Technical Report - UPC-DAC-RR-CAP-2010-17 Short-Interval Voltage and Frequency Scaling: Characterization and Optimization Opportunities

Ramon Bertran*, Marc Gonzàlez†, Xavier Martorell*, Nacho Navarro† and Eduard Ayguadé*
*Barcelona Supercomputing Center
Cr. Jordi Girona 29, 08034 Barcelona, Spain
email: {ramon.bertran,xavier.martorell,eduard.ayguade}@bsc.es
†Department of Computer Architecture - Universitat Politècnica de Catalunya
Cr. Jordi Girona 1-3, 08034 Barcelona, Spain
email: {marc,nacho}@ac.upc.edu

***Abstract*—This paper studies the effect of a short-interval DVFS system in a multi-core architecture on a representative workload. We have configured an Intel® Core™ 2 processor to operate using 11 different DVFS P-states that cover a range of frequencies from 2.5GHz to 0.8GHz. We have run the SPECcpu2006 benchmark suite and studied the response of every application in front of smooth changes of frequency. The study characterizes every application in terms of the performance, power and energy levels observed. With the output of this study, we explore the space that a short-interval DVFS system offers to design OS policies. We quantify the power and energy savings that can be obtained assuming certain grade of performance degradation in direct relation to the decrements produced in the operating frequency.**

We have observed that smooth changes in frequency within the lower range of available frequencies have the same effects no matter the characteristics of the target application. On the other side, when the smooth changes occur in the higher levels of the available frequencies, then the memory access ratio exposed by the applications do matter, and the response of the target application is different in terms of performance, power and EDP but equally in terms of energy.

The results show that for the studied architecture, it is possible to define OS power policies that reach power savings of 30% while keeping the performance degradation always under a 17% of slowdown. For energy, it is possible to implement OS energy policies that save a 20% of energy, but with a slowdown about 19%. The basis for this level of optimization is a short-interval DVFS system capable of featuring smooth changes in frequency within a wide range of P-states.

I. Introduction

Energy, power density and power consumption have attracted the interest of researchers since they have become limiting factors in processor designs [1], [2]. Power density reduces the reliability and lifetime of processors [3] and limits the operating frequency [4]. Energy and power consumption are key factors in all market segments. For instance, they are important to extend the battery life of mobile devices or to reduce the need of power supply and energy bills of HPC data centers [5]. As a result, there are several techniques –working at different levels– addressing power consumption issues.

Most of recent processor designs include specific features to control and to adapt the power consumption in order to maximize power efficiency and reliability. For instance, clock gating and clock throttling [6] operate at a very low level and specifically for selected hardware components in the architecture. These mechanisms are automatically triggered by the hardware; the first improves energy efficiency by detecting that a specific component is not used and disabling its clock signal; and the second ensures processor reliability by reducing its activity when a thermal threshold is reached. More flexible hardware mechanisms allow the software to manage the trade-off between power, energy and performance. Dynamic Voltage and Frequency Scaling (DVFS) [7] allows to change the processor voltage and frequency and dynamically adapt the processor performance to power and energy requirements.

Because of obvious energy constraints, DVFS has been largely studied within the context of embedded systems. There are many proposals for OS policies to meet certain restrictions on power and energy consumption levels. For these type of systems, many different configurations (P-states) are available and the architecture is cable to gradually change the operating frequency.

Recently, similar power and energy restrictions have appeared for general purpose multi-core systems. In this other context, DVFS has been also introduced, but has not been so largely studied, specially for what concerns to the number and variety of P-states that should be available to implement power and energy OS policies. At the moment, only few (usually not more than three) and very distant P-states are visible from the OS, although the hardware supports a larger range of configurations not visible to the OS. For instance, our platform firmware offers to the OS three different operating frequencies (2.5GHz, 1.6GHz and 0.8GHz) while its Intel® Core™ 2 Duo processor supports many more.

For multi-core and general purpose systems, DVFS techniques have been only empirically studied with very small ranges of frequencies, and the obtained results have pointed out that the space for designing OS policies is restricted, lead-

ing to implementations that do not expose enough sensibility. It is well known that not all the workloads respond equally in front of changes of the operating frequency. For instance, depending on the memory access ratio, some applications see their energy efficiency greatly improved while others do not. With the current space of P-states, it is not possible to design mechanisms that take advantage of the specific traits of each application or workload.

This paper studies the effect of a short-interval DVFS in a multi-core architecture on a representative workload. We have configured an Intel® Core™ 2 processor to operate using 11 different DVFS P-states that cover a range of frequencies from 2.5GHz to 0.8GHz. We have run the SPECcpu2006 benchmark suite and studied the response of every application in front smooth changes of frequency. The study characterizes every application in terms of the performance, power and energy levels observed. With the output of this study, we explore the space that a short interval DVFS system offers to design OS policies. We quantify the power and energy savings that can be obtained assuming certain grade of performance degradation in direct relation to the increment/decrements produced in the operating frequency. To our best of our knowledge, this is the first paper that studies the effects of a short-interval DVFS mechanism on a multi-core general purpose system, and explores the chances for energy and power optimization in such conditions.

The rest of this paper is organized as follows: Section II describes some background information and related work. Section III describes the experimental framework that has been used to perform the study in this paper. Section IV includes the performance, power and energy characterization of the SPEC 2006 suite. Section V describes the opportunities that a short-interval DVFS system offers to implement power and energy aware policies. Finally, Section VI concludes the paper with its main conclusions.

II. Background and Related Work

The Advance Configuration and Power Interface (ACPI), introduced in 1996, defines a platform-independent interface that allows to detect, configure, manage and monitor platform devices. Although there are other aspects to the specification, ACPI is just an interface that allows to perform power management functions, avoiding the need to rely on firmware code (BIOS) to do them. Hence, it gives the OS (Operating System) all the responsibility to implement Operating System-directed Power Management (OSPM) policies. This turns out to be quite effective, because the OS normally is in a position to know the status of the overall system.

We can distinguish two type of power management regarding to the CPU: the management when the CPU becomes idle or when it has some work to do. For the first type, ACPI defines the C-states. These states rely on power down progressively CPU components, entering in more 'deeper' states. Thus, saving more power at the expense of higher latency for 'wake-up' the CPU. And for the second one, ACPI defines the P-states, called also processor performance states. A common technique implemented by CPU vendors for adjusting performance is to change the frequency and adjust the voltage. These mechanism, called DVFS (Dynamic Voltage and Frequency Scaling)[1], results in high power savings.

The flexibility that is offered by the ACPI services has caused the industry to adopt it as one standard to implement OSPM policies. These address the goal of achieving satisfactory levels of energy efficiency while fulfilling certain problem-specific constraints. Mainly these constraints are performance and power. The first means to do not let the performance decrease more than a given threshold. Thus, maintaining a certain system responsiveness, task throughput or fulfilling the application deadlines. The second means to not surpass a certain level of power consumption in order to reduce temperature, avoid thermal issues and cooling costs. At the end, there have been implemented many solutions optimizing all such aspects.

For instance, C.Hsu et al. [8] and R. Ge et al. [9] present on-line systems to derive the ideal frequency (P-state) given a maximum allowed slowdown. Therefore, they achieve the optimal energy and power savings for the given slowdown. Other approaches try also to save energy while minimizing the impact on performance. B.Rountree et al. [10] propose a technique to detect the critical path of parallel applications and then reduce the unbalance between threads by slowing down the fastest ones. Q.Wu et al. [11] proposes a dynamic compilation framework that manages DVFS with the aim of saving energy without reducing the performance.

Other solutions, study how to maximize performance given a maximum power budget. C.Isci et al. [12] evaluate several different policies for global multi-core power management. R.Teodorescu et al. [13] propose variation-aware algorithms for application scheduling on CMPs for saving power and maximizing throughput. The problem of meeting a global power-budged is also targeted in the work of K.Meng et al. [14]. R.Kotla et al. [15] propose a scheduling mechanism constrained by both metrics, the maximum power budget and the maximum performance lost. Other works, target the optimization of direct related metrics. For instance, Y.Wang et al. [16] uses power management mechanism to control and constrain the temperature and A.Merkel et al.[17] introduces CMP scheduling for maximizing the energy delay product.

There are many power management proposals but few of them evaluate the room they have for optimization, given a certain constraints. As a result, they do no know how far from or close to the optimal solution they are. There exist some works which characterize the DVFS mechanisms but targeting other purposes than the ones of this paper. For instance, D.Snowdown et al.[18], [19] proposes performance and power models under DVFS but for embedded platforms. W.Bircher et al. [20] evaluates power management mechanisms centering the discussion on the effects of different P-state/C-state combination within a single chip.

[1]This technique is known as SpeedStep in Intel processors, as PowerNow! or Cool'n'Quiet in AMD processors, and as PowerSaver in VIA processors.

As a result, there is some well-known assumptions about the effects of DVFS (P-states) on power, energy and performance. But real quantification (not simulated) of them on multi-core current super-scalar architectures has not been perform at great depth so far in the literature. As far a we know, we are the first who evaluate empirically up to eleven different P-states, extending the 3 provided by the firmware (BIOS), on an Intel® Core™ 2 Duo platform. The short-interval P-state definition allows a fine-grained characterization of DVFS effects on power, energy and performance. Moreover, our evaluation unveils and quantifies the optimization space lost when the P-state provided are a subset of the one supported by the processor.

III. Experimental System Setup

We have performed a characterization study on an Intel® Core™ 2 Duo T9400 microprocessor [21] featuring 2x2GB DDR1066 memory chips. The Intel® Core™ 2 microprocessor integrates two cores on a single die.

This architecture has been designed with energy efficiency as a main design factor. It is conceived to optimize performance across a range of market segments and power envelopes. Hence, it includes several power related mechanisms. For instance, clock gating is extensively used to switch down the clock signal on several of the microarchitecture components. Moreover, the processor implements DVFS[1], allowing to modify dynamically the operational frequency and voltage. The frequency and voltage ranges are [0.800,2.533] GHz and [0.75,1.25][2] volts respectively. The core frequency is a multiple of the external (bus) clock speed, which in our system is 266 Mhz. Frequencies below 1.600 GHz are achieved by halving the external (bus) clock speed to 133 MHz. As a result, the access to main memory is affected when frequencies are below 1.600 GHz.

The operating system used is Ubuntu Linux 8.10 i686 version [23]. The system is booted in level 1 mode[3] while all the power and performance data has been gathered. This nullifies the overhead and possible interferences that other processes might introduce. Only one of the both cores was enabled during the experiments in order to evaluate and characterize the DVFS mechanism without the indeterminism introduced when multiple cores are enabled, such as the different shared resource contention produced by process scheduling.

The embedded controller firmware of the platform –the BIOS– provides information about the power source, which fulfills the SMAPI specification [24]. This specification defines an interface to obtain power consumption measurements with a guaranteed granularity and accuracy. As a result, the platform provides power measurements in a granularity of milliwatts with a maximum error of 2%.

[1]Also known as Enhanced Intel® SpeedStep Technology.

[2]Each processor is programmed with a maximum valid voltage identification value (VID), which is set at manufacturing and cannot be altered. Individual maximum VID values are calibrated during manufacturing such that two processors at the same frequency may have different settings within the VID range. [22]

[3]Also know as standalone mode.

Table I: Defined ACPI *P-States* on our experimental platform

Freq (GHz)	Mode	Bus Freq (Mhz)	Multi--plier	% reduc--tion	slow--down
2.533	*HFM*	266	9.5x	0.00 %	1.00x
2.400		266	9.0x	5.25 %	1.05x
2.266		266	8.5x	10.5 %	1.12x
2.133		266	8.0x	15.8 %	1.19x
2.000		266	7.5x	21.0 %	1.27x
1.866		266	7.0x	26.3 %	1.35x
1.600	*LFM*	266	6.0x	36.8 %	1.58x
1.200		133	9.0x	52.6 %	2.11x
1.066		133	8.0x	57.9 %	2.37x
0.933		133	7.0x	63.2 %	2.71x
0.800	*SuperLFM*	133	6.0x	68.4 %	3.16x

We installed a Linux® kernel 2.6.28 with the required patches to allow access to the performance monitoring counters (PMCs). The `tp_smapi` [25] module is loaded in order to be able to gather the power information. This module creates some entries in the `/sys` file-system that provide information such as the current power consumption. We use a modified version of `pfmon` [26] to access to the PMCs and power consumption simultaneously. We have switched off all sources of power consumption –i.e. the display– that we do not want to interfere. For platform components where it is not possible to switch off, we configured them at constant operation mode. For example, we found that the fan of the processor introduces up to 0.5W variations on power consumption. Hence, we configured it to always operate at full-speed. Under this conditions, we tracked the power consumption during five minutes before each experiment and we found the baseline power consumption for the idle system to be 9.8W. The fact that for all the experiments carried out we found a fairly constant idle power consumption demonstrated that we nullified possible interferences. For the purpose of power estimation only, we assumed that baseline, 9.8W, to be the power consumption of the platform except the processor and the memory. This assumption and the accuracy ensured by the SMART specification avoids the need of more complicated measuring techniques [27].

We also used the mechanism provided by the Linux® kernel to override the Differentiated System Description Table (DSDT). The DSDT table is part of the ACPI specification and it supplies configuration information about a base system such as the P-state descriptions. Originally, the P-state table of our platform only provided three P-states. They are the *HFM* (High Frequency Mode), the *LFM* (Low Frequency Mode) and the *SuperLFM* (Super Low Frequency Mode) listed in Table I. Our modified DSDT enabled several P-states between the predefined ones, allowing to explore in more detail the effects of the DVFS mechanism. Table I also shows the percentage of external bus frequency, the frequency multiplier, frequency reduction and slowdown of each P-state. From now on, we will refer to each P-state using their frequency.

We have collected data for 26 applications of the SPEC-cpu2006 [28] benchmark suite using `pfmon` using each available frequency. We programmed `pfmon` to run for 30 minutes or less if the benchmark ends earlier and print the PMCs

values and power measurements three times every second. The PMCs reported values are the number of cycles, the number of instructions retired and the number of memory bus requests. In every case, only for the first 400.000 million instructions are processed and for each benchmark and frequency pair, we compute the average power consumption, IPC and memory access ratios plus other derived metrics such as energy or EDP.

IV. Short-Interval DVFS Characterization

In this section we characterize the set of metrics that any OSPM policy must consider. We analyse the performance, the power consumption, the energy and the energy delay product (EDP) of the entire SPECcpu2006 benchmarks when running on our experimental platform for all the available P-states. Our goal is to obtain a general understanding of the behavior of such metrics for the different P-states. This allows to know which are the most suitable P-states when implementing OSPM policies. The following subsections discuss the general trends of each metrics studied. First we start analysing the performance behavior and we perform a benchmark classification that allows to analyse and understand the effects of DVFS on rest of metrics.

A. Performance

Figure 1 summarizes four the metrics analysed for each benchmark and P-state evaluated. Concretely, Figure 1a shows the time required to execute the first 400.000 million instructions of each benchmark on each P-state.

From the benchmarks point of view, we can see that the 26 benchmarks show significant differences in execution time. For instance, for the slowest frequency, the fastest benchmark –the `454.calculix`– last about 250 seconds and the slowest one –the `429.mcf`– last more than 800 seconds. This means that the instructions per cycle (IPC) ratio that the benchmarks exhibit are distributed among a wide range, allowing us to perform a characterization of a wide range of performance situations.

Analysing the behavior by P-state, we observe that the effect on the execution time looks similar and quite homogeneous on all the benchmarks when frequency is changed. However, there are cases where the same reduction on frequency does not affect equally the execution time. For example, the `462.libquantum` benchmark doubles the execution time when the frequency is decreased from the maximum one to the minimum one. But, the `454.calculix` slowdown for the same frequency change is about 2.5x. This differences suggest that the IPC of the benchmarks does not vary uniformly. Figure 1b shows the IPC of each benchmark normalized to the maximum frequency. This figure presents clearly the different impact on performance due to frequency scaling depending on each benchmark. For example, the IPC improvement can be up to about 45% for `462.libquantum`, whereas `454.calculix` does not exhibit any improvement at all. The reason of such improvements is the decoupling of the operational frequencies between the processor and the main memory. Therefore, the memory bound programs take advantage of decreasing the frequency since they have to wait less cycles for data to be brought from main memory. Figure 1c shows the average memory access ratio (memory bus transactions per cycle) for each benchmarks. We see that the benchmarks with more memory access ratios are the ones which exhibited more IPC improvement in Figure 1b.

Table II: Benchmark classification

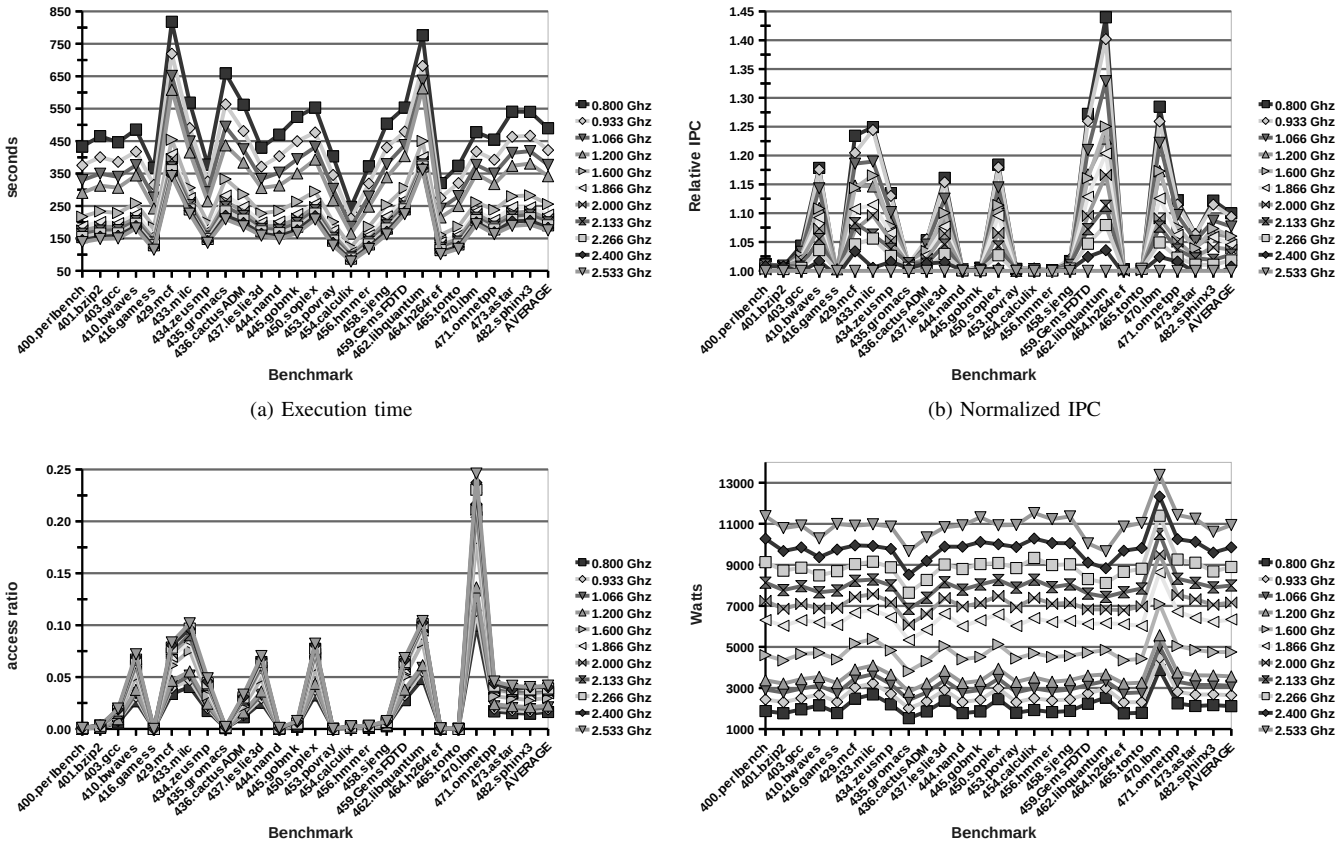
Type	Benchmarks
Memory bound	410.bwaves, 429.mcf, 433.milc, 437.leslie3d, 450.soplex, 459.GemsFDTD, 462.libquantum, 470.lbm
Mix	403.gcc, 434.zeusmp, 436.cactusADM, 471.omnetpp, 473.astar, 482.sphinx3
CPU bound	400.perlbench,401.bzip2, 416.gamess, 435.gromacs, 444.namd, 445.gobmk, 453.povray, 454.calculix, 456.hmmer, 458.sjeng, 464.h264ref, 465.tonto

Since the performance behavior is a key factor to take into account for designing any OSPM policy and because performance affects the power, the energy and the EDP metrics, we classify the benchmarks based on their performance behavior in front of frequency variations. We have shown that this is directly related to the memory behavior of the applications, therefore we classify in Table II the benchmarks in three types: memory bound, mix and CPU bound. This classification will be used during the rest of the paper to facilitate the discussion.

Figure 2 summarizes the behavior of the four main metrics analysed for each type of benchmark and frequency. The result shown are normalized to maximum frequency and the frequency range is split in three ranges with different Y-axis scale to improve readability. Moreover, in order to analyse the impact of each frequency change on such metrics, we plotted the derivative in Figure 3 which is the inclination of the curve between each pair of adjacent frequencies.

Figure 2a summarizes the average effect on performance. In overall, the CPU bound benchmarks present higher slowdown than the Mix benchmarks. And these last ones also always exhibit higher slowdowns than the memory bound benchmarks, corroborating the well-known effect that the impact of frequency reduction is less noticeable on benchmarks with memory activity. Another important point to remark is that the effect on performance of frequency scaling is much higher on lowest rage of frequencies, as shown in Figure 3a. Finally, both figures show that for the entire frequency range the impact on performance differ depending on the application type.

To finish the performance discussion, we overview the opportunities for power aware policies from the performance point of view. The frequencies below the 1.600 GHz, which present a high performance degradation, are not suitable for any power aware policies with the aim of minimizing the impact on performance. As a result, the suitable frequencies are the ones in which the response of every application is in accordance to its type or category. Therefore, OSPM policies should be aware of the type of benchmark being executed since the impact on performance can differ vastly (doubles) depending on the benchmark type. For instance, for the 2.266 GHz P-state, the CPU bound benchmarks exhibit



(a) Execution time

(b) Normalized IPC

(c) Memory access ratio

(d) Average Power

Figure 1: Metrics summary of the SPECcpu2006 benchmarks for all available frequencies

about 12% slowdown whereas the Memory bound ones present about 6%.

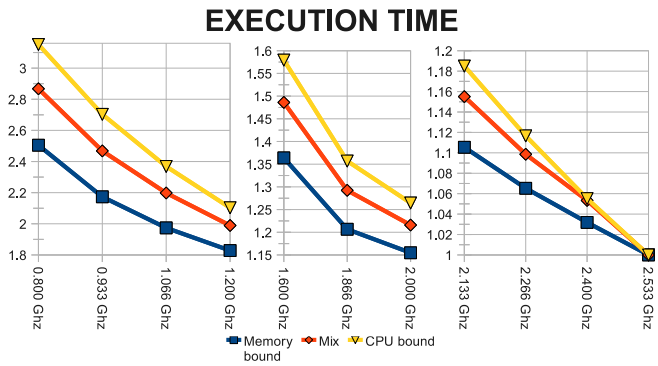
B. Power

Figure 1d shows the average power consumption for each benchmark. In overall, every benchmark behaves similarly, exposing the same power level in each frequency (within a ± 1 watt range). The exception, is `470.lbm` which consumes about 2 watt more than the rest in all frequencies. This is because this benchmark has a high memory activity in conjunction with a reasonably high IPC. In overall, we observe up to 10 watts of difference between the highest and the lowest frequencies.

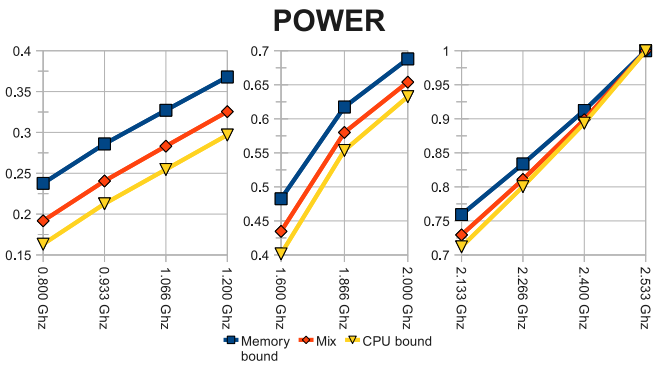
In Figure 2b, we show that the power savings of the CPU bound applications are always higher than the Mix benchmarks, and that the Memory bound benchmarks are the ones which show less power savings. The reason is twofold. First, the higher the memory access ratios, the higher the power consumption of main memory. Therefore, lowering the CPU frequency has less impact on the global power consumption, as it just affects the power consumption within the processor chip. Second, the IPC increment –shown during the performance discussion– means more activity on the CPU. Since the CPU power consumption is direct related to the CPU activity, some of the power reduction due to DVFS is nullified by the experimented IPC increment. This is case of the benchmarks with a high memory activity.

Figure 3b shows how the frequency impact changes on power depending on the frequency range. We detect two different trends. For frequencies below 1.60 GHz, the impact on power is similar across all the benchmark types. For frequencies higher or equal to 1.60 GHz, the impact on power is different depending on the benchmark type. In general, we observe that that memory bounded programs are less impacted by DVFS than CPU bounded ones, corroborating the behavior shown in Figure 2b. As initial conclusion, if OSPM policies address power saving objectives, depending on the range of frequencies to explore, the benchmark type might or might not be an issue. For a frequency change in the range of the lower frequencies, the benchmark type distinction is useless. For the rest of frequency ranges, the knowledge about the memory access ratios might lead to a better selection of the application to be affected by a frequency change.

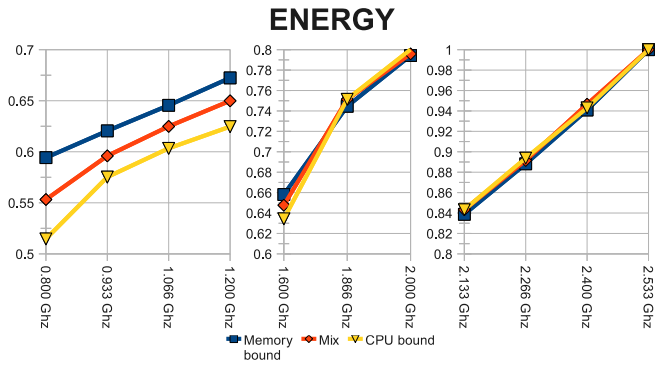
However, when both power and performance are considered, different conclusion are obtained. Figure 4 relates some of the analysed metrics to performance. Concretely, Figure 4a shows the power savings/performance degradation ratio. A ratio above 1 means that more percentage of power is saved



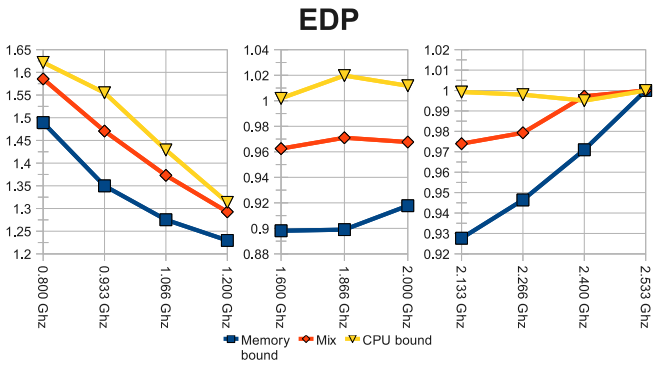
(a) Execution time



(b) Power

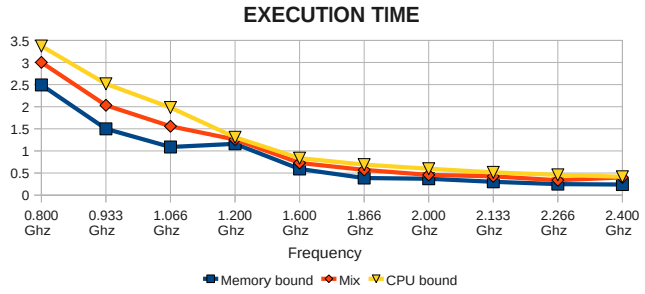


(c) Energy

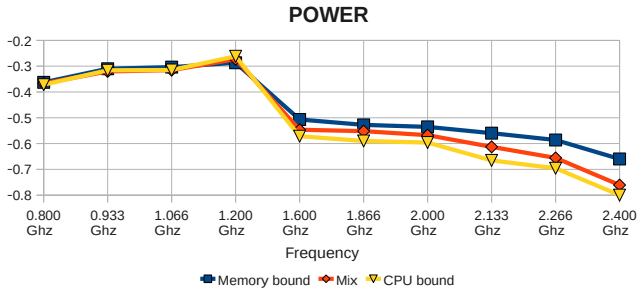


(d) EDP

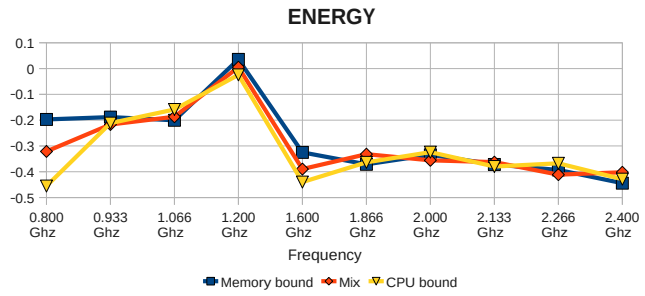
Figure 2: Metrics summary of the SPECcpu2006 benchmarks for all available frequencies, normalized to maximum frequency



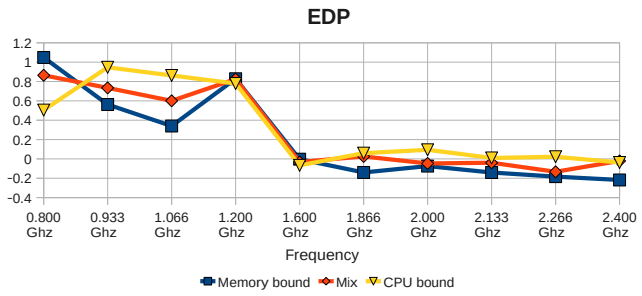
(a) Execution time



(b) Power



(c) Energy



(d) EDP

Figure 3: Derivative for each adjacent frequency pair studied of the SPECcpu2006 benchmarks of all metrics studied

than percentage of execution time is increased. Now, we see that memory bound benchmarks obtain higher ratios than the other benchmarks. Actually, for frequencies above 1,866 GHz the memory bound benchmarks reduce the double of power than performance is lost. The rest of benchmarks exhibit less remarkable ratios, but they are higher than one. Therefore, depending on the metric to optimize, OSPM policies working on this higher frequency range should take into account the type of benchmark because the optimal option would differ. For example, if the main constrain is to complete the workload on time, the OSPM policy would choose a memory bound benchmark in order to minimize the performance impact and fulfill the deadline. However, if the constrain is that instantaneous power must never exceed a given power budget, the OSPM policy would choose a CPU bound program in order to maximize the power reduction while affecting only one benchmark of the workload.

Finally, for the range of frequencies below the 1.600 GHz is worth to explain two trends. First, the differences in behavior across the benchmark types are unnoticeable. Therefore, the choice of the optimal solution for an OSPM policy should be based on other factors. And second, in contrast to the higher frequency ranges where ratios are always above 1, in this case, all ratios are below one. Therefore, on this range, the performance degradations are always higher than the power savings. Again, this means that OSPM policies only should use these lower P-states on situations when power must be reduced at any cost.

C. Energy and EDP

Energy efficiency is also an important factor for any OSPM policy. This factor is usually left behind the power and performance requirements. However, it should be maximized when the other are met or do not exist. We also analyze the energy delay product (EDP), which is a common metric used to evaluate energy efficient solutions while considering also the performance degradation.

Energy consumption levels are exposed in Figure 2c and Figure 3c. First thing to notice is that the different behavior in performance and power for higher frequencies no longer exists. Therefore, from the energy perspective it does not make a difference the benchmark type as the effects in performance and power are canceled when energy is calculated. This means that for OSPM policies applied when once performance and power constrains are met, there is not difference on energy reduction between choosing a CPU bound benchmark of a memory bound one. Second, for the lower frequencies the CPU bound benchmarks tend to reduce more the energy because we have seen that they consume less power and even they exhibit greater performance slowdowns than the memory bound benchmark do. Again, the rationale of this behavior is the difference weight of memory power consumption depending on the benchmark type. And third, there is not any race-to-idle or race-to-sleep [29] when taking into account only CPU and memory energy consumption. Therefore, reducing the frequency improves the energy efficiency in all the cases. There is only one step, when entering to *SuperLFM* –between 1.600 GHz and 1.200 GHz frequencies– that energy is not reduced. This is shown in Figure 3c, where the 1.200 GHz show slopes close to zero.

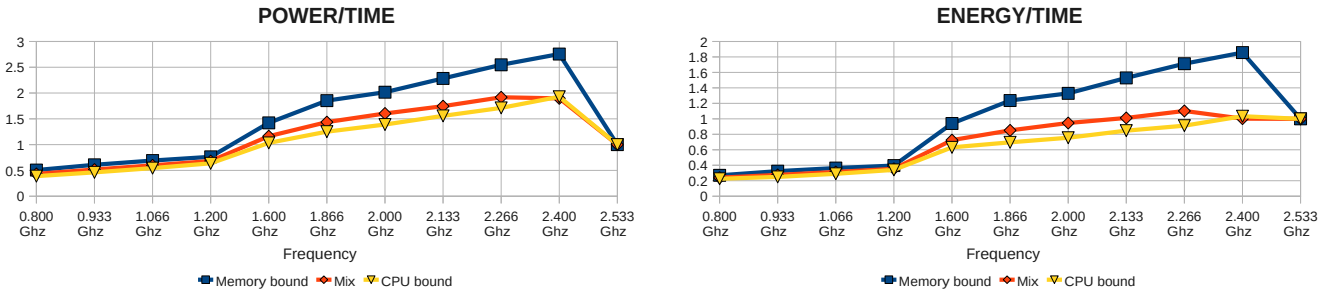
Figure 4b relates the energy gains with the performance degradation. We can see very similar trends for both energy and power (Figure 4a). However, all the curves are displaced down resulting in that only the memory bound benchmarks consistently save more energy than loose performance. The CPU bound benchmarks always loose more performance than save energy, except on the 2.400 GHz P-state, in which they have a one to one ratio.

This relation can be seen also as the EDP. Figure 2d and Figure 3d show the normalized EDP and their derivative respectively. In the first one, in overall the EDP is close to one, except when decreasing the frequency below the 1.600 GHz point, when it raises quickly. In addition, we see that the EDP of the memory bound benchmarks always is between than the one at maximum frequency. Concretely, memory bound benchmarks obtain the optimal EDP at 1.600 GHz point, when it is reduced by about 10%. These trends are also shown in Figure 3d where all the benchmark types exhibit low impacts on EDP –near to zero– but when frequency is below the 1.600 GHz ratio the impact grows quickly.

V. Optimization space for power/energy aware policies

In this section, we implement an off-line solution to solve the common problem where a set of benchmarks (a workload) has to be executed without surpassing the power or energy budged available, but at the same time the performance of each benchmark should no be degraded more than a given factor. This is the usual situation where there is a need to find a trade-off between energy or power savings and performance degradation. For instance, system administrators need to reduce overall energy and power costs, while keeping each user satisfied. Concretely, we want to know how much performance degradation should the applications accept to achieve a given power or energy reduction, while defining a feasible optimization space by excluding unreachable. For instance, the extreme case where we want to reduce 50% the overall power consumption but allowing only 5% degradation of performance on each benchmark.

We have designed the following algorithm to assign frequencies to applications and fulfill both the given the target power/energy reductions and the maximum slowdown applicable to every application. First, we compute the minimum frequency of each benchmark where the obtained slowdown is above the maximum specified. At that point, we have the maximum overall energy or power savings. If they are below the target reduction no solution exists. Otherwise, we have the situation where we obtain more power or energy savings than the target. Therefore, we have room to reduce the performance degradation by incrementing the frequency of some benchmarks in such a way that the power or energy requirements still are met. Exploring all possible combinations



(a) Execution time

(b) Power

Figure 4: Power and Energy gains relative to Performance looses

Table III: % Percentage of energy savings, percentage of slowdown and percentage of affected benchmarks for different targets of energy reductions and maximum allowable per benchmark slowdown

Target Workload %Energy Reduction	Maximum %Performance Lost Allowable per Benchmark								
	10%	15%	20%	25%	30%	35%	40%	45%	50%
5%	5.1/3.7/62%	5.1/4.0/38%	5.2/4.9/31%	5.3/5.2/27%	5.1/5.1/27%	5.4/5.7/27%	5.4/6.5/31%	5.2/7.2/27%	5.2/7.2/27%
10%	-	10.1/8.4/77%	10.1/9.9/58%	10.0/10.1/54%	10.0/10.7/46%	10.2/11.5/42%	10.2/11.7/42%	10.0/11.8/38%	10.0/11.8/38%
15%	-	-	15.0/13.5/77%	15.3/14.5/77%	15.2/16.6/73%	15.0/17.4/65%	15.0/18.1/62%	15.1/18.8/58%	15.1/18.8/58%
20%	-	-	-	20.2/19.1/96%	20.0/20.7/88%	20.3/22.1/77%	20.4/24.0/81%	20.1/24.8/77%	20.2/25.2/77%
25%	-	-	-	-	-	25.2/27.5/100%	25.2/28.6/92%	25.0/29.4/88%	25.1/29.9/88%
30%	-	-	-	-	-	-	-	-	30.0/35.9/100%

Table IV: % Percentage of power reduction, percentage of slowdown and percentage of affected benchmarks for different targets of power reductions and maximum allowable per benchmark slowdown

Target Workload %Power Reduction	Maximum %Performance Lost Allowable per Benchmark								
	10%	15%	20%	25%	30%	35%	40%	45%	50%
5	5.2/2.5/35%	5.2/3.2/19%	5.1/3.7/19%	5.1/3.2/15%	5.2/3.2/15%	5.0/3.2/15%	5.0/3.2/15%	5.0/3.2/15%	5.0/3.2/15%
10	10.1/4.8/73%	10.3/5.7/46%	10.0/6.3/35%	10.0/6.4/31%	10.1/6.6/31%	10.1/6.7/27%	10.1/6.5/27%	10.2/7.1/27%	10.2/7.1/27%
15	-	15.3/8.3/69%	15.3/9.5/54%	15.1/9.8/46%	15.1/10.2/42%	15.2/11.0/38%	15.0/11.1/35%	15.3/10.6/38%	15.3/10.6/38%
20	-	20.2/10.3/92%	20.2/12.1/69%	20.4/12.8/65%	20.0/13.4/58%	20.3/14.7/54%	20.3/15.3/50%	20.0/15.7/46%	20.0/15.7/46%
25	-	-	25.3/14.6/85%	25.3/16.2/77%	25.3/16.0/69%	25.0/17.7/65%	25.4/18.7/62%	25.3/19.4/58%	25.2/19.4/58%
30	-	-	30.2/17.2/100%	30.3/18.5/92%	30.3/19.8/81%	30.1/20.5/77%	30.4/21.1/73%	30.1/22.8/69%	30.3/23.3/69%
35	-	-	-	-	35.1/22.3/92%	35.3/24.1/88%	35.0/24.6/81%	35.2/26.2/77%	35.2/25.9/77%
40	-	-	-	-	-	40.2/27.5/100%	40.1/28.4/92%	40.1/29.4/88%	40.3/29.9/88%
45	-	-	-	-	-	-	-	45.3/33.2/100%	45.2/33.3/96%

falls to an exponential problem, being an NP-problem [30]. Therefore, our algorithm avoids that complexity by using an greedy heuristic for selecting the benchmark to which apply a frequency increment. Given the current configuration, we increase the frequency of the benchmark that will result in the minimal impact on the target energy or power reduction. Then, we repeat this process using the new configuration until the energy or power reduction is no longer met, keeping the last valid configuration. We will see that even not all solutions are explored, the algorithm always obtain a configuration satisfying the target energy or power reductions.

Table III and IV summarizes the results obtained when targeting a specific energy and power reduction respectively. All the results analysed in Section IV are used as the input workload to optimize. In each cell three percentages are shown for the target reduction and the allowed maximum performance lost per benchmark. The first is the obtained reduction for the entire workload, which always is equal or greater than the target one. The second is the percentage of slowdown for the entire workload. And the third, is the percentage of benchmarks which their frequency has been reduced. If the cell is empty, no configuration fulfils the requirements. For instance, the top left cell in Table III shows the results to achieve a 5% energy reduction and allowing a maximum benchmark slowdown of 10%. In this case, our algorithm obtains 5.1% energy reduction with an overall workload slowdown of 3.7% affecting 62% of the workload benchmarks.

If we analyze the trends of the results presented in Table III, we see that for all the cases the achieved energy reduction is fairly close to the target. Another point to remark is that the overall slowdown for achieving a certain percentage of energy reduction, is always below the percentage of energy reduction. However, in all cases some of the benchmarks exhibit slowdowns higher than the energy reduction. Notice, that there is not a valid configuration where the energy reduction target is higher or equal to maximum percentage of slowdown. Another

point to remark is that when allowing low slowdowns per application (below 30%), the energy savings are always higher than the performance looses. This trend changes when the allowed performance lost is higher than 30%. Then, the energy savings are below the overall workload slowdown. Finally, if we analyse the trends of the percentage of benchmarks affected by a reduction of frequency, it decreases when the maximum allowable slowdown per benchmark increases. This suggests that our heuristic tends to concentrate the frequency decrements on particular applications when the allowed slowdown is high. But when the slowdown has to be in the range of 10%, then the energy or power savings are achieved by distributing the performance degradation among all benchmarks. Besides, the lesser the concentration of the frequency changes, the lesser the workload slowdown. This is a general trend, but can be clearly observed in the sequence of values for the case of an energy reduction of 5%. When a 10% of slowdown is accepted per application, the workload slowdown corresponds to a 3.7%, while when the accepted slowdown is 50%, then the workload slowdown corresponds to 7.2%.

We see the same trends when the reduction is applied to power instead to energy. Table IV summarizes the results. However, there exist some differences. The overall power reduction is always higher than the overall performance lost, but their difference decreases when higher slowdowns per benchmark are allowed. Table V summarizes the frequency assigned to each benchmark for a given power/performance threshold configuration. On the left, we see the frequencies used when all of them are available. We see the importance of extensive short-interval DVFS configurations because they allow to perform more fine-grained trade-off between power/energy and performance.

Finally, we have repeated the experiments but only allowing the usage of the three frequencies available in our platform: 2.54GHz, 1.60GHz and 0.8GHz. Table VI summarizes the results obtained. The exact frequency assigned to each benchmark of some of the possible configuration is also detailed on the right part of Table V. Clearly, the range of applicable slowdown is now restricted and starts at a 35% of slowdown. This means that no possible energy or power savings are possible unless the applications admit a performance degradation of 35%. In conclusion, this shows the necessity of a short-interval DVFS system to implement flexible power/energy OS policies, taking advantage of the different traits a workload might expose.

Table VI: Percentage of reduction, slowdown and affected benchmarks for different targets of energy and power reductions

Target Workload %Energy Reduction	Maximum %Performance Lost Allowable per Benchmark			
	35%	40%	45%	50%
5	6.8/5.9/12%	7.5/7.5/15%	5.2/5.7/12%	5.1/5.6/12%
10	-	10.2/9.4/19%	10.1/11.1/23%	11.1/12.5/27%
15	-	-	-	15.3/15.9/35%

Target Workload %Power Reduction	Maximum %Performance Lost Allowable per Benchmark			
	35%	40%	45%	50%
5	5.9/5.9/12%	5.5/5.5/12%	5.5/5.4/12%	5.5/5.4/12%
10	-	-	11.5/11.2/23%	11.6/11.0/23%
15	-	-	15.8/14.6/31%	15.7/14.4/31%
20	-	-	-	20.2/17.6/38%

VI. Conclusions

This paper has studied the effect on a representative workload (the SPEC 2006 benchmark suite) of a short-interval DVFS system. The outcome of this study is that of specifying for what frequency ranges the distinction of Memory-bound and CPU-bound applies to implement power/energy aware policies. We have observed that smooth changes in frequency within the lower range of available frequencies have the same effects no matter the characteristics of the target application. On the other side, when the smooth changes occur in the higher levels of the available frequencies, then the memory access ratio exposed by the applications do matter, and the response of the target application is different in terms of performance, power and EDP, but equally in terms of energy.

Finally, the results show that for the studied architecture, it is possible to define OS policies that reach power savings of 30% while keeping the performance degradation always under a 17% of slowdown. For energy, it is possible to implement OS policies that save a 20% of energy, but with an slowdown about 19%. The basis for this level of optimization is a short-interval DVFS system capable of featuring smooth changes in frequency within a wide range of P-states.

References

[1] T. Mudge, "Power: A first-class architectural design constraint," *Computer*, vol. 34, no. 4, pp. 52–58, 2001. 1

[2] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," in *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 66–77. 1

[3] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture*. New York, NY, USA: ACM, 2003, pp. 2–13. 1

[4] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *DAC '03: Proceedings of the 40th annual Design Automation Conference*. New York, NY, USA: ACM, 2003, pp. 338–342. 1

[5] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM, 2001, pp. 103–116. 1

[6] Y.-M. Kuo, S.-H. Weng, and S.-C. Chang, "A novel sequential circuit optimization with clock gating logic," in *ICCAD '08: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*. Piscataway, NJ, USA: IEEE Press, 2008, pp. 230–233. 1

[7] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 78–88. 1

[8] C.-h. Hsu and W.-c. Feng, "A power-aware run-time system for high-performance computing," in *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. Washington, DC, USA: IEEE Computer Society, 2005, p. 1. 2

[9] R. Ge, X. Feng, W.-c. Feng, and K. W. Cameron, "Cpu miser: A performance-directed, run-time system for power-aware clusters," in *ICPP '07: Proceedings of the 2007 International Conference on Parallel Processing*. Washington, DC, USA: IEEE Computer Society, 2007, p. 18. 2

Table V: Frequencies assigned to each benchmark for different power/performance trade-off and available frequencies.

Benchmark Benchmark	Power/Performance threshold												
	All frequencies									3 frequencies			
	5/10	10/10	15/15	20/15	25/20	30/20	35/30	40/35	45/45	5/35	10/45	15/45	20/50
400.perlbench	2.400	2.400	2.270	2.270	2.130	2.130	2.000	2.000	1.870	2.530	2.530	2.530	2.530
401.bzip2	2.400	2.400	2.270	2.270	2.130	2.130	2.000	1.870	1.870	2.530	2.530	2.530	2.530
403.gcc	2.400	2.400	2.270	2.270	2.130	2.130	2.000	1.870	1.870	2.530	2.530	2.530	2.530
410.bwaves	2.270	2.270	2.130	2.130	2.000	2.000	1.870	1.870	1.600	2.530	1.600	1.600	1.600
416.gamess	2.400	2.400	2.270	2.270	2.130	2.130	2.000	2.000	1.870	2.530	2.530	2.530	2.530
429.mcf	2.130	2.130	2.130	2.130	1.870	1.870	1.870	1.600	1.600	1.600	1.600	1.600	1.600
433.milc	2.270	2.270	2.000	2.000	2.000	2.000	1.870	1.870	1.600	2.530	1.600	1.600	1.600
434.zeusmp	2.400	2.270	2.130	2.130	2.000	2.000	1.870	1.870	1.870	2.530	2.530	2.530	1.600
435.gromacs	2.530	2.400	2.270	2.270	2.130	2.130	2.000	1.870	1.870	2.530	2.530	2.530	2.530
436.cactusADM	2.530	2.400	2.270	2.270	2.130	2.130	2.000	1.870	1.870	2.530	2.530	2.530	2.530
437.leslie3d	2.530	2.270	2.130	2.130	2.000	2.000	1.870	1.870	1.600	2.530	1.600	1.600	1.600
444.namd	2.530	2.400	2.270	2.270	2.130	2.130	2.000	2.000	1.870	2.530	2.530	2.530	2.530
445.gobmk	2.530	2.400	2.270	2.270	2.130	2.130	2.000	1.870	1.870	2.530	2.530	2.530	2.530
450.soplex	2.530	2.270	2.130	2.130	2.000	2.000	1.870	1.870	1.600	2.530	2.530	1.600	1.600
453.povray	2.530	2.400	2.270	2.270	2.130	2.130	2.000	2.000	1.870	2.530	2.530	2.530	2.530
454.calculix	2.530	2.400	2.400	2.270	2.130	2.130	2.000	2.000	1.870	2.530	2.530	2.530	2.530
456.hmmer	2.530	2.400	2.530	2.270	2.130	2.130	2.000	2.000	1.870	2.530	2.530	2.530	2.530
458.sjeng	2.530	2.400	2.530	2.270	2.130	2.130	2.000	1.870	1.870	2.530	2.530	2.530	2.530
459.GemsFDTD	2.530	2.530	2.130	2.130	1.870	1.870	1.870	1.870	1.600	2.530	1.600	1.600	1.600
462.libquantum	2.000	2.000	1.870	1.870	1.870	1.870	1.600	1.600	1.600	1.600	1.600	1.600	1.600
464.h264ref	2.530	2.530	2.530	2.270	2.130	2.130	2.000	2.000	1.870	2.530	2.530	2.530	2.530
465.tonto	2.530	2.530	2.530	2.270	2.270	2.130	2.000	2.000	1.870	2.530	2.530	2.530	2.530
470.lbm	2.530	2.530	2.530	2.130	2.530	2.000	1.870	1.600	1.600	1.600	2.530	1.600	1.600
471.omnetpp	2.530	2.530	2.530	2.400	2.530	2.000	1.870	1.870	1.870	2.530	2.530	2.530	1.600
473.astar	2.530	2.530	2.530	2.530	2.530	2.130	2.530	1.870	1.870	2.530	2.530	2.530	2.530
482.sphinx3	2.530	2.530	2.530	2.530	2.530	2.270	2.530	2.270	2.270	2.530	2.530	2.530	2.530
AVERAGE	2.449	2.384	2.297	2.232	2.142	2.070	1.980	1.894	1.802	2.423	2.315	2.244	2.172
MAX	2.530	2.530	2.530	2.530	2.530	2.270	2.530	2.270	2.270	2.530	2.530	2.530	2.530
MIN	2.000	2.000	1.870	1.870	1.870	1.870	1.600	1.600	1.600	1.600	1.600	1.600	1.600

[10] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: making dvs practical for complex hpc applications," in *ICS '09: Proceedings of the 23rd international conference on Supercomputing*. New York, NY, USA: ACM, 2009, pp. 460–469. 2

[11] Q. Wu, M. Martonosi, D. W. Clark, V. J. Reddi, D. Connors, Y. Wu, J. Lee, and D. Brooks, "A dynamic compilation framework for controlling microprocessor energy and performance," in *MICRO 38: Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 271–282. 2

[12] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 347–358. 2

[13] R. Teodorescu and J. Torrellas, "Variation-aware application scheduling and power management for chip multiprocessors," in *ISCA '08: Proceedings of the 35th International Symposium on Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 363–374. 2

[14] K. Meng, R. Joseph, R. P. Dick, and L. Shang, "Multi-optimization power management for chip multiprocessors," in *PACT '08: Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. New York, NY, USA: ACM, 2008, pp. 177–186. 2

[15] R. Kotla, S. Ghiasi, T. Keller, and F. Rawson, "Scheduling processor voltage and frequency in server and cluster systems," in *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11*. Washington, DC, USA: IEEE Computer Society, 2005, p. 234.2. 2

[16] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," in *ISCA '09: Proceedings of the 36th annual international symposium on Computer architecture*. New York, NY, USA: ACM, 2009, pp. 314–324. 2

[17] A. Merkel and F. Bellosa, "Memory-aware scheduling for energy efficiency on multicore processors," in *In Proceedings of the Workshop on Power Aware Computing and Systems (HotPower'08)*, San Diego, CA, Dec. 7 2008, publication. [Online]. Available: http://i30www.ira.uka.de/research/publications/pm/ 2

[18] D. C. Snowdon, S. M. Petters, and G. Heiser, "Accurate on-line prediction of processor and memoryenergy usage under voltage scaling," in *EMSOFT '07: Proceedings of the 7th ACM & IEEE international conference on Embedded software*. New York, NY, USA: ACM, 2007, pp. 84–93. 2

[19] D. C. Snowdon, G. van der Linden, S. M. Petters, and G. Heiser, "Accurate run-time prediction of performance degradation under frequency scaling," in *Proceedings of the 3rd Workshop on Operating System Platforms for Embedded Real-Time Applications*, Pisa, Italy, Jul 2007. 2

[20] W. L. Bircher and L. K. John, "Analysis of dynamic power management on multi-core processors," in *ICS '08: Proceedings of the 22nd annual international conference on Supercomputing*. New York, NY, USA: ACM, 2008, pp. 327–338. 2

[21] V. George, S. Jahagirdar, C. Tong, K. Smits, S. Damaraju, S. Siers, V. Naydenov, T. Khondker, S. Sarkar, and P. Singh, "Penryn: 45-nm next generation Intel® Core™ 2 processor," in *ASSCC'07 IEEE Asian Solid-State Circuits Conference*, 2007. 3

[22] Intel, "Intel Core2 Duo Mobile Processor, Intel Core2 Solo Mobile Processor and Intel Core2 Extreme Mobile Processor on 45-nm Process. For platforms based on Mobile Intel 4 Series Express Chipset Family. Datasheet." March 2009. [Online]. Available: http://download.intel.com/design/mobile/datashts/32012001.pdf 3

[23] [Online]. Available: http://www.ubuntu.com 3

[24] SBSIF, "SMART specifiaction rev.1.1 Dec 11, 1998," [Online] Available: http://sbs-forum.org. 3

[25] " Thinkpad SMAPI kernel module version 0.40. http://tpctl.sourceforge.net/ ." 3

[26] "Perfmon2. http://perfmon2.sourceforge.net/ ." 3

[27] K. Singh, M. Bhadauria, and S. A. McKee, "Real time power estimation

and thread scheduling via performance counters," *SIGARCH Comput. Archit. News*, vol. 37, no. 2, pp. 46–55, 2008. 3

[28] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, 2006. 3

[29] M. Garrett, "Powering down," *Commun. ACM*, vol. 51, no. 9, pp. 42–46, 2008. 7

[30] S. Zhang and K. S. Chatha, "Automated techniques for energy efficient scheduling on homogeneous and heterogeneous chip multi-processor architectures," in *ASP-DAC '08: Proceedings of the 2008 Asia and South Pacific Design Automation Conference*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2008, pp. 61–66. 7