rate. That is,

$$r_{next} = \alpha \cdot r_{last} + (1-\alpha) \cdot r_{sample} \quad (1)$$

where r_{next} is the predicted request rate for the next interval based on the measured (sample) request rate, r_{sample}, of the current interval, and the last prediction r_{last}. This executes on interval boundaries and at the end of each interval a prediction is made for the next interval based on the computed r_{next} value. These predictions are then used to make a sleep or wake-up decision for the master and proxy.

A. Energy savings simulation

We simulated the exponential smoothing of Eq. (1) on the one-month KETI trace. Figure 4 shows the procedure used to simulate sleep and wake decisions on a processed trace file. The processed trace file was the KETI trace reduced to the number of requests per one minute interval (with 1440 such entries per day). The prediction procedure takes three variables

```
SleepSimulation(α, threshold, maxRequestRate)
1.  r_last ← r_sample ← 0
2.  sleepCount ← overCount ← 0
3.  for (all entries in processed trace file ) do
4.    r_next ← α· r_last + (1– α) · r_sample
5.    r_last ← r_next
6.    r_sample ← next entry in processed trace file
7.    if (r_next < threshold) then
8.      sleepCount ← sleepCount + 1
9.      if (r_next > maxRequestRate) then
10.       overCount ← overCount + 1
11.  return(sleepCount, overCount)
```

Figure 4. Procedure for prediction experiment

to be initialized, α, *threshold*, and *maxRequestRate*. The variable *maxRequestRate* is the maximum request rate that the low-power proxy can handle. The variables *threshold* (note that *threshold* < *maxRequestRate*) and α are "tuning parameters".

Table 4 shows the results for *maxRequestRate* = 50 and *maxRequestRate* = 100. In all cases, α = 0.25 and the values of *threshold* are shown in the second column. The variables *sleepCount* and *overCount* (in Figure 4) are used to calculate the percentage of sleep time and time where the proxy is performance limited (and clients may experience performance penalty such as an increased response time). It can be seen that even with a simple method of prediction, significant sleep times are possible, but with some impact to performance. The fourth column shows the performance impact as the percentage of one minute intervals with a request rate higher than *maxRequestRate*. Table 2 shows the maximum possible sleep (61% and 74% for *maxRequestRate* 50 and 100, respectively).

B. Calculating the payback period for adding the proxy

The additional cost of the proxy must be recovered from the energy savings achieved by allowing the master to sleep. Once the additional cost of the proxy is recovered, additional energy savings become an economic savings (a reduced operational expenditure). If we assume a $100 proxy cost, $0.10 per kWh of electricity (a standard rate in the US), 10 W power draw from the proxy and 200 W from the master, then 5263 hours of sleep time for the master are needed to recover the cost. For the KETI trace, such a payback would be achieved in about 1.2 years if we assume a sleep time of 50%.

V. SUMMARY AND FUTURE WORK

In this paper we have proposed and developed a simple hybrid web server to reduce energy use of single-computer SME web servers. A low-power platform is used to proxy for a sleeping high-power master platform during periods of low demand. Request level traces from a SME (in this case, the KETI main server) show significant periods of time where demand is very low suggesting that a proxy could meet performance requirements. Initial benchmarking results show that a plug computer can serve static pages as fast as a PC, but cannot deliver active pages as fast as a PC. Initial work using simple exponential smoothing for prediction of periods with low request rate was explored. We found that even with this very simple form of prediction, energy saving sleep times

Table 4. Results from prediction experiment

maxRequestRate	*threshold*	Sleep	Performance hit
50	10	41.5%	4.3%
50	25	51.0	6.7
50	40	56.0	8.6
100	25	51.0	2.6
100	50	59.5	4.3
100	75	67.4	6.6

ranged from about 41% to 67% for the one-month KETI trace (the exact value depending on the expected request rate that the proxy could actually handle).

This is only an initial work. Future work includes investigating:

- Mechanisms that are more robust and stable than the prototype dual agent approach for switching platforms.
- Additional benchmarking of the entire system.
- Policies (with prediction) that achieve closer to maximum possible energy savings than does simple exponential smoothing.
- Policies that consider the lifetime reduction cost of frequent powering up and down of computers as part of the equation to reduce overall operational costs.
- Deeper integration of a proxy with (or even within) a master for effective proxying of server protocols and applications at a lower cost.

Finally, we plan to develop and benchmark an actual system that hosts a real website (such as the KETI website).

ACKNOWLEDGMENT

This work supported by the Energy Efficiency & Resources of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Ministry of Knowledge Economy of the Korean government (2010T100100341, A Study on Energy Efficiency for Network Device).

REFERENCES

[1] Y. Agarwal, S. Hodges, J. Scott, R. Chandra, P. Bahl, and R. Gupta, "Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage," *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 365-380, April 2009.

[2] L. Barroso and U. Holzle, "The Case for Energy-Proportional Computing," *IEEE Computer*, pp. 33-37, December 2007.

[3] J. Chase and R. Doyle, "Balance of Power: Energy Management for Server Clusters," *Proceedings of the 8th Workshop on Hot Topics in Operating Systems*, May 2001.

[4] B.-G. Chun, G. Iannaccone, G. Iannaccone, R. Katz, G. Lee, and L. Niccolini, "An Energy Case for Hybrid Datacenters," *Proceedings of OSP Workshop on Power Aware Computing and Systems (HotPower '09)*, October 10, 2009.

[5] KETI (Korea Electronics Technology Institute), 2011. URL: http://www.keti.re.kr.

[6] "PlugComputer Community," 2011. URL: http://www.plugcomputer.org/.

[7] J. Reich, M. Goraczko, A. Kansal, and J. Padhye, "Sleepless in Seattle No Longer," *Proceedings of the USENIX Annual Technical Conference*, June 2010.

[8] "Report to Congress on Server and Data Center Energy Efficiency," Public Law 109-431, U.S. Environmental Protection Agency ENERGY STAR Program, August 2, 2007.

[9] "Standard ECMA-393 ProxZzzy for Sleeping Hosts," 1st edition, February 2010.