Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns

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

Abstract

Dynamic voltage and frequency scaling (DVFS) is a commonly-used power-management technique where the clock frequency of a processor is decreased to allow a corresponding reduction in the supply voltage. This reduces power consumption, which can lead to significant reduction in the energy required for a computation, particularly for memory-bound workloads.

However, recent developments in processor and memory technology have resulted in the saturation of processor clock frequencies, larger static power consumption, smaller dynamic power range and better idle/sleep modes. Each of these developments limit the potential energy savings resulting from DVFS. We analyse this trend by examining the potential of DVFS across three platforms with recent generations of AMD processors. We find that while DVFS is effective on the older platforms, it actually increases energy usage on the most recent platform, even for highly memory-bound workloads.

1 Introduction

Dynamic voltage and frequency scaling (DVFS) is a commonly-used technique to save power on a wide range of computing systems, from embedded, laptop and desktop systems to high-performance server-class systems.

DVFS is able to reduce the power consumption of a CMOS integrated circuit, such as a modern computer processor, by reducing the frequency at which it operates, as shown by

$$P = CfV^2 + P_{static} \quad (1)$$

where C is the capacitance of the transistor gates (which depends on feature size), f is the operating frequency and V is the supply voltage. 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. This can yield a significant reduction in power consumption because of the V^2 relationship shown above.

Previous research [6–8] has attempted to use DVFS to improve the energy efficiency of processors by analysing the way workloads use memory. In the past, a reduction in CPU frequency did not have a significant impact on the performance of workloads with a high miss ratio in the last-level cache (LLC). This is because the processor wastes a large proportion of cycles stalled waiting for the memory-subsystem to provide operands for instructions.

Unfortunately, the energy saving benefits of using DVFS are diminishing. In this paper, we will look at several factors influencing the effectiveness of DVFS including:

- scaling of silicon transistor technology;
- increasing memory performance;
- improved idle/sleep modes; and
- complexity of multi-core processors.

We analyse three generations of systems from the last seven years based on processors from the AMD Opteron family, specifically looking at how the above factors have influenced the energy saving opportunities presented by DVFS. We show how accounting for idle energy changes the perceived benefits of DVFS and how sleep/idle modes affect this.

Section 2 outlines some of the prior work that has attempted to leverage DVFS as a power management technique. Section 3 discusses our experimental methodology and we present our findings in Section 4. An analysis of the factors influencing the effectiveness of DVFS is given in Section 5 and in Section 6 we present our conclusions.

2 Related Work

Previous research has attempted to leverage DVFS as a means to improve energy efficiency by lowering the CPU frequency when cycles are being wasted, stalled on memory resources. Energy can only be saved if the power consumption is reduced enough to cover the extra time it takes to run the workload at the lower frequency. It is this trade-off which we will analyse in the next section.

Weiser et al. were the first to propose the use of DVFS to reduce the energy consumption of computer processors [7]. They used simulated execution traces and the

System	Compucon K1-1000D	Dell PowerEdge SC1435	HP ProLiant DL365 G5
CPU model	246	2216	2384
CPU die codename	**Sledge-hammer**	**Santa Rosa**	**Shanghai**
Year	2003	2006	2009
Core count	1	2	4
Frequency (GHz)	0.8 - 2.0	1.0 - 2.4	0.8 - 2.7
Voltage (V)	0.9 - 1.55	0.9 - 1.35	1.0 - 1.35
TDP	89 W	95 W	75 W
Feature size	130 nm	90 nm	45 nm
Die size	193 mm^2	230 mm^2	285 mm^2
Transistor count	106 M	243 M	463 M
L1 cache	64 KiB I & 64 KiB D per core		
L2 cache (per core)	1 MiB	1 MiB	512 KiB
L3 cache	-	-	6 MiB (shared)
Line-size	64 bytes at all levels		
Memory type	DDR 400 ECC	DDR2 667 ECC	
Memory size	512 MiB	4 GiB	8 GiB
DRAM channels	1	2	2
Prefetch distance (in cachelines)	2	2	3
System idle power	74 W	145 W	133 W

Table 1: Specifications of the three AMD Opteron processors and systems we analyse. All systems have a single CPU package [1]. Thermal design power (TDP) is the maximum processor power dissipation expected.

level of slack time to choose a new CPU frequency at each OS scheduler invocation. A similar approach is used by the Linux *ondemand* governor, which attempts to minimise idle time by changing CPU frequency in response to load.

When Weiser published in 1994, transistor feature sizes were approximately 0.8 μm and typical core voltages were 5 V. Furthermore, the ratio of dynamic power (which DVFS can reduce) to static leakage power was high. Therefore, energy savings resulting from DVFS could be significant. In contrast, today's CPUs are built from transistors with feature sizes of 32 nm and core voltages at the highest frequency are 1.1–1.4 V. The small feature sizes result in leakage power reaching or exceeding dynamic power, and the low core voltages reduce the voltage-scaling window (which is limited by the 0.7 V threshold voltage of silicon transistors). Therefore, the potential to save energy via DVFS is dramatically reduced.

Weissel and Bellosa developed models based on parameters such as *memory requests per cycle* and *instructions per cycle*, which can be counted using the performance-monitoring unit (PMU) available in most processors [8]. By predicting a workload's response to a change in frequency, a more energy-efficient frequency can be chosen at each scheduler invocation. By using this technique, they were able to save up to 15 % energy for some workloads on an Intel XScale processor.

Snowdon enhanced this approach by developing a technique to automatically choose the best model parameters from the hundreds of possible events that modern PMUs can measure [5]. His *Koala* framework was able to save up to 20 % energy on some memory-bound workloads running on a Pentium M processor [6].

In 2002, Miyoshi et al. analysed three systems and found that the energy consumed when idle must be accounted for if saving energy overall is the goal [4].

3 Methodology

Our experimental setup includes the three platforms described in Table 1, each running Linux 2.6.33, compiled with all required modules built in. We used an Extech 380801 AC power analyser to measure power consumption at the power supply. All tests were carried out inside a RAM backed temporary file system to ensure no disk activity.

We used SPEC CPU2000 [2] as our suite of test benchmarks, as the execution time of these workloads was reasonable on all of our test platforms. The same benchmark binaries were used on each platform, compiled with gcc 4.2 with high optimisation. We ran 10 iterations and then averaged the results to obtain a statistically sound outcome. Standard deviations were all less than 1 % of the mean.

The CPU2000 benchmark 181.mcf is an odd case. This benchmark is known as the '*cache-buster*' because of the memory access pattern that it generates, which results in a high miss ratio in all levels of cache. Because of this, the performance of 181.mcf is primarily dependent on memory speed, hence scaling the frequency of the CPU should only have a modest effect on its performance. We use 181.mcf as a *worst-case* memory-bound benchmark where DVFS has the best chance of being effective at reducing energy consumption. The mcf workload is also part of the SPEC CPU2006 suite (429.mcf) where it has a larger input dataset. We found that this increase in dataset size only served to increase the runtime of the benchmark and not the nature of its workload.

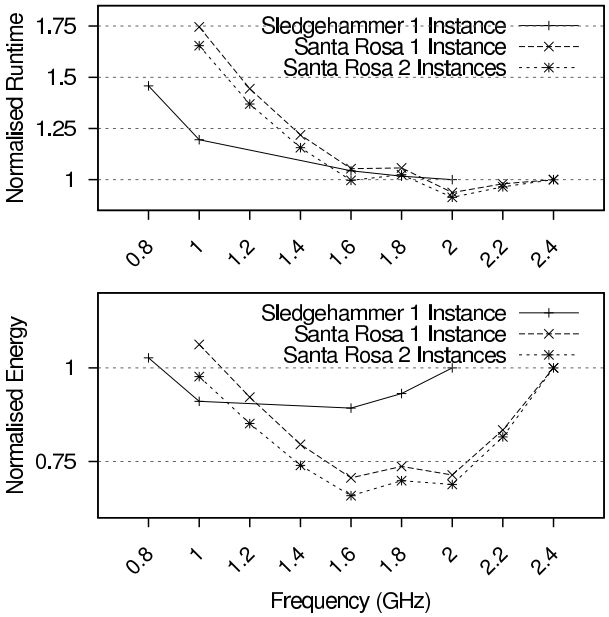

Figure 1: Normalised runtime (top) and energy consumption (bottom) of 181.mcf at different frequencies on the *Sledgehammer* and *Santa Rosa* platforms.

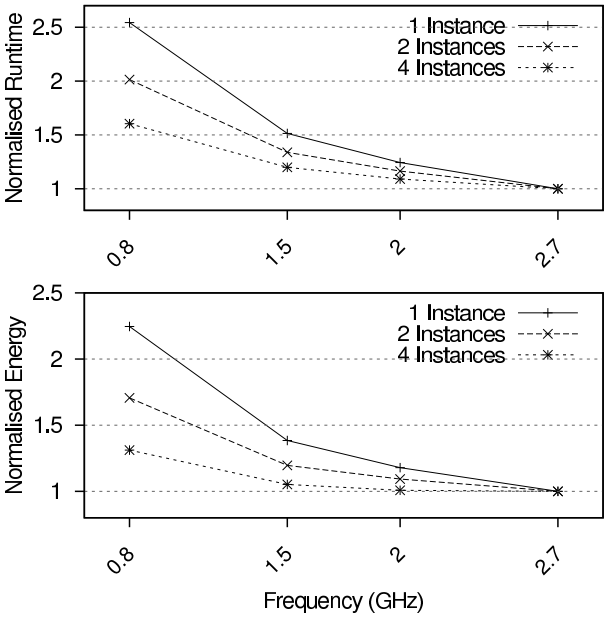

Figure 2: Normalised runtime (top) and energy consumption (bottom) of 181.mcf at different frequencies on the *Shanghai* platform.

4 Experimental Results

Figure 1 shows results obtained from running a single instance of 181.mcf on the *Sledgehammer* based platform and one or two instances (running in parallel on different cores) on the *Santa Rosa* based platform. At the top, runtime is normalised to that at the maximum frequency and a performance hit is taken at the lower frequencies. At the bottom, we show that energy consumption can be reduced by using DVFS. In some cases, a saving of up to a 34 % can be achieved.

In contrast, Figure 2 shows either one, two or four instances of 181.mcf on the more recent *Shanghai* based platform. Again, the top graph shows normalised runtime which increases with a reduction in frequency. However, the bottom figure shows that energy consumption *increases* with the use of DVFS in all three cases.

5 Analysis

5.1 Scaling of Silicon Transistor Technology

As shown in Table 1, the transistor feature sizes of the processors in our test systems have decreased from $130\,nm$ to $45\,nm$ in the 6 years spanning their release. Smaller transistors have a lower threshold voltage and, because sub-threshold leakage grows exponentially, more current is lost into the transistor substrate. Processors with smaller transistors can run at higher frequencies with lower supply voltages. The net effect is a reduction in the dynamic range of power consumption that DVFS can utilise and an increase in static power consumption.

5.2 Improved Memory Performance

In the past, there has been a growing gap between CPU and memory performance. Processor vendors have attempted to mitigate this by including multiple levels of cache in the memory hierarchy and using various other techniques to leverage data locality in workloads. We no longer see significant increases in the clock speeds of CPUs, however due to transistor scaling, we have much larger transistor *budgets*, which are often used for multiple cores and larger caches. However, memory speed is still improving with respect to a single CPU core. This increases the scope to use *prefetching* to further hide memory access latency by reducing the number of cache misses. This allows for better utilisation of the available memory bandwidth for workloads that can only make use of a single core. Newer generations of processors also have a larger prefetch distance, reducing cache miss rates even more.

Figure 3 demonstrates the performance benefit achieved from DRAM prefetching for SPEC CPU2000 workloads on the *Shanghai* platform. Some workloads more than double their execution time when prefetching is disabled. This clearly shows how important prefetching is on newer platforms, where a single core cannot issue memory requests fast enough to saturate the memory bus. 181.mcf is an exception, where the DRAM prefetcher actually reduces performance by about 12 % because it wastes memory bus cycles by pre-fetching useless data.

We also observed that on the older platforms, mem-

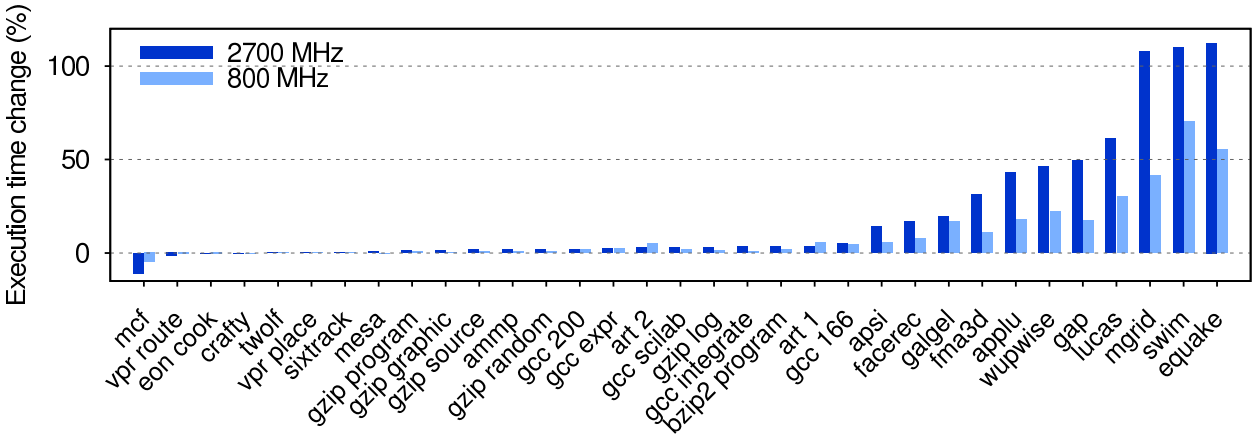

Figure 3: Impact of disabling prefetching on execution time for a single instance on the *Shanghai* platform.

ory frequency is dependent on CPU frequency. On the *Sledgehammer* platform, when the CPU frequency is reduced to 800 MHz, the memory frequency drops from 200 MHz to 160 MHz. On the *Santa Rosa* platform, the memory frequency was observed to vary between 280 MHz and 333 MHz depending on the chosen CPU frequency. Furthermore, at the maximum frequency of 2.4 GHz on *Santa Rosa*, the memory frequency was found to be 300 MHz, lower than the rated speed of 333 MHz. The memory frequency increases to the rated 333 MHz at a CPU frequency of 2.0 GHz which explains why the performance of 181.mcf actually increases with a reduction in CPU frequency in Figure 1. In contrast, the memory frequency on the *Shanghai* platform was consistently measured to be 333 MHz at all CPU frequencies.

Lower memory access latency reduces pipeline stalls which in turn reduces the opportunities to save energy using DVFS.

5.3 Improved Sleep/Idle Modes

The analysis in Section 4 is simplified by assuming that the machine consumes zero power after the benchmark completes. In reality, the system stays powered on but goes into an idle (*sleep*) state. The depth of sleep determines both the power consumption during this period, and the latency to "wake up". In the ACPI-defined C1 sleep state, a processor is halted and its clocks may be gated to reduce power consumption, however, its caches must be kept active to maintain coherence. Table 1 lists the measured idle power for each of our test platforms when the processor is in this state. This value includes power required to keep disks spinning and other devices active. For this reason, a direct comparison of idle power cannot be made between the platforms, as they have different motherboards and hard drives installed. Memory also contributes to idle power because DRAM must be refreshed periodically to retain data. Furthermore, because caches must be kept coherent, processors with larger caches will consume more power in the C1 state. The *Shanghai* processor has more than eight times the cache of the *Sledgehammer*.

The lower the power used in an idle mode, the less benefit there is in using DVFS: overall, less energy is used by running at a high frequency then spending longer idle.

Unfortunately, most systems still consume considerable power when idle, and to give a realistic representation of energy savings, we must "pad" the benchmarks with idle energy to extend the length of the benchmark to that of the lowest frequency.

Figure 4 shows the padded energy consumption of 181.mcf on the *Shanghai* platform. As can be seen, DVFS appears to become much more effective when idle power is factored in. Following this line of reasoning, optimal energy efficiency is achieved by running at the lowest frequency—the opposite of our findings above. This highlights an interesting problem when benchmarking the effects of DVFS—the drop in performance must be accounted for somehow. Often, the objective is not to *minimise* energy use, but to find a good balance of energy savings and performance degradation. In such cases the quantity that should be optimised is the energy-delay product (EDP). Figure 4 shows that on the *Shanghai* platform, EDP is minimised at 0.8 GHz for four instances of 181.mcf, but at the maximum frequency (2.7 GHz) for one or two instances.

5.4 Multi-core Processors

DVFS implementations on multi-core processors are more complex than on single-core processors and are often simplified by limiting the available voltage and/or frequency domains. *Chip-wide* DVFS forces each core on a package to operate at the same frequency and voltage. This further constrains the effectiveness of DVFS

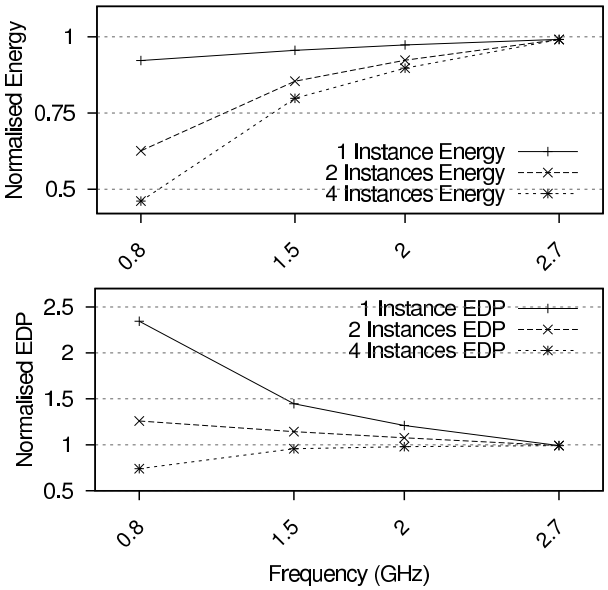

Figure 4: Energy consumption and energy-delay product for 181.mcf on the *Shanghai* platform when **padded** with idle energy.

because workloads running on multiple cores must be analysed as a whole in order to determine whether or not to scale frequency. The most recent AMD Opteron (*Shanghai*) used in our tests provides a single voltage domain, but independent frequency domains. Each core can operate at a different frequency, but the voltage must be no lower than that required by the core operating at the highest frequency.

6 Conclusions

We have analysed the best-case effectiveness of DVFS on three recent generations of AMD Opteron processors, using an extremely memory-bound benchmark. Our results show that on the most recent platform, the effectiveness of DVFS is markedly reduced, and actual savings are only observed when shorter executions (at higher frequencies) are "padded" with the energy consumed when idle (i.e. looking at the energy use for a fixed amount of work). The frequently employed energy-delay product, which balances energy savings against performance losses, is minimised at the highest frequency in all but the most extreme case where highly memory-bound processes are run on *all* cores concurrently.

Our analysis is simplified by only considering a single memory-bound benchmark, however this is the situation where DVFS has the highest chance of being effective. We have also only considered the effectiveness of DVFS on server-class platforms, while DVFS may still be effective on other platform, such as smart-phones [3].

The research presented in this paper also focuses only on processors manufactured by AMD. Intel processors, while experiencing the same trends, can also use DVFS to increase the frequency of a single core when other cores are idle. This so-called "Turbo-Boost" technology can improve performance and energy-efficiency of single-threaded workloads, by opportunistically increasing frequency in order to complete work in a shorter time, and then entering low-power sleep modes.

AMD has already released their next-generation 45nm Opteron processors, codenamed *Istanbul* and *Magny Cours* which utilise faster DDR3 DRAM, have more cores running at lower voltages and have larger DRAM prefetch distances [1]. Intel has moved production into their 32nm fabrication process and AMD will soon follow with their Bulldozer architecture. Furthermore, both AMD and Intel have 22 and 16nm transistor feature sizes in their roadmaps which will further diminish the benefits of DVFS due to rising static power consumption and reduced dynamic power range. Given the shrinking potential for saving energy, it seems only a matter of time until manufacturers abandon DVFS in favour of ultra low-power sleep modes.

Acknowledgements

We would like to thank Godfrey van der Linden and Aaron Carroll for their valuable input during the experimental work conducted for this paper.

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] AMD Opteron processor reference. http://products.amd.com/en-us/OpteronCPUResult.aspx.

[2] SPEC CPU2000 benchmark. http://www.spec.org/cpu2000, Aug 2000.

[3] CARROLL, A., AND HEISER, G. An analysis of power consumption in a smartphone. In *2010 USENIX* (Boston, MA, USA, Jun 2010).

[4] 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.

[5] SNOWDON, D. C. *OS-Level Power Management*. PhD thesis, School Comp. Sci. & Engin., University NSW, Sydney 2052, Australia, Mar 2010. Available from publications page at http://www.ertos.nicta.com.au.au/.

[6] 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).

[7] 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.

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