Power Saving for Web Servers Using Proxies

Karl J. O'Dwyer, Eoin Creedon, Mark Purcell & David Malone

***Abstract*—Electricity is a major cost in running a data centre, and servers are responsible for a significant percentage of the power consumption. With the rising cost of electricity and slow adoption of cleaner electricity-generating technology, servers should become more energy efficient.**

This paper looks at web servers, as HTTP is a common service provided by data centres. Reverse proxies are commonly used to improve the performance of web servers. In this paper, we consider how reverse proxies might be used to improve energy efficiency. We suggest that when demand on a server is low, it may be possible to switch off servers. In their absence, an embedded system with a small energy footprint could act as a reverse proxy serving commonly-requested content. When demand outstrips its capacity, the reverse proxy can power on the servers to meet this new load. Our initial results indicate that such a scheme could be practical and save significant power on servers with lower load.

I. Introduction

IT now contributes measurably to the global consumption of electricity. While this is still less than other sectors [1], there is room for improvement. New server technologies promise to lower power consumption when resources are idle. Servers are specified to meet or exceed current peak requirements, however common power saving techniques are unable to power the server off while idle, as the server must remain responsive to new requests. Empirical tests show that while powered off or sleeping, servers consume significantly less power than in the lowest-power idle states [2].

If we can overcome the adverse effect of powering off a system on availability, mechanisms for temporarily turning servers on and off do exist. Recent server hardware supports the Advanced Configuration and Power Interface (ACPI) power-saving modes, which were previously available on laptop and desktop computers. These include methods to Suspend to Disk (i.e. hibernation) and Suspend to RAM, which are low power modes with quick recovery to normal operation. The Wake-On-LAN (WoL) standard for remote activation of devices offers a convenient method to initiate recovery remotely.

The use of reverse proxies for website acceleration or load balancing is relatively commonplace. In this paper we consider using a low-power device acting as a reverse proxy. It will have the additional ability to shut down the server when demand is low, serve cached content while the server is sleeping and wake the server if new content is required. In contrast to previous papers, which have considered how to reduce power usage when a pool of servers provide a service (e.g. [2], [3]), we are targeting services typically provided by a single server which may have low-demand periods (e.g. in-house servers at night, low-demand hosted web servers, ...).

We explore this approach by developing a small testbed that allows us to replay web access patterns, estimate energy savings, etc. In Section II we describe the relevant power-management features and measure their power usage. In Section III we describe the web access patterns that we observe on a campus web server and their implications for designing a power-saving scheme. In Section IV we use a model of the idealised amount of power that can be saved in order to estimate how large power savings can be. Section VI discuss these results and the effectiveness of the scheme.

II. Power States and Recovery

Various systems are available for controlling the power state of servers. In particular, we will make use of an interface for putting the system into a low power state (ACPI) and then waking it at some later point (Wake-On-LAN).

ACPI is a standard that aims to consolidate all power management and configuration standards [4]. The ACPI Standard defines a number of Power States, from G0 (active) to G3 (mechanical off). The sleeping state, G1, is subdivided into 4 states (S1-S4). For us, S3 and S4 are interesting as they define Suspend to RAM and Suspend to Disk respectively. As we will see, in these states, servers consume almost as little power as when powered off. With operating system support, the power state of a server can be changed.

Wake-On-LAN is an industry standard for remotely powering on computers. It requires a compatible network interface card which remains powered after the computer is powered off. The signal used to wake up a computer is called the *Magic Packet*, a broadcast frame with a payload which is 6 bytes set to 255 followed by the target computer's MAC address repeated 16 times. WoL can usually be configured via the BIOS/firmware or operating system.

To illustrate these power-management features, we first investigate the power consumption of two devices which we use as servers (Dell® PowerEdge™1800 and Dell® Optiplex™755). We make power measurements using an off-the-shelf plug-in energy monitor (Maplin™200MU-UK). For these systems we also measure the time needed to put the system to sleep and the time to wake after a WoL message, which will influence the responsiveness of

978-3-901882-56-2 ©2013 IFIP

our scheme. We call the sum of these the *turn-around-time*. For comparison, we also show the power consumption of a Soekris net5501, which we will use as a reverse web proxy, during quiet periods.

Dell® PowerEdge™1800 This is typical of slightly older server-grade hardware, including Intel® Xeon® processors, RAID controllers, etc. While the power supply is rated up to 650 Watts, Table I shows that our configuration uses considerably less while idle. With firmware updates, the system does support both WoL and Suspend to Disk under Linux [1]. After investigation, we found that only relatively long turn-around-times are possible (over one minute). The non-zero power usage while the server is off is typical of modern PC hardware, and allows the support of WoL.

Dell® Optiplex™755 This is a desktop PC with more modern power control features and similar specifications to some low-end servers. These features make it more attractive to use in our tests. In particular, it supports Suspend to Ram as well as Suspend to Disk. Table I shows its power usage and turn-around-times. Both power usage and recovery times are lower than for the PowerEdge™1800, with the short turn-around time for suspend to ram being particularly attractive for our application.

Soekris net5501 The Soekris net5501 is a single board PC based around the AMD Geode™, using a max of 20W, but typically much lower as we see in Table I. We do not use or report on the sleep/wake features of the net5501, as we intend to use this device as the reverse proxy.

III. Investigation Of Traffic

Our aim is to exploit patterns in web traffic in order to turn off web servers when they are not required. There has been considerable work to characterise web traffic (e.g. [5], [6], [7], [8], [9]), and it is known that web access patterns are bursty.

A reverse proxy is a web server which accepts requests from clients and forwards them to a *back-end* server, which stores or generates all the web site's content available. The reverse proxy caches the content as it is served, and uses the cached content to answer requests where possible. As websites often have 'hot' content, or content that is expensive to generate but can be cached once generated, reverse proxies can often result in performance improvements. We use Varnish [10] as a reverse proxy. We know the reverse proxy's ability to cache content, and so to save power, will depend on the details of the accesses. Consequently, we will design and assess our scheme using actual requests from a campus web server.

This web server serves the websites of around 400 student clubs, societies and individuals. It has a variety of content and is accessed frequently by those on and off

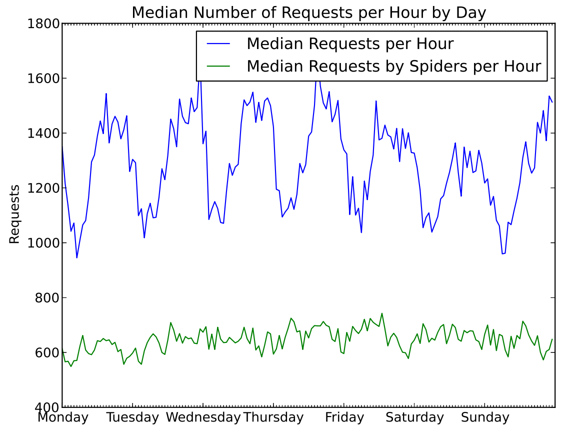

Fig. 1. The median number requests per hour by day, sample web logs.

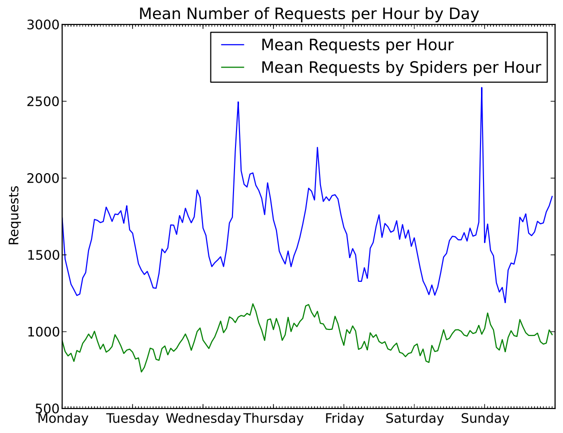

Fig. 2. The mean number requests per hour by day, sample web logs.

the campus, and exhibits a mix of web content and access patterns. In total it amounts to over 77GB of content in 400,000 files, excluding content stored in databases, though frequently accessed content represents a considerably smaller subset. The server is not busy, typically serving around 30,000 requests per day, but the volume of content should make caching more challenging.

We looked at the median and mean number of requests per hour over 270 days worth of data and grouped them by a specific hour during a week. The median is shown in Figure 1, with the noisier mean in Figure 2. We do not see periodic activity, but a diurnal pattern emerges, showing significant variation throughout the day.

Further inspection of the data reveals that a significant number of requests are from web spiders (e.g. search engines crawlers). Based on a manual inspection of the User Agent field of the log file, and consulting lists of common

[1]Linux Distribution: Ubuntu 12.04.2 LTS

	State	Power Usage (Watts)	Recovery from State	Time to Enter (mm:ss)	Time to Recover (mm:ss)
PowerEdge™1800	Power Off	8.6	Wake-On-LAN		1:14
	Hibernate (suspend-to-disk)	8.6	Wake-On-LAN	00:12	1:08
	Sleep (suspend-to-RAM)	NA	NA	NA	NA
	Power On (Idle)	109	NA	NA	NA
Optiplex™755	Power Off	1.3	Wake-On-LAN		1:33
	Hibernate (suspend-to-disk)	1.3	Wake-On-LAN	00:08	0:30
	Sleep (suspend-to-RAM)	2.5	Wake-On-LAN	00:03	0:05
	Power On (Idle)	60.3	NA	NA	NA
Soekris net 5501	Power On (Idle)	5	NA	NA	NA

TABLE I
MEASURED POWER PROFILE FOR DEVICES.

`Googlebot`	`Slurp`	`Baiduspider`
`bingbot`	`urlresolver`	`Speedy Spider`
`Sosospider`	`Sogou web spider`	`Gigabot`

TABLE II
STRINGS USED TO IDENTIFY COMMON SPIDERS.

spiders, we found that matching the list in Table II allowed us to identify the majority of requests made by spiders to this site. Random sampling suggests that of the requests that do not match this list, only $\sim 5\%$ come from spiders.

We calculated the median and mean request rate from spiders in this list and also show this median and mean in Figure 1 and Figure 2 respectively. We see that a significant number of requests are actually from these spiders and that this traffic does not exhibit the same diurnal pattern. This suggests that it may be useful to handle web traffic from spiders as a special case.

If we want to put a server to sleep between requests, then an important factor is the period between requests, which we estimate using the length of gaps between logged requests (logged at a resolution of 1 second). The length of these gaps will indicate if it may be possible to switch off the web server between requests, or if such opportunities are limited.

Using a subset of our data, amounting to 40305 requests over 28 hours, we look at the distribution of the gaps. Figure 3 shows the frequency of the gap of a particular duration. As we expect, gaps are typically quite short, which limits our chances to turn a server on and off without impacting on web traffic.

To consider the impact of caching on the gaps between requests to the (back-end) web server we replayed the requests to the campus server using Varnish[10] as a reverse proxy to cache the content[2]. The resulting distribution of gap lengths is shown in Figure 4. We see an increase in the number of long duration gaps, representing an increase in opportunities to put the server to sleep.

For comparison, we also consider the distribution of gaps between requests when we omit requests from spiders. We

[2]The cache starts empty. We use the default Varnish configuration.

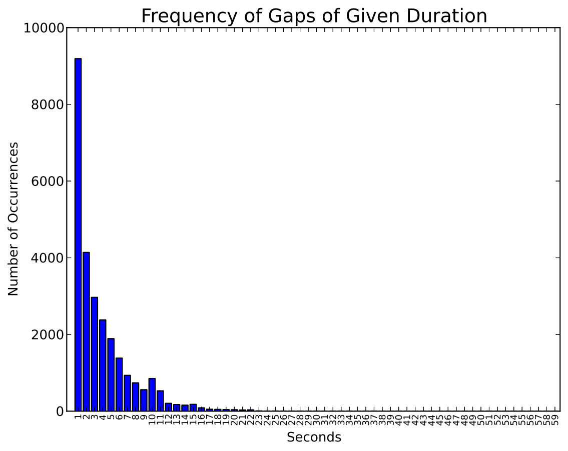

Fig. 3. Number of gaps of a particular duration between requests. We omit gaps of duration less than one second.

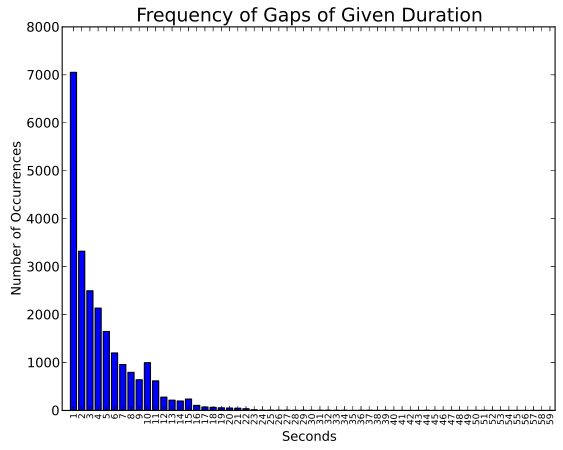

Fig. 4. Number of gaps of a particular duration between requests to back end omitting requests served by reverse proxy. We omit gaps of duration less than one second.

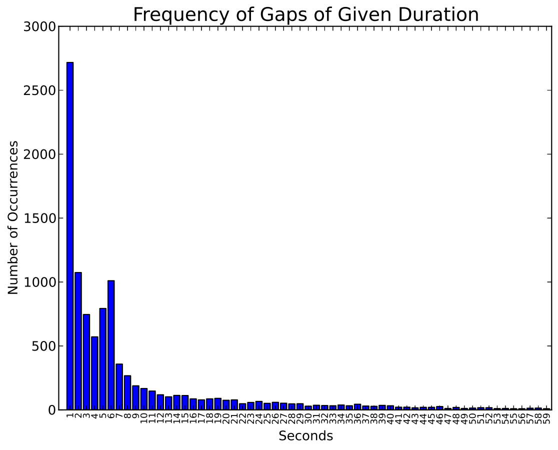

Fig. 5. Number of gaps of a particular duration between non-spider requests. We omit gaps of duration less than one second.

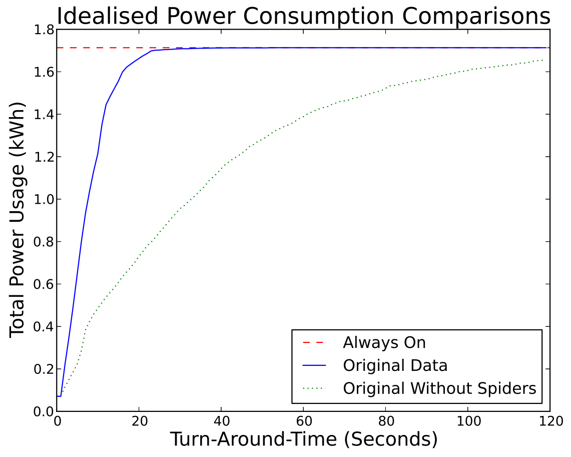

Fig. 6. Power saving for the idealised scheme that results in no delay, as a function of the turn-around-time.

do this on the basis that a spider does not usually need the content immediately, and indeed, some spiders can be told to make their requests later using a HTTP 503 response with a Retry-After header [11], [12]. The results are shown in Figure 5. We now see that the tail of the distribution of gap durations has thickened, which suggests we may have a better chance of powering the server off without impacting on user requests. Comparing these results with Figure 4, it appears that for this log file, smart handling of spiders might have a bigger impact than caching of commonly-accessed content.

IV. Idealised Power Saving Model

In this section, we show how to estimate the possible power savings for a web server, given information about power usage, gaps between requests and how quickly it can be turned on and off. Let us initially consider an idealised situation, where the server could somehow determine when the next request will arrive. After serving each request, it could see if the time to the next request is greater than its turn-around-time. If so, the server goes immediately to sleep and schedules a wake up just in time to serve the next request. This would allow the server to serve all requests without introducing any delays, while sleeping for the maximum time possible.

We can calculate the power saving given by this idealised scheme. Let $(t_i)_{i=1..N}$ be the sequence of gaps between requests. We can find T_o and T_s, the time that the server spends on or sleeping respectively. For this scheme,

$$T_o = \sum_{i=1, t_i < t_{tat}}^{N} t_i$$

and

$$T_s = \sum_{i=1, t_i \geq t_{tat}}^{N} t_i,$$

where t_{tat} is the turn-around-time. Total energy usage, in kWh, will then be

$$\frac{T_o P_o + T_s P_s}{3.6 \times 10^6},$$

where our measurements for P_o and P_s can be found in Table I for our hardware.

Using this model, we can assess the power saving possible without introducing extra delays. Figure 6 shows the total energy consumption possible as a function of the turn-around-time. We see that for the original log file, for a turn-around-time of 20s or more, there are few opportunities to sleep, and the total power usage is similar to having the server always on. However, turn-around-times of 2–3s would result in significant savings.

Following our observations from Section III we consider the impact of spiders and caching. Figure 6 also shows the results if we are willing to ignore requests from spiders. Here the situation is much more promising. With our idealised scheme, a turn-around-time of 20s allows a reduction of energy consumption to around 35% of the consumption for a server that is always on.

Figure 7 considers the impact of caching, by using the idealised power-saving model on the requests that are served from the back end, rather than from the content cached by the reverse proxy. As expected from Section III, we see a saving over answering all requests, however the improvement is not as large as the saving for ignoring requests from spiders. We also see that combining caching with the special handling of requests from spiders does result in useful gains.

V. Discussion

These results indicate that it is possible to save power by putting a high-power web server to sleep during low activity periods, where a low-power reverse proxy fields requests. We have also built a testbed where the reverse

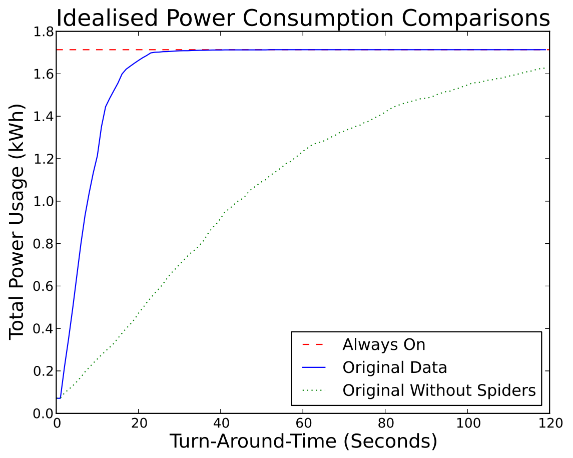

Fig. 7. Power saving for idealised scheme, answering requests that cannot be responded to by reverse proxy as a function of the turn-around-time.

proxy manages the power state of the web server via ACPI and WoL. In our system, the power-on and power-off decisions will be driven by the reverse proxy. We have modified Varnish to allow the cache to manage the power state of the server so that the impact of traffic patterns and caching can be assessed.

Our testbed also allows us to assess the performance impact of different power-on/off policies on the HTTP content served. Our initial results show that an on-off policy combined with a modest cache size and reasonably aggressive cache settings allows us make savings of tens of percent while serving content with small delays for the vast majority of requests.

Note, we believe that the low-power proxy will not cause performance problems during periods of high load. In practice, this can be achieved by directly serving requests (via DNS or IP mapping) or switching in a higher-power proxy at higher loads. Alternatively, smart high-performance network devices, such as IBM® PowerEN™[13] could be used as reverse proxies.

We have considered a situation with a single server hosted on physical hardware that can be powered down. A similar configuration with a higher-powered server should result in the scheme achieving larger savings. The principle of the scheme also generalises to other configurations. Multiple servers hosted on multiple physical systems could be cached by a single reverse proxy, spreading the cost of a higher-performance proxy over several web servers.

VI. Conclusion

In summary, we have explored the use of a low-powered reverse proxy to save power. We considered how a web server could be powered on and off, and what features of web traffic might be used to facilitate this, particularly the prevalence of spiders. We conducted tests to show that if a server is not too busy and can recover from a low-power state quickly (e.g. a turn around time of 10s) then significant power savings (e.g. 70%) might be possible. In future work, we will design a practical power-saving scheme and show its impact on HTTP performance.

Acknowledgements

This research was supported by HEA PRTLI Cycle 5 TGI and Science Foundation Ireland under Grant No. 07/SK/I1216a.

References

[1] L. Hilty, V. Coroama, M. de Eicker, T. Ruddy, and E. Müller, "The role of ICT in energy consumption and energy efficiency," *Report to the European Commission, DG INFSO, Project ICT ENSURE: European ICT Sustainability Research, Graz University*, 2009.

[2] A. Gandhi, M. Harchol-Balter, and M. A. Kozuch, "The case for sleep states in servers," in *Proceedings of the 4th Workshop on Power-Aware Computing and Systems*. ACM, 2011, p. 2.

[3] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and P. Doyle, "Managing energy and server resources in hosting centers," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 103–116, Oct. 2001. [Online]. Available: http://doi.acm.org/10.1145/502059.502045

[4] "Advanced configuration and power interface," http://www.acpi.info, Dec. 2011.

[5] H. Braun and k. claffy, "Web traffic characterization: an assessment of the impact of caching documents from the NCSA's web server," in *Second International World Wide Web (WWW) Conference '94*, Chicago, IL, Oct 1994.

[6] J. Pitkow, "Summary of WWW characterizations," *World Wide Web*, vol. 2, no. 1-2, pp. 3–13, Jan. 1999. [Online]. Available: http://dx.doi.org/10.1023/A:1019284202914

[7] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in web client access patterns — characteristics and caching implications," *World Wide Web*, vol. 2, pp. 15–28, 1999.

[8] L. Bent, M. Rabinovich, G. Voelker, and Z. Xiao, "Characterization of a large web site population with implications for content delivery," in *Proceedings of the 13th international conference on World Wide Web*, ser. WWW '04. New York, NY, USA: ACM, 2004, pp. 522–533. [Online]. Available: http://doi.acm.org/10.1145/988672.988743

[9] J. Cao, W. Cleveland, Y. Gao, K. Jeffay, F. Smith, and M. Weigle, "Stochastic models for generating synthetic HTTP source traffic," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 3, march 2004, pp. 1546 –1557 vol.3.

[10] P.-H. Kamp, "Varnish cache," 2013, [Online; accessed 11-January-2013, Version: varnish-3.0.1 revision 6152bf7]. [Online]. Available: https://www.varnish-cache.org

[11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "RFC 2616: Hypertext transfer protocol — HTTP/1.1," 1999.

[12] J. Jerkovic, *SEO Warrior: Essential techniques for Increasing Web Visibility*. O'Reilly Media, 2009.

[13] A. Golander, N. Greco, J. Xenidis, M. Hyland, B. Purcell, and D. Bernstein, "IBM's PowerEN developer cloud: Fertile ground for academic research," in *IEEE 26th Convention of Electrical and Electronics Engineers in Israel (IEEEi)*, 2010, pp. 803–807.