Slow Down or Sleep, that is the Question

Etienne Le Sueur and Gernot Heiser
NICTA and The University of New South Wales
{elesueur,gernot}@nicta.com.au

Abstract

Energy consumption has become a major concern for all computing systems, from servers in data-centres to mobile phones. Processor manufacturers have reacted to this by implementing power-management mechanisms in the hardware and researchers have investigated how operating systems can make use of those mechanisms to minimise energy consumption. Much of this research has focused on a single class of systems and compute-intensive workloads.

Missing is an examination of how much energy can actually be saved when running realistic workloads on different classes of systems. This paper compares the effects of using dynamic voltage and frequency scaling (DVFS) and sleep states on platforms using server, desktop and embedded processors. It also analyses workloads that represent real-world uses of those systems. In these circumstances, we find that usage of power-management mechanisms is not clear-cut, and that it is critical to analyse the system as a whole, including the workload, to determine whether using mechanisms such as DVFS will be effective at reducing energy consumption.

1 Introduction

Energy consumption, which was previously only a concern for mobile systems, has become important for all classes of systems. Processor manufacturers have implemented various mechanisms for managing energy consumption, some of which allow the operating system (OS) to trade performance against power. These are collectively known as *power-management* mechanisms.

Some mechanisms, such as dynamic voltage and frequency scaling (DVFS) and fine-grained clock-gating, are used while the processor is executing instructions, while other mechanisms are designed to reduce power consumption when the processor is not in use. The most common of these is the *sleep mode* or C state, which can be used to put the processor (or some part of it) into a low-power mode. Modern processors include several different C states which result in different power consumption and have different overheads.

These two types of power-management mechanisms present a trade-off to the OS designer: either run a task at a reduced CPU frequency (consuming less power) but remain active for a longer period of time (i.e. *slow down*), or, run a task at a high CPU frequency (with higher power draw) and as soon as possible enter a low-power idle state (an approach known as *race-to-halt* or *sleep*).

Prior research has focused on improving energy efficiency by using DVFS, resulting in a number of techniques that can be employed by the OS [6, 8, 10]. However, much of this research used workloads, especially SPEC CPU benchmarks, which are not representative of real-world system use. In addition, the methodology used in some of these studies unfairly biases the results toward those using high CPU frequencies—less static energy is consumed at a high frequency due to reduced task execution time. Consequently, the findings from this research must be interpreted with caution.

We recently presented an analysis of several server-class systems examining the effectiveness of DVFS at improving the energy efficiency of CPU-intensive workloads, such as the SPEC CPU workloads [5]. However, that work suffered from a common limitation in that it used unrealistic workloads. It also focused exclusively on high-end platforms. In this paper, we address these shortcomings by looking at workloads that are more representative of real system use, such as multimedia decoding/playback and serving of web pages. These workloads exhibit frequent, short idle periods (or bursty behaviour) which allows the trade-offs presented by DVFS and multiple C states to be examined more thoroughly.

We also look at a wider range of platforms in order to better understand the technology trends that we identified in our previous publication. Specifically we look at a desktop-class system based on an Intel Core i7 870 processor (the Dell Vostro 430s) and two platforms built on the most popular high-end low-power processor micro-architectures—the Intel Atom Z550 (the fitPC2) and the ARM Cortex A9-based Texas Instruments OMAP4430 (the Pandaboard). The specifications of these systems are provided in Table 1.

The rest of this paper is structured as follows. Section 2 discusses related work and Section 3 provides an overview of the power-management mechanisms that are available on the three platforms. Section 4 presents our experimental methodology. Section 5 then discusses the findings of our evaluation and Section 6 outlines our conclusions.

System	Dell Vostro 430s	fit-PC2	Pandaboard
Processor	**Intel Core i7 870**	**Intel Atom Z550**	**OMAP 4430 Cortex A9**
ISA	64-bit x86	32-bit x86	32-bit ARMv7
Class	Desktop	Embedded	Embedded
Cores / Threads	4 / 8	1 / 2	2 / 2
Frequency (GHz)	1.2–2.93, TB 3.6	0.8–2.0	0.3–1.008
Voltage (V)	0.65–1.40	0.75–1.1	0.93–1.35
Process	45 nm		
TDP	95 W	2.4 W	Unknown
L2 cache	4×256 KiB	512 KiB	1 MiB (shared)
L3 cache	8 MiB	-	-
Memory	4 GiB DDR3 1,333 MHz	1 GiB DDR2 533 MHz	512 MiB LPDDR2 800 MHz
Storage	500 GiB 3.5" SATA hard-drive	80 GiB SATA 2.5" hard-drive	64 GiB USB SSD 4 GiB SD
MPEG decode assist	None	PowerVR VXD	IVA-HD

Table 1: Specifications of the three processors and systems we analyse [1, 2]. Thermal Design Power (TDP) is the maximum processor power dissipation expected.

2 Related Work

Barroso and Hölzle presented the case for *energy-proportional computing* in 2007 [4]. Their analysis of several thousand servers in a Google data-centre showed that these servers spent most of their time significantly under-utilised. This is an interesting finding, and suggests that power-management research should be focused on workloads that result in this low level of system utilisation. In this paper, we intend to show how DVFS and sleep states can improve energy efficiency for under-utilised systems running real-world workloads, such as serving web pages.

A separate problem which has been tackled by many researchers is when to invoke DVFS and to what level. These decisions are often made by the operating system (OS). OS power management has been under active investigation since 1994 when Weiser et al. introduced the idea of changing the CPU frequency based on the system load [9]. This technique is now widely used, including in mainstream Linux.

More complex systems have been devised which make use of mathematical models to estimate the impact of reduced CPU frequency on the performance of a workload. Systems such as those proposed by Weissel and Bellosa [10] and Snowdon et al. [8] use hardware *performance counters* which are commonly available to parameterise models on which to base power-management decisions. Using these techniques, energy savings of up to 20 % were achieved on systems based processors such as the Pentium-M and PXA255. However, Snowdon et al. focused on using SPEC CPU workloads which do not cover a wide range of real-world use-cases. This limits what can be learnt from this work about the practical potential for energy management.

In 2002, Miyoshi et al. showed that the decrease in slack time resulting from running at a lower CPU frequency could offset any savings achieved by using DVFS [7]. By using a web-server workload similar to our own they found that it was more energy-efficient to run at a high frequency (i.e. *race-to-sleep*) for both high-utilisation and low-utilisation scenarios. However, the systems that were available 10 years ago are very different from those available today. Many low-power idle states are now available, and their usage results in significantly reduced power draw. Furthermore, *static power* is growing as a proportion of total system power, reducing the impact of DVFS on overall power draw.

This paper builds on previous work by looking at a wider range of workload classes and more recent systems with modern power-management mechanisms.

3 CPU Power-management Mechanisms

With the increasing importance of reduced energy consumption, it is not surprising that processor manufacturers have implemented an increasing number of power-management mechanisms. Two of these, DVFS and sleep states are described below.

3.1 DVFS

DVFS is a mechanism that exploits the relationship between the power consumption of a CMOS device, and the frequency at which it is clocked,

$$P = CfV^2 + P_{static}, \tag{1}$$

where C is the sum of capacitances within the circuit (which depends on transistor feature size), f is the operating frequency and V is the supply voltage. P_{static} represents power consumed from leakage mechanisms such as sub-threshold (weak-inversion), short-circuit and gate leakage. The voltage required for stable operation is determined by the frequency at which the circuit is clocked and can be reduced if the frequency is also reduced.

In the past, DVFS has been used to optimise the energy consumption of a system by reducing the CPU frequency

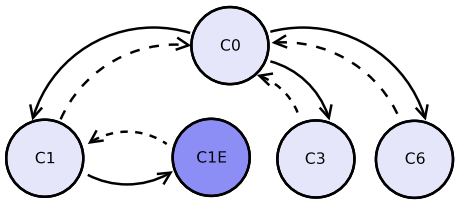

Figure 1: Possible C state transitions for the Core i7. All transitions must go through C0 except the special C1E state, which is used when all cores enter C1.

when it is determined that a lower-performing system would be acceptable. Unfortunately, static power is unaffected by frequency and a longer run time resulting from a lower clock rate increases the energy consumed through static power. Static power draw is increasing in modern systems and is a major factor contributing to the declining effectiveness of DVFS [5].

The Core i7 processor in the Dell Vostro also has a feature called *TurboBoost*, which increases the frequency of one or more cores if sibling cores are idle. The processor is sold as a 2.93 GHz model, however, the frequency of one or more cores can be increased up to 3.6 GHz (in steps) depending on the power requirements and heat dissipation of the processor. This can improve single-threaded workload performance. When and how much the frequency is increased is managed by firmware.

3.2 Idle states (C states)

Modern processors tend to offer multiple idle states often referred to as ACPI C states. These are denoted by Cx where x is a number from 0 to some maximum. Higher x values yield a *deeper* idle state, resulting in lower power consumption, but higher entry and exit latencies. C states are defined at the thread, core and package level, with $C0$ being the state in which a core is executing instructions. Beyond that there is no specification of C states, allowing manufacturers freedom to choose implementation techniques.

Because a single processor can have multiple cores and multiple hardware thread contexts (HyperThreads), constraints exist between thread, core and package idle states. There are also constraints on state transitions, as shown in Figure 1 for the Core i7 processor.

For the Core i7 and Atom Z550, Intel defines several C states which progressively apply more power-saving techniques in order to reduce power draw. For example, on the Core i7, entering C1 simply uses clock-gating to reduce processor activity, while entering C3 causes a core's local L2 cache to be flushed to the shared L3 cache and then powered down. When entering C6, a core's power supply is completely shut off, reducing leakage as well. To illustrate the impact of C state usage on system power draw, Figure 2 shows the *total* system power draw for the Dell Vostro 430s when all cores in the Core

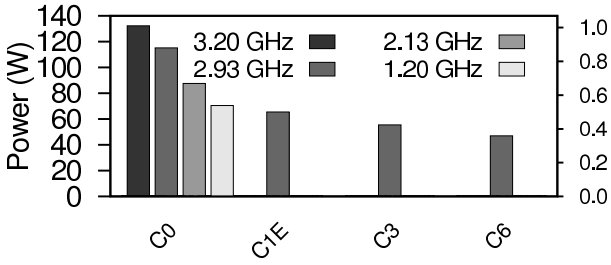

Figure 2: Idle power draw for the Dell Vostro 430s when all four cores of its Core i7 processor are placed in the same C state. Normalised scale on right-hand side.

i7 are placed in the same C state. C0 is essentially a tight idle loop, thus the power drawn in C0 is frequency dependent.

In contrast, ARM does not specify any C states for the OMAP 4430—it is left to the OS designer to choose (based on hardware constraints) what parts of the processor to power down and to what level.

Overhead from C state usage varies for a number of reasons. Firstly, in deeper C states, more power-reducing actions are taken, such as cache flushes (allowing caches to be powered down) and voltage/frequency changes. Flushing caches takes time as cache lines are written to backing stores, and voltage/frequency changes take time as voltage regulators settle and PLLs relock. Secondly, workload characteristics determine C state transition rates. A higher rate of transitions will cause higher overhead. Thirdly, the workload's memory access pattern will determine how much overhead results from cache flushes. Workloads which already have a high cache-miss rate will not be affected as much as workloads with a low miss-rate that still rely on the reuse of cached data.

4 Experimental Methodology

As discussed in Section 1, the SPEC CPU workloads that are commonly used for energy efficiency studies are not representative of the workloads that are run on most real systems. Real systems usually exhibit some level of idleness, allowing CPU sleep states to be used frequently. In contrast, the SPEC CPU workloads are CPU intensive, never allowing the CPU to idle during execution.

Therefore, we chose three real-world workloads that we believe cover a range of workload scenarios:

- MPEG playback using *mplayer* and *gstreamer*,
- serving web-pages using *Apache*, and
- the SPEC JBB2005 Java benchmark.

The first is a common workload for mobile and desktop systems and, being a single-threaded soft real-time task, it creates prolonged periods of idleness. The second

is a common multi-threaded server workload, which creates idleness due to the sporadic nature of web requests. Finally, the third is another multi-threaded server workload, but is written using Java and runs inside a Java virtual machine (JVM). These three workloads all allow the CPU's cores to enter idle states during their execution, thus allowing the effectiveness of both DVFS and different C states to be examined. Furthermore, these tasks all run for a fixed length of time regardless of CPU frequency. This means we can analyse the energy efficiency of each of them running at different CPU frequencies without having to account for static energy consumption (by padding) due to different execution times.

The three test systems (as shown in Table 1) are connected to their power sources through a power meter. As a result, all energy consumption data we report is for the total system.

We ran Ubuntu Linux (10.10) on each system, with kernel version 2.6.35 and use the *cpuidle* framework to measure C state transition rate. We used a custom cpuidle governor to allow us to choose the C state that would be used. We ran ten iterations of each benchmark, and averaged these results to obtain the final result. Standard deviation was less than 1% of the mean.

For the MPEG decode/playback workload, we decoded and played the first 60 seconds of an H.264 video on each of the platforms. We used a high-definition (HD) movie on the Core i7, and a lower resolution movie on the two embedded platforms, since due their reduced computational performance, they were unable to decode the HD movie on the applications processor without dropping frames and losing sync.

For the web-server workload, we used the ERTOS public website (http://www.ertos.nicta.com.au) as the web-server root, which contains a mix of static HTML pages and large PDF files. Tests were run over 10 minute intervals to allow for a warm-up period, allowing data to be brought from disk to the buffer-cache in memory. We used *Siege* [3] on two separate machines to generate load on the test system.

For the SPEC JBB2005 workload, we used from 1–4 warehouses. Throughput decreased as the number of warehouses was increased past four. Tests ranged from 0.5–4 minutes depending on the number of warehouses.

5 Results

In the following sections, we analyse the results from each of the three workload types. Graphs for some workloads and platforms are omitted for brevity.

5.1 MPEG playback workload

On the Core i7, decoding a high-definition (HD) MPEG stream resulted in very low CPU utilisation of between 7–17 % depending on CPU frequency. Due to the single-threaded nature of the MPEG decoding workload, only a single core could be utilised and all other cores could reside in a deep C state. The C state transition rate was approximately one transition every 10 ms, which is an order of magnitude lower than the web-server workload. As a result, negligible overhead was observed when deep C states were used, as shown in the top graph of Figure 3. There was a small anomaly in system load when using TurboBoost, as due to higher power draw in C0, there were fewer opportunities for TurboBoost to be invoked, resulting in slightly higher system load when C0 was used.

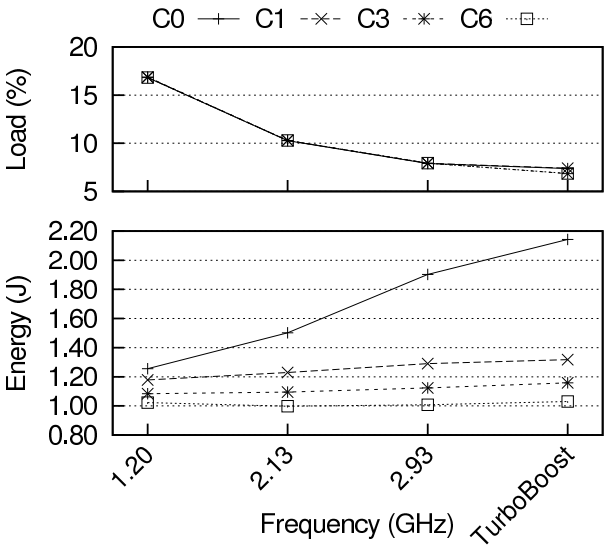

Figure 3: System load (top) and normalised energy consumption (bottom) when playing an HD-quality movie on the Vostro Desktop (Core i7) at several CPU frequencies with different C states.

As shown in the bottom graph of Figure 3, when the C0 C state was used, reducing the CPU frequency using DVFS on this platform resulted in significant energy savings. However, as deeper C states were used, DVFS became much less effective at reducing energy consumption. All data was normalised to the lowest observed energy consumption at 2.13 GHz using C6. No noticeable reduction in playback quality (dropped frames or loss of audio/video sync) was observed.

MPEG decoding and playback on the two embedded platforms was more interesting, as their processors have dedicated MPEG decode acceleration hardware (i.e. a DSP). Figure 4 shows energy consumption when several different power-management scenarios were used, including the Linux *ondemand* and *conservative* governors. Data is normalised to the maximum CPU frequency of each platform (circle points). As the bottom graph shows, using the DSP (as well as reduced CPU frequency) on the OMAP to decode the MPEG stream

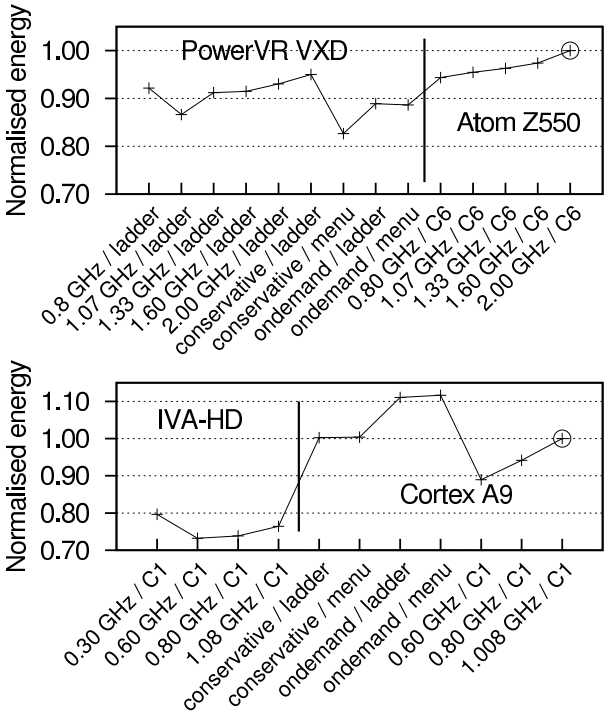

Figure 4: Normalised energy consumption of the fitPC (Atom, top) and the Pandaboard (OMAP, bottom). Points in the left-hand side are using the DSP to decode, points in the right-hand side are using the CPU.

resulted in energy savings of up to 25%, with no loss in playback quality. In fact, these DSPs are designed to decode HD-quality MPEG streams, which the CPU's on these platforms are incapable of doing in real-time—at 0.3 GHz on the OMAP, mplayer was forced to drop frames to maintain sync, thus we omit that point. Using the DSP on the Atom also resulted in significant energy savings of up to 18 % when combined with *conservative* and *menu* on the fitPC2.

5.2 Apache web-server workload

As Barroso and Hölzle found, Google's servers spend most of their time under-utilised [4]. Therefore, to test the effectiveness and impact of DVFS and C state usage on this workload, we used Siege to generate a request rate that resulted in an under-utilised system. The top graph in Figure 5 shows that CPU utilisation on the Dell Vostro was between 12–28% depending on the CPU frequency and C state used. We observed a higher rate of C state transitions—greater than once per millisecond, more than ten times as frequent as was observed with the MPEG playback workload. As a result, using the C3 and C6 C states caused slightly higher system load. Despite this, total system energy consumption was minimised at the lowest CPU frequency (1.20 GHz) using the deepest C state (C6) as shown in the middle graph. Additionally, we found that reducing the CPU frequency had no measured impact on either throughput or response latency. As a result, energy efficiency (in requests per Joule) was maximised at the lowest CPU frequency, using the deepest C state, as shown in the bottom graph of the figure.

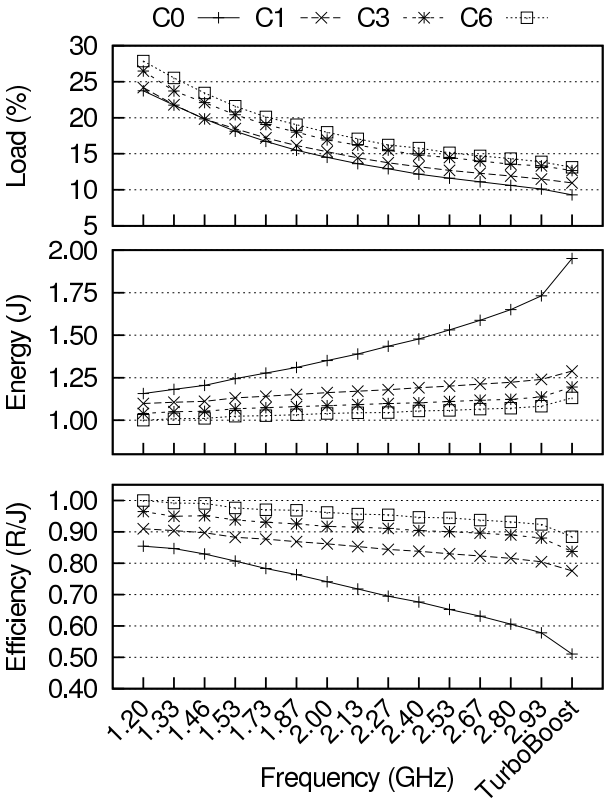

Figure 5: System load (top), normalised energy consumption (middle) and energy efficiency in requests per Joule (bottom) for Apache on the Dell Vostro (Core i7).

Similarly to the MPEG workload, we found that when deeper C states were used, the effectiveness of DVFS at improving energy efficiency was diminished. As C states improve in the future, this will become even more marked.

Using embedded-class systems in the data-centre is becoming a hot topic. Therefore, we also ran the Apache workload on the fitPC2 and Pandaboard. We found similar results to the Vostro Desktop. However, the throughput achieved on these systems was much lower than on the Vostro, and resulted in energy efficiency being significantly lower. Given that these platforms were not designed for this purpose—the Pandaboard uses a USB network interface—it is unclear whether low-power processors will have a significant impact on energy efficiency in the data-centre. Further investigation is required.

5.3 SPEC JBB2005 workload

The SPEC JBB2005 workload is a throughput-oriented benchmark. It stresses the CPU and memory

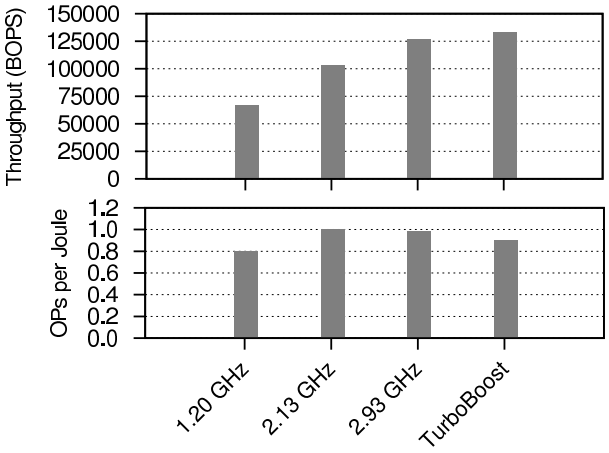

Figure 6: Throughput in billions of operations per second (top) and normalised energy efficiency in operations per Joule (bottom) of SPEC JBB2005 with four warehouses on the Vostro 430s using the C6 sleep state at several CPU frequencies.

hierarchy and is also designed to test the scalability of SMP systems. This resulted in a much higher level of CPU utilisation than the MPEG playback or web-server workloads. The total CPU utilisation was also dependent on the number of warehouses that were used. Using a single warehouse resulted in a system load of approximately 30 % spread over all cores. As the number of warehouses was increased, CPU utilisation also increased to a point where contention on other resources resulted in a bottleneck. The maximum CPU utilisation we observed with four or more warehouses on the quad-core Core i7 was about 90 %. CPU frequency had a negligible effect on the level of CPU utilisation, but throughput was impacted when CPU frequency was reduced, as shown in the top graph of Figure 6. We found that energy-efficiency (shown in the bottom graph) was maximised at 2.13 GHz using the C6 C state and four warehouses. C state transition frequency was similar to the MPEG workload, and, as a result, negligible overhead was observed when C6 was used.

6 Conclusions

We have extended previous work on energy-efficiency optimisation with DVFS by looking at realistic workloads on different classes of systems based on recent processors. Using these workloads rather than SPEC CPU benchmarks, we have shown that DVFS can still improve energy efficiency on systems that are under-utilised. This suggests that simple approaches to DVFS based on system load, like those taken by the Linux *ondemand* governor, should perform well. We also found that use of deep C states had only a small negative effect on performance, but significantly improved energy efficiency. The important factors to consider are: is the system is under-utilised and, will scaling the CPU frequency affect throughput or latency (QoS). For the MPEG playback and web-server workloads, we found that reducing CPU frequency and using the deep C states had no measurable impact on either, resulting in improved energy efficiency. This suggests that DVFS could be beneficial in the datacenter where servers must be provisioned based on the expected worst-case load, and therefore spend most of the time under-utilised. Further investigation is needed to confirm this finding for different web-server workloads, such as those that are highly dynamic and database driven. This is left as future work. We found that the SPEC JBB workload was different because CPU utilisation was independent of CPU frequency. This resulted in significantly lower throughput when CPU frequency was reduced.

From our analysis, it appears that system-level energy efficiency can be improved by both slowing the CPU down *and* using deep sleep states. However, the trends we previously identified [5] will continue, and, as a result, we will surely have to come back to this question in the future.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] Intel processor specifications, retrieved December 2010. http://ark.intel.com/.

[2] OMAP 4430 technical reference manual, Texas Instruments. http://focus.ti.com/lit/ml/swpt034a/swpt034a.pdf.

[3] Siege, an HTTP load testing and benchmarking utility. http://www.joedog.org/index/siege-home.

[4] Barroso, L. A., and Hölzle, U. The case for energy-proportional computing. *IEEE Comp. 40*, 12 (Dec 2007), 33–37.

[5] Le Sueur, E., and Heiser, G. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *2010 HotPower (HotPower'10)* (Vancouver, Canada, Oct 2010).

[6] Merkel, A., and Bellosa, F. Resource-conscious scheduling for energy efficiency on multicore processors. In *5th EuroSys Conf.* (Paris, France, Apr 2010).

[7] Miyoshi, A., Lefurgy, C., Hensbergen, E. V., Rajamony, R., and Rajkumar, R. Critical power slope: understanding the runtime effects of frequency scaling. In *16th Int. Conf. Supercomp.* (New York, NY, USA, Jun 2002), ACM Press, pp. 35–44.

[8] Snowdon, D. C., Le Sueur, E., Petters, S. M., and Heiser, G. Koala: A platform for OS-level power management. In *4th EuroSys Conf.* (Nuremberg, Germany, Apr 2009).

[9] Weiser, M., Welch, B., Demers, A. J., and Shenker, S. Scheduling for reduced CPU energy. In *1st OSDI* (Monterey, CA, USA, Nov 1994), pp. 13–23.

[10] Weissel, A., and Bellosa, F. Process cruise control—event-driven clock scaling for dynamic power management. In *CASES* (Grenoble, France, Oct 8–11 2002).