Crank It Up or Dial It Down: Coordinated Multiprocessor Frequency and Folding Control

Augusto Vega
IBM Corporation
Research Division
ajvega@us.ibm.com

Alper Buyuktosunoglu
IBM Corporation
Research Division
alperb@us.ibm.com

Heather Hanson
IBM Corporation
Systems & Technology Group
heather.l.hanson@gmail.com

Pradip Bose
IBM Corporation
Research Division
pbose@us.ibm.com

Srinivasan Ramani
IBM Corporation
Systems & Technology Group
sramani@us.ibm.com

ABSTRACT

Dynamic power management features are now an integral part of processor chip and system design. Dynamic voltage and frequency scaling (DVFS), core folding and per-core power gating (PCPG) are power control actuators (or "knobs") that are available in modern multi-core systems. However, figuring out the actuation protocol for such knobs in order to achieve maximum efficiency has so far remained an open research problem. In the context of specific system utilization dynamics, the desirable order of applying these knobs is not easy to determine.

For complexity-effective algorithm development, DVFS, core folding and PCPG control methods have evolved in a somewhat decoupled manner. However, as we show in this paper, independent actuation of these techniques can lead to conflicting decisions that jeopardize the system in terms of power-performance efficiency. Therefore, a more robust coordination protocol is necessary in orchestrating the power management functions. Heuristics for achieving such coordinated control are already becoming available in server systems. It remains an open research problem to optimally adjust power and performance management options at runtime for a wide range of time-varying workload applications, environmental conditions, and power constraints.

This research paper contributes a novel approach for a systematically architected, robust, multi-knob power management protocol, which we empirically analyze on live server systems. We use a latest generation POWER7+ multi-core system to demonstrate the benefits of our proposed new *coordinated* power management algorithm (called PAMPA). We report measurement-based analysis to show that PAMPA achieves comparable power-performance efficiencies (relative to a baseline decoupled control system) while achieving conflict-free actuation and robust operation.

MICRO '46, December 07-11 2013, Davis, CA, USA

Categories and Subject Descriptors

C.0 [**General**]: Hardware/software interfaces; C.1 [**Processor Architectures**]: Parallel Architectures; C.4 [**Performance of Systems**]: Reliability, availability, and serviceability; D.4 [**Operating Systems**]: Process Management

General Terms

Performance, Management, Measurement

Keywords

robust power management, dynamic voltage and frequency scaling, per-core power gating, multi-core systems

1. INTRODUCTION

Modern multi-core systems incorporate support for dynamic power and thermal management. These features are controlled by algorithms implemented in the system firmware-software stack, governed by real-time information like processor utilization [27, 31]. The available power control knobs have become progressively multiple: e.g. dynamic voltage and frequency control (DVFS), followed by core folding[1] and subsequently supported by per-core power gating (PCPG). As such, independently developed algorithms to actuate such knobs have evolved and been experimented with in a somewhat *decoupled* manner when it comes to first generation power-managed systems. A key advantage of decoupled power management is that individual algorithms (e.g. DVFS and PCPG) are less complex and independently verifiable. However, independent operation of DVFS and PCPG can result in interference that leads to undesirable effects on performance and/or power consumption. For example, a decoupled approach can waste precious time thrashing between states as PCPG turns cores on and off and DVFS frequency fluctuates, each separate control system striving to maintain a separate set point as they drastically affect each other's input. Separate dynamic controllers can easily miss settling into "sweet spots" for optimal control.

In this paper, we argue in favor of a *coordinated*, *in-order* actuation of DVFS and core-folding (supported by PCPG) to reduce the potential interference across the two techniques

[1]We use the term *core folding* to refer to consolidation of work into fewer cores to save power. See Section 2.

and hence to create a more robust framework for power management in current generation chip multiprocessors (CMPs). It is important to note that for this class of servers, "more robust" pertains to high performance and high availability under a wide range of conditions, and does not guarantee more power savings.

The argument for *coordinated* power management is addressed in prior works [9, 10, 28]. However, to the best of our knowledge, our work is the first published research evaluating these techniques on a real hardware testbed in which *both* DVFS and PCPG are operable. Specifically, the key contributions of this work are:

- We expound on the potential drawbacks of the decoupled DVFS and PCPG actuation approach. These unwanted effects are, in part, the result of the individual operation of the DVFS and PCPG algorithms.
- We present a detailed measurement-based evaluation and comparison of product-level power management policies available in a modern multi-core system.
- We present PAMPA, a *Power-Aware Management of Processor Actuators* algorithm, capable of attaining power-performance efficiency comparable to (although not necessarily better than) the most aggressive power management approach, while achieving conflict-free actuation of the power management activities. PAMPA implements a *coordinated* control of the DVFS and PCPG knobs, which reduces the potential interference between them. Even though it jointly operates DVFS and PCPG, PAMPA is still an extremely simple algorithm which just requires access to on-line core-level utilization information. PAMPA does not require any kind of off-line pre-processing and performs very light-weight computation to make its decisions.
- We implement and evaluate PAMPA in a real commercial server, a 16-core POWER7+ system. Therefore, results presented in this work are not based on any simplifying assumptions and include all possible overheads and constraints associated with the actuation of DVFS and PCPG.

PAMPA's ultimate goal is to dynamically operate DVFS, core folding and PCPG in a coordinated fashion, guided by run-time processor and workload variations. In contrast to most of the utilization-based power management algorithms deployed in today's systems, PAMPA adopts a novel approach to guide its decisions —it dynamically "detects" situations in which an application may be bound by *single-thread-performance* or *throughput* and actuates the power management knobs accordingly. When evaluated on a real AIX/POWER7+ system, PAMPA exhibits power-performance efficiencies for a set of throughput-oriented and single-thread-performance-sensitive workloads comparable to the most aggressive power management approach. At the same time, PAMPA avoids excessive performance degradation in cases where the independent DVFS and PCPG operation results in conflicting decisions.

The remainder of the paper is organized as follows. Section 2 introduces background concepts and describes our baseline power management setting. Section 3 outlines our experimental methodology. Section 4 illustrates the problem of destructive interference between DVFS and PCPG actuators with measured results. Section 5 presents distinct scenarios for optimizing latency and/or throughput. Section 6 presents the *Power-Aware Management of Processor Actuators* algorithm (PAMPA), which is evaluated for the SPECpower benchmark in Section 7.1 and for highly-parallel scientific applications in Section 7.2. Finally, Section 8 provides a summary of related work and Section 9 presents our conclusions.

2. BACKGROUND

Dynamic voltage and frequency scaling (DVFS) is a key function supported in first generation power-managed systems [27, 34]. It provides an effective mechanism for power and thermal management, by enabling the system to adjust voltage and/or frequency dynamically. Core folding constitutes a system software-level technique in which one or more processor cores are temporarily removed from the set of "usable" cores and their assigned tasks are reallocated to other (not folded) cores. Without the support of per-core power gating (PCPG), core folding alone is able to save considerable active power through built-in features like clock-gating or just turning clocks off for idle cores. Later, as PCPG became available [17, 33], core folding supported by PCPG was able to garner significantly more savings in power. Folded cores are not accessible from the operating system scheduler standpoint, and hence can be power gated to virtually eliminate all power (leakage + active).

In this work, we use a current generation POWER7+ [33] based system as the evaluation platform for quantifying the benefit of the PAMPA algorithm. Figure 1 shows the interaction and responsibilities of the EnergyScale microcontroller [7, 12, 13], Hypervisor, operating system (OS) and processors in a typical POWER7+ system. DVFS is automatically actuated when the microcontroller is set to the Dynamic Power Save (DPS) mode. In our study, we build upon a previous generation DVFS algorithm (*core-pool*) [29] to construct our own baseline DVFS control logic. The core-pool algorithm varies processor frequency and voltage based on the utilization of the system's processors. It employs two utilization thresholds: $threshold_{active}$ determines which processor cores are idle (those with utilization values below this threshold are considered inactive) and $threshold_{slack}$ determines which cores exhibit enough intermittent idle time (slack) among the active ones. The algorithm then computes the ratio of the core count below the slack threshold to the total active cores, which is used to determine whether to increase, decrease, or do not change the processor frequency —decisions are made every 32 ms [15, 23].

In addition to DVFS, EnergyScale-ready OSs can also take advantage of core folding supported by PCPG to virtually eliminate power consumption in idle cores, when feasible. In this work, we study the efficacy and robustness of coordinated management protocols involving DVFS and core folding-with-PCPG, assuming an operational environment as depicted in Figure 1. We construct a baseline core folding/PCPG control algorithm which dynamically scales the number of cores needed in proportion to the overall processor utilization. We follow an approach similar to the one followed in the AIX OS [16]. After determining the number of needed cores, the ones that remain unused (or "empty") are folded and the POWER Hypervisor [7] is notified to exploit special purpose idle states available on selected POWER7/POWER7+ systems. In our modeled experiments, the Hypervisor switches each folded core to sleep mode, which powers off the core as well as its private L2

cache [33]. Core folding-related decisions are made every second by the OS scheduler. For the rest of the paper, we refer to core folding and its consequent PCPG action as "PCPG".

In this work, we purposely deploy the two algorithms that constitute our decoupled power management baseline at different levels —the core folding and PCPG algorithm at system software level, and the DVFS algorithm at the EnergyScale microcontroller level— in order to mimic a decoupled power management scenario. It is important to note that these are expert-driven algorithms highly-tuned to improve power-performance efficiency. In other words, our decoupled power management baseline constitutes a highly power-performance efficient scenario and it is considered the most *aggressive* approach.

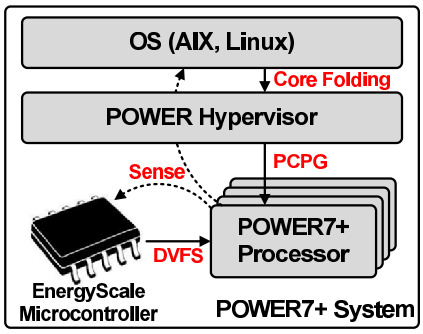


Figure 1: Organization and interaction of the power management components in a POWER7+ system.

3. METHODOLOGY

In this work, we present the *Power-Aware Management of Processor Actuators* algorithm (PAMPA) which aims to provide robust chip-level power management by coordinating the operation of DVFS, core folding and PCPG. In this context, our experimental system consists of an IBM POWER7+ system where the PAMPA algorithm operates at OS level (Figure 2). PAMPA collects on-line hardware events information from the PM_RUN_CYC performance counter available in POWER7+ [26]. PM_RUN_CYC counts the non-idle processor cycles at physical thread-level. In other words, it filters out those processor cycles during which a particular physical thread is idle and, therefore, constitutes a suitable proxy to estimate physical thread-level utilization. We then average the utilization of all physical threads in each core to estimate core-level utilization. PAMPA also connects to the EnergyScale microcontroller through the IBM Automated Measurement of Systems for Temperature and Energy Reporting software (AMESTER) [12, 22], to adjust the processor-level frequency (DVFS). In addition, PAMPA also actuates core folding at OS level, with its consequent PCPG action at Hypervisor level. It is important to note that the POWER7+ system used in our experiments also supports per-core dynamic frequency scaling (DFS). In this work, however, we consider processor-level DVFS (along with PCPG) and defer the use of per-core DFS to future work.

The system has two IBM POWER7+ processors [33] running at 4.088 GHz (nominal) and 64 GB of DDR3 SDRAM. Processor frequency can be increased above its nominal value —the *turbo mode* [12]. Each POWER7+ processor is composed of eight cores, each one capable of four-way simultaneous multi-threading (SMT) operation. Therefore, the two POWER7+ processors provide a total of 64 hardware (or physical) threads. Each core has access to a private 256-KB L2 cache and to a local 10-MB L3 region. Eight L3 regions constitute a 80-MB on-chip L3 cache (local L3 regions provide low-latency access to the cores). In addition to its private L2 cache, a core can also obtain data from other cores' L2 and L3 caches through the coherence fabric.

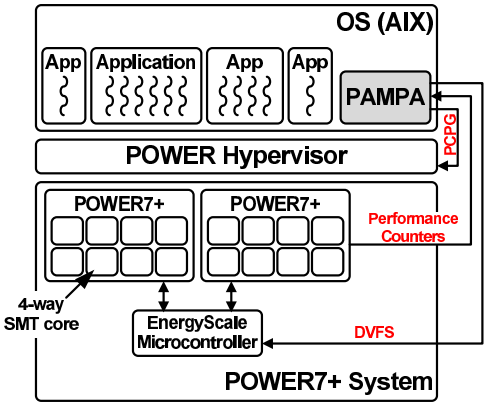


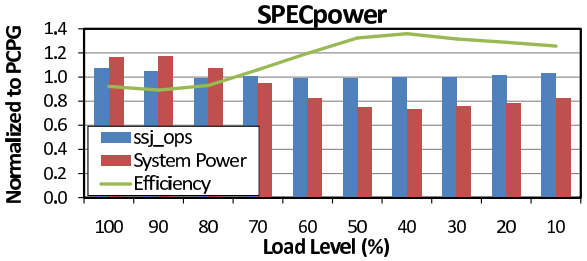
Figure 2: Experimental system.

The platform runs IBM AIX 7.1 OS. We use the SPECpower_ssj2008 benchmark suite [1], a subset of the PARSEC 2.1 benchmarks [4] and the CoMD and Lulesh applications [11, 20]. PARSEC does not officially support the AIX operating system, so only a handful of the benchmarks were run on the AIX/POWER7+ system. We verify the repeatability of our study by experimenting with multiple runs for each benchmark.

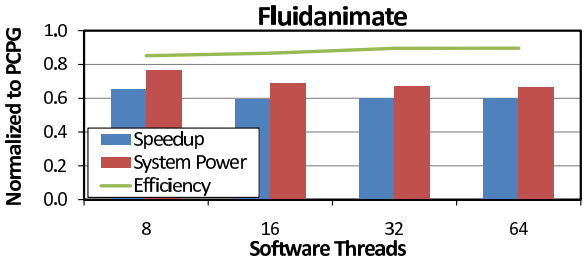
4. DECOUPLED DVFS AND PCPG DRAWBACKS

Today's system designers advocate for decoupled power management approaches in which the responsibility of power management algorithms is shared between the system software (e.g. the operating system) and the power management firmware, operating independently of one another. There are two main arguments in favor of the use of decoupled power management policies [24]. First, individual DVFS and PCPG control algorithms can be less complex compared to an algorithm which jointly operates both knobs. In addition, DVFS and PCPG exhibit different transition time overheads, which make these techniques suitable to track short-term and long-term trends, respectively. Therefore, both techniques require independent actuation to exploit their different timescale granularities. In this scenario, it is essential to define clear *protocols* to guide the actuation of the power management techniques. For example, in the presence of variations in processor utilization, should we begin scaling frequency or would it be better to power cores on/off? To illustrate the implications of this kind of decisions, we execute the SPECpower benchmark [1] and one multi-threaded application (Fluidanimate) from the PARSEC benchmark suite [4] on our AIX/POWER7+ testbed platform. For the sake of this study, we construct a baseline scenario where both DVFS and PCPG operate independently in a decoupled way. As we explain in Section 2, we build two algo-

rithms, one for DVFS control and one for PCPG control. Figure 3a presents the performance (number of operations per second or *ssj_ops*), system power and power-performance efficiency (ssj_ops/Watt) for the different SPECpower load levels (100% to 10%) when this benchmark is executed on our AIX/POWER7+ system with both DVFS and PCPG enabled. The results are normalized to the same execution in which only PCPG is enabled. When both power reduction techniques are accessible, the system power is significantly reduced as the load decreases, with no performance degradation. In other words, the operation of DVFS *and* PCPG improves the power-performance efficiency for the SPECpower benchmark over the same scenario where only PCPG is enabled. Figure 3b presents the performance (execution time speedup), system power and power-performance efficiency for the Fluidanimate application when it is executed with different numbers of threads. As in the SPECpower case, the results correspond to the execution with both DVFS and PCPG enabled (using our own baseline algorithms) and are normalized to the case where only PCPG is enabled. In this case, we observe that the availability of DVFS (in addition to PCPG) does not improve power-performance efficiency. Even worse, it severely degrades performance in all the cases (8, 16, 32 and 64 threads). Clearly, the SPECpower and Fluidanimate benchmarks present different characteristics which mislead the power management algorithms in the latter case.



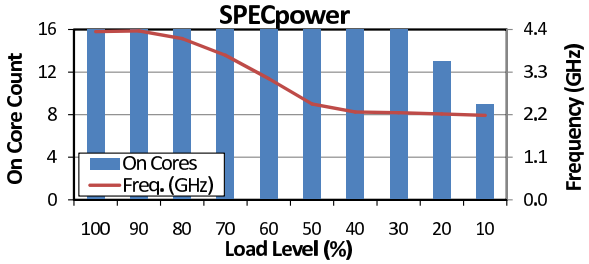
(a) SPECpower benchmark



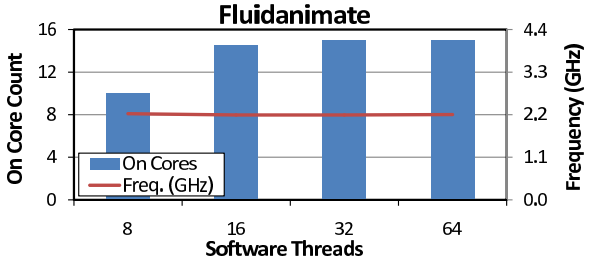
(b) Fluidanimate benchmark

Figure 3: SPECpower and Fluidanimate benchmark executions on a 2 × POWER7+ based system (16 cores total). DVFS, core folding and PCPG capabilities are enabled. The results are normalized to the same system in which only PCPG is enabled (no DVFS). SPECpower executions are calibrated to the same maximum sustainable throughput.

To understand the reasons behind the behaviors presented in Figures 3a and 3b, we monitor how the number of turned-on cores and processor frequency change during the execution of SPECpower and Fluidanimate as a consequence of our baseline, decoupled DVFS and PCPG algorithms operation. Figure 4a presents the number of turned-on cores and processor frequency for the SPECpower run. As we can observe, the system starts scaling frequency down as the load decreases. This is done by the DVFS algorithm, independently of the PCPG algorithm operation. The frequency reduction keeps cores' utilization close to 100% (not shown in the figure), which prevents the PCPG algorithm from kicking in. At the 20% load level, the cores' utilization is so low that the PCPG algorithm starts folding and power gating cores. Figure 4b presents the number of turned-on cores and processor frequency for the Fluidanimate run. This application exhibits a high level of parallelism. As a result, the OS balances the running application threads across as many cores as possible. From the standpoint of the PCPG algorithm, the processor is quite highly utilized, and therefore it keeps as many cores turned-on as needed. Because the PCPG and DVFS algorithms operate independently of one another, they may also use different thresholds (or even different logic) to guide their decisions. As a result, the DVFS algorithm "sees" lower core utilization levels compared to the PCPG algorithm and consequently scales frequency down — in this case to its minimum value (2.2 GHz). This action has the undesirable consequence of a severe performance degradation. As we can observe from this simple experiment, the independent operation of DVFS and PCPG opens up concerns about the robustness of such *decoupled* power management approaches.



(a) SPECpower benchmark



(b) Fluidanimate benchmark

Figure 4: Number of turned-on cores and processor frequency for the SPECpower and Fluidanimate executions. This is the result of the operation of DVFS and core folding with PCPG on our AIX/POWER7+ system.

5. PERFORMANCE *AND* THROUGHPUT AWARENESS

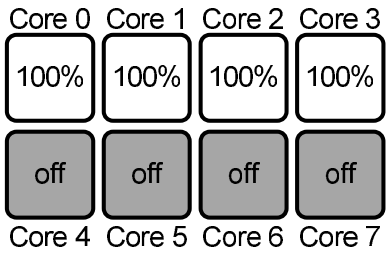
The two main arguments in favor of the use of decoupled power management policies are the simplicity of independent algorithms and the possibility of exploiting their different transition time overheads. In this work, we approach this matter from a different angle, which is not covered in

prior art: DVFS and PCPG are also suitable to differently exploit power management depending on if an application's current execution phase is *single-thread-performance bound* or *throughput bound*. To illustrate this, we define the following three distinct scenarios when the processor cores' utilization is observed at run-time:

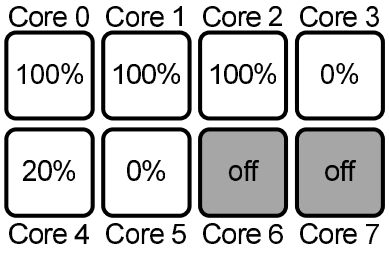
- **Scenario A: all the turned-on cores are highly utilized.** This situation indicates that the application may be either traversing a single-thread-performance bound or throughput bound execution phase. Either scaling frequency up or turning on an extra core may lead to performance or throughput improvement, respectively. Empirical evidence shows that if all the turned-on cores are highly-utilized, applications benefit more from unfolding an extra core than from scaling frequency up. Opening up cores will likely provide better performance by allowing the OS to distribute threads to additional cores. Increasing frequency alone cannot improve performance if the core is overloaded. This scenario is presented in Figure 5a where, on a hypothetical 8-core processor, four cores are turned on and are highly utilized and four cores are turned off.
- **Scenario B: some turned-on cores are highly utilized.** In this case, the application is likely bound by single-thread performance. Otherwise, if the application were bound by throughput, we should expect the other (not fully utilized or empty) cores to become utilized by the application. Scaling frequency up should lead to performance improvement. This scenario is presented in Figure 5b where three cores are highly utilized, one core is low utilized, two cores are empty and two cores are turned off.
- **Scenario C: all the turned-on cores are low utilized or empty.** Finally, in this case the cores exhibit enough intermittent idle time (slack) due to, for example, intensive I/O activity. Either scaling frequency down or turning one or more cores off should lead to chip power reduction with minimal (or no) performance impact. Empirical evidence shows that reducing frequency first and then powering cores off lead to the best results when all the turned-on cores are low utilized or empty. Frequency and voltage decrease provide the best power savings due to approximately cubic reduction in power and their lower actuation latency compared to PCPG, especially if that decision is wrong and has to be reversed quickly. This scenario is presented in Figure 5c where five cores are low utilized, two cores are empty and one core is turned off.

These scenarios are easier to handle when the knobs are jointly available, because they have to be coordinated following a particular order (*protocol*) regardless of their different timescale granularities. For example, under a decoupled power management control, the OS may decide to unfold and turn on some cores in scenario A. This decision may result in a reduction in utilization which leads the processor to scale the frequency down, potentially hurting performance.

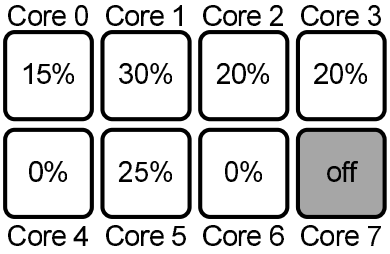
This simple analysis of some potential scenarios that we can observe on a chip multiprocessor is leveraged to help DVFS- and PCPG-related decision making and constitutes the basis of the *Power-Aware Management of Processor Actuators* algorithm (PAMPA) presented in the next section.



(a) Scenario A



(b) Scenario B



(c) Scenario C

Figure 5: Plausible scenarios that can be leveraged to help power management decisions.

6. PAMPA ALGORITHM

Based on the core utilization scenarios described in Section 5, we devise the *Power-Aware Management of Processor Actuators* (PAMPA) algorithm. PAMPA's main characteristic is the *coordinated* operation of the available power management techniques (e.g. DVFS and PCPG) to reduce the potential interference between them. At run-time, PAMPA collects core-level utilization information to determine if the running application is traversing any of the scenarios described in Section 5. Based on this characterization, PAMPA decides if DVFS and/or PCPG have to be actuated and how. Core-level utilization is estimated using the PM_RUN_CYC performance counter available in POWER7+ [26], as we explain in Section 3.

PAMPA is an extremely simple closed-loop control algorithm, which is triggered every T milliseconds. It can be deployed either at software or hardware level with the only condition that it requires a holistic view of the processors as well as access to the power management actuators (DVFS, PCPG). This means that in the case of a virtualized environment, with multiple OSs sharing a pool of processors, PAMPA has to be implemented in the software layer closest to the processors. The testbed used in this work is an AIX/POWER7+ system, in which we envision PAMPA implemented at Hypervisor level, the virtualization layer in IBM's Power Systems. Due to lack of access to propri-

etary Hypervisor code, our research PAMPA prototype is developed in user space with complete access to the power management knobs and processor sensors. Figure 6 presents PAMPA's flowchart, where the parts of the algorithm that handle the three different scenarios are demarcated by dashed boxes.

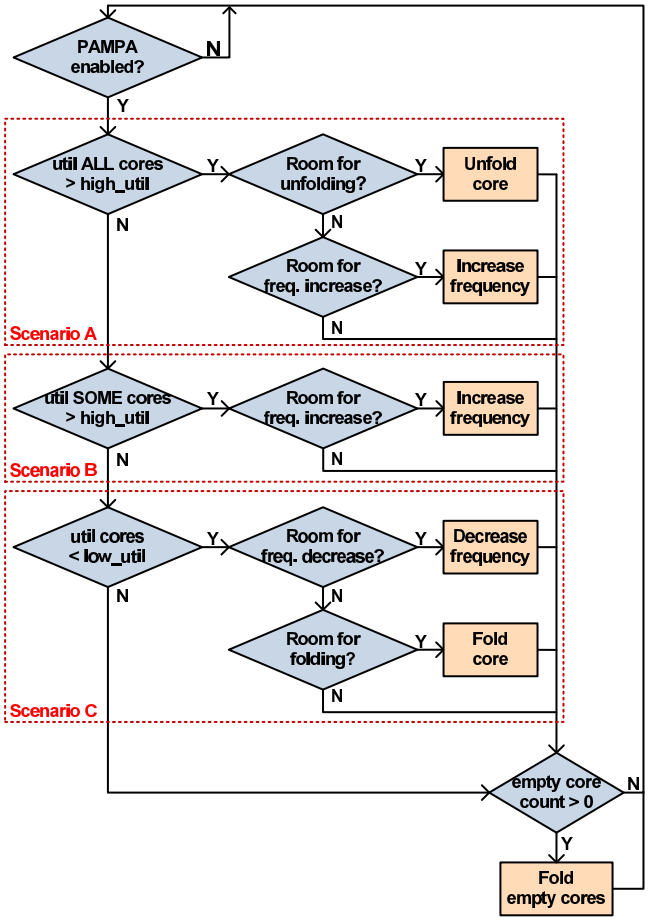


Figure 6: PAMPA algorithm.

6.1 Scenario A

The top part of the flowchart presented in Figure 6 evaluates if *all* the turned-on cores are highly utilized. If this is the case, the application may be either traversing a single-thread-performance bound or throughput bound execution phase. In other words, there are two choices: to scale frequency up or to unfold and turn on one or more cores. PAMPA opts for unfolding and turning on cores over frequency scaling —if all the turned-on cores are highly-utilized, applications benefit more from unfolding an extra core than from scaling frequency up. If all the cores are already unfolded and turned-on, PAMPA increases frequency instead.

If after the unfolding decision, the just enabled cores remain empty or low utilized, the application was not throughput bound but single-thread performance bound. In other words, *scenario A* becomes *scenario B* (where some, but not all, the cores are highly utilized) and PAMPA will actuate accordingly in its next iteration.

6.2 Scenario B

The middle part of the flowchart presented in Figure 6 evaluates if *some*, but not all, the turned-on cores are highly utilized. In this case, PAMPA assumes that the application is traversing a single-thread-performance bound execution phase and decides to increase frequency (if frequency is not already at its maximum value). Otherwise, if the application were bound by throughput, we should expect the other (not fully utilized or empty) cores to become utilized by the application.

6.3 Scenario C

The bottom part of the flowchart presented in Figure 6 evaluates if all the turned-on cores are either lightly utilized or empty. In this case, PAMPA assumes that it is safe to reduce frequency or, if frequency is already at its minimum value, to fold and power off cores. The consideration of *scenario C* is one of the pillars of PAMPA's robustness, because frequency is reduced and cores are folded only if *all* the turned-on cores have low utilization. In other words, PAMPA prevents frequency reduction and core folding even if just one physical thread in the processor presents high utilization to avoid severe performance degradation.

After evaluating the three scenarios, PAMPA also folds and turns off empty cores. This action is complementary and independent from the treatment of scenarios A, B and C. In PAMPA's current implementation, all but one empty cores are turned off. The benefits of keeping one empty core "alive" are twofold: it helps to absorb temporary utilization peaks and, more important, it prevents PAMPA from entering into unstable situations. For example, turning *all* empty cores off may convert scenario B into scenario A and, after that, PAMPA may convert scenario A into scenario B again by unfolding cores. This *ping-pong* behavior between two scenarios is "broken" by keeping one empty core turned on.

6.4 PAMPA Adjustment Parameters

The efficiency of the PAMPA algorithm can be adjusted based on the following configuration parameters:

Monitoring interval (T): PAMPA is triggered (and core-level utilization sensors are read) every T milliseconds.

History length (H): PAMPA considers the last H core-level utilization samples to represent the utilization of each core. If $H = 1$, PAMPA takes just the most recent sample to represent core-level utilization. If $H > 1$, the average of the last H samples is used. The smaller the H value, the faster PAMPA can make decisions. However, too small history values can lead to wrong decisions due to lack of information to characterize the utilization of each core. On the other hand, large H values can help to smooth application behavior, reducing the risk of wrong decisions due to utilization outliers.

Enabling threshold (K): even if PAMPA is triggered every T milliseconds, the algorithm executes only during stable execution phases. To determine if the current execution phase is stable, PAMPA computes the average and standard deviation of the last H core-level utilization samples: $core\_i\_util_{avg}$ and $core\_i\_util_{stdev}$, respectively. If $core\_i\_util_{stdev} \leq K \times core\_i\_util_{avg}$ for each core i, PAMPA assumes that the application is traversing a stable computation phase and, therefore, it is safe to make decisions. It prevents PAMPA from making decisions during phase tran-

Parameter	Value	Range Explored
T	1000ms	Smaller values are not explored due to infrastructure limitations to collect power readings at finer granularities.
H	3	1, 2, 3, 4, 5
K	0.2	0.0, 0.1, 0.2, 0.4, 0.6, 0.8
low_util	0.70	0.70, 0.72, 0.74, 0.76, 0.78, 0.80
high_util	0.80	0.80, 0.82, 0.84, 0.86, 0.88, 0.90
empty	0.10	0.10

Table 1: PAMPA configuration parameters and ranges explored. "Value" column shows the ones used in the experiments, which maximize the power-performance efficiency.

sitions, which are wrong in many cases. Too small K values will keep PAMPA silent most of the time. On the other hand, too large K values may lead to PAMPA decisions even during phase changes.

Low utilization threshold (*low_util*): PAMPA considers that a core is low utilized if its utilization is smaller than this threshold.

High utilization threshold (*high_util*): PAMPA considers that a core is highly utilized if its utilization is larger than this threshold.

Empty core threshold (*empty*): In addition, PAMPA considers that a core is "empty" if its utilization is lower than a fixed threshold, which in the current implementation is 10%.

We explore all the possible combinations of the adjustment parameters to determine the set which maximizes the power-performance efficiency (energy per instruction) averaged across all the PARSEC benchmarks, the Lulesh and CoMD applications and the SPECpower benchmark. Table 1 lists the parameters used for the experiments of Sections 7.1 and 7.2.

7. PAMPA EVALUATION

7.1 SPECpower Benchmark

We execute PAMPA in the context of the SPECpower benchmark [1], which combines power and performance measurements. A SPECpower run goes through a calibration phase to determine the maximum sustainable throughput, and a running phase at different percentages of the calibrated throughput (or *load levels*): 100%, 90% ... 10%.

SPECpower consists of independent data warehouses (implemented as Java threads) which receive transaction requests from users. Each warehouse receives transactions generated based on a probability distribution; therefore, load levels across warehouses may be different at any given time. We evaluate PAMPA in a SPECpower configuration with 16 Java Virtual Machines (JVMs), 4-warehouse each, for a total of 64 warehouses.

For the study presented in this section, we compare the proposed PAMPA algorithm against four different power management settings:

- **No power management** — In this configuration, we disable both PCPG and DVFS in the POWER7+ system under evaluation. This scenario constitutes the baseline.
- **Only PCPG** — Only the core folding and PCPG algorithm is enabled, which dynamically adjust the number of needed cores based on the processors utilization. After one or more cores are folded, the Hypervisor is also notified to power gate the just-folded cores.
- **Only DVFS** — Only the DVFS algorithm is enabled, which dynamically adjust the processors frequency, following an approach similar to the core-pool algorithm [29].
- **Both PCPG and DVFS enabled** — Both algorithms are enabled and can make independent DVFS- and PCPG-related decisions. We purposely deploy the algorithms as separate *agents* in the POWER7+ system in order to mimic a decoupled power management approach. These are expert-driven algorithms highly-tuned to improve power-performance efficiency. Therefore, the PCPG+DVFS decoupled approach constitutes a highly (power-performance) efficient scenario.

It is important to note that we calibrate the different SPECpower executions to the same maximum sustainable throughput to make a fair comparison among them.

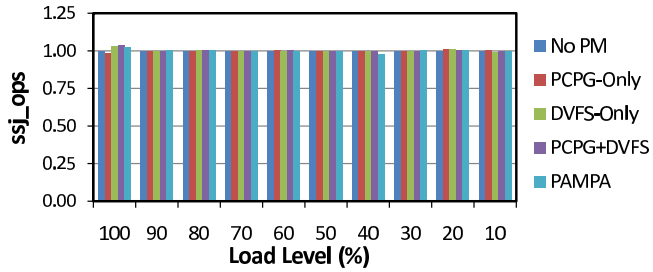
Figure 7a presents the actual throughput for the different load levels in terms of SPECpower operations per second (*ssj_ops*). The throughput attained by the different settings is similar, with exception of the 100% load level. In this case, DVFS-Only, PCPG+DVFS and PAMPA slightly improve throughput with respect to the baseline by scaling frequency above its nominal value (Figure 7b).

It is important to note that PAMPA's main objective is not (necessarily) to outperform the most aggressive, decoupled PCPG+DVFS technique in terms of power-performance efficiency. PAMPA's ultimate goal is, instead, to attain similar efficiency levels as PCPG+DVFS while fixing those cases where the independent PCPG and DVFS operation leads to conflicting decisions. Although PCPG+DVFS does not exhibit conflicting decisions for SPECpower, in the next two sections we show lack of robustness in the PCPG+DVFS operation for some applications.

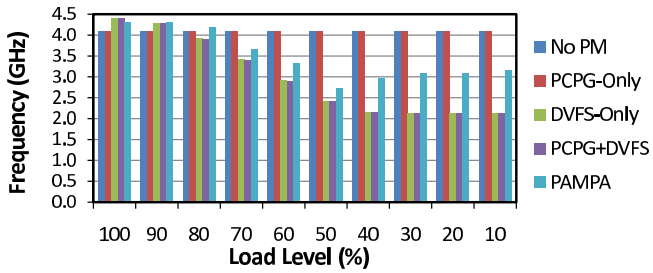
7.2 Highly-Parallel Scientific Applications

The previous section evaluates PAMPA for the SPECpower benchmark. We show that PAMPA can attain power-performance efficiency levels similar to the most aggressive, decoupled PCPG+DVFS algorithm. In this section, we present the counterpart evaluation for the PARSEC applications. PARSEC applications are highly-parallel workloads representative of next-generation shared-memory programs for chip-multiprocessors [4]. PARSEC does not officially support the AIX OS, so only the Blackscholes, Canneal, Fluidanimate and Streamcluster applications could be run on the AIX/POWER7+ system. The PARSEC benchmarks are multi-threaded applications, in which the number of software threads is specified at the onset of the execution. In this work, we execute them with 8, 16, 32 and 64 threads.

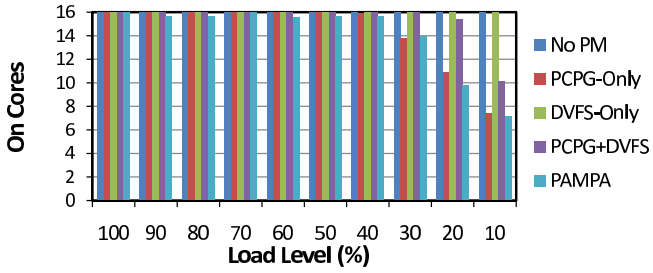
Figure 8 presents the total execution time, average system power consumption, energy per instruction (EPI), average processor frequency and average number of turned-on cores when the PARSEC applications are executed under the supervision of the evaluated power management settings. The results corresponding to each application are organized column-wise. Each chart shows the results for the 8-, 16-,



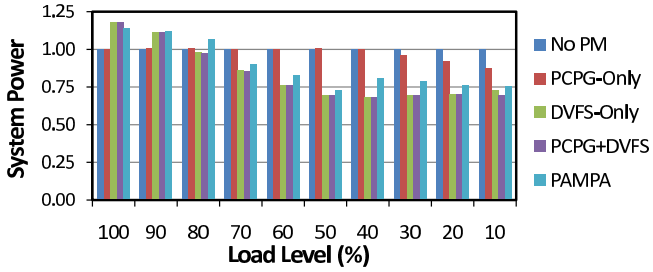
(a) SPECpower operations per second (*ssj_ops*)



(b) Processor frequency



(c) Number of turned-on cores



(d) System power consumption

Figure 7: Comparison of four different power management settings against the PAMPA algorithm. The evaluated settings are: a baseline where no power management is performed ("No PM"), a scenario with just OS-directed PCPG enabled ("PCPG Only"), a scenario with just processor-directed DVFS enabled ("DVFS Only"), and a scenario with both knobs enabled ("PCPG+DVFS"). The figure presents the SPECpower operations per second (*ssj_ops*), processor frequency, number of turned-on cores and system power consumption for the SPECpower_ssj2008 benchmark. The results are normalized to the baseline ("No PM").

32- and 64-thread cases, normalized to the baseline (No-PM). For Blackscholes and Canneal, all the settings exhibit similar execution times. However, the ones that operate core folding and PCPG (PCPG-Only, PCPG+DVFS and PAMPA) manage to lower system power consumption significantly with respect to the baseline. Blackscholes and Canneal are applications that spend a significant portion of the time with few executing threads. Core folding (in conjunction with PCPG) better exploits this behavior. As a result, PCPG-Only, PCPG+DVFS and PAMPA show better efficiencies (i.e. lower EPI values) with respect to the baseline. We observe in Figure 8 that these three settings are capable of executing Blackscholes and Canneal at nominal frequency or higher with just five or six cores on average.

For Fluidanimate and Streamcluster, DVFS-Only and PCPG+DVFS exhibit unacceptable performance degradations. When they are compared against the baseline, DVFS-Only increases execution time (on average) 73% for Fluidanimate and 26% for Streamcluster, while PCPG+DVFS increases execution time 89% for Fluidanimate and 49% for Streamcluster. Even though they significantly prune system power, the severe performance degradation results in poor efficiency for Fluidanimate (i.e. EPI values larger than the baseline). This undesirable behavior is the result of an unexpected frequency reduction in these cases. We say *unexpected* because both Fluidanimate and Streamcluster are applications that run several CPU-intensive threads most of the time (in contrast to Blackscholes and Canneal, which spend a significant portion of the time with few threads). Therefore, we expect processor frequency to be kept at its highest possible value. Instead, Figure 8 shows that DVFS-Only and PCPG+DVFS lower frequency to its minimum value. This problem is fixed by PAMPA, which "detects" that these applications are single-thread performance bound and keeps frequency at its maximum value. In other words, Fluidanimate and Streamcluster fall into the Scenario B (Section 5) which leads PAMPA to increase frequency to alleviate single-thread performance limitations. In the Fluidanimate case, PAMPA lowers EPI (on average) with respect to the most aggressive, decoupled approach (PCPG+DVFS). In the Streamcluster case, EPI efficiency is similar between PAMPA and PCPG+DVFS. However, in all the cases PAMPA achieves such efficiency by avoiding severe performance degradation.

Blackscholes and Canneal represent applications with opportunities for power reduction, particularly because they spend part of the execution with few running threads. On the other hand, Fluidanimate and Streamcluster are good examples of applications with little or no margin for power savings. Because of that, Fluidanimate and Streamcluster constitute excellent vehicles to put algorithms to the test. For example, PCPG+DVFS presents significant system power reductions for the four applications. However, PCPG+DVFS is "broken" in terms of performance for Fluidanimate and Streamcluster. In these cases, one may consider running the applications with no power management support to avoid any performance impact. Indeed, the No-PM setting keeps performance "healthy" at the expense of no power savings. This is where the potential of our PAMPA algorithm can be better appreciated. It manages to keep performance at acceptable levels most of the time, while still providing power benefits whenever possible (Blackscholes and Canneal cases).

In addition to the PARSEC applications, we also evaluate the Lulesh and CoMD benchmarks parallelized with Message Passing Interface (MPI) support. Lulesh is the Livermore Unstructured Lagrange Explicit Shock Hydrodynamics

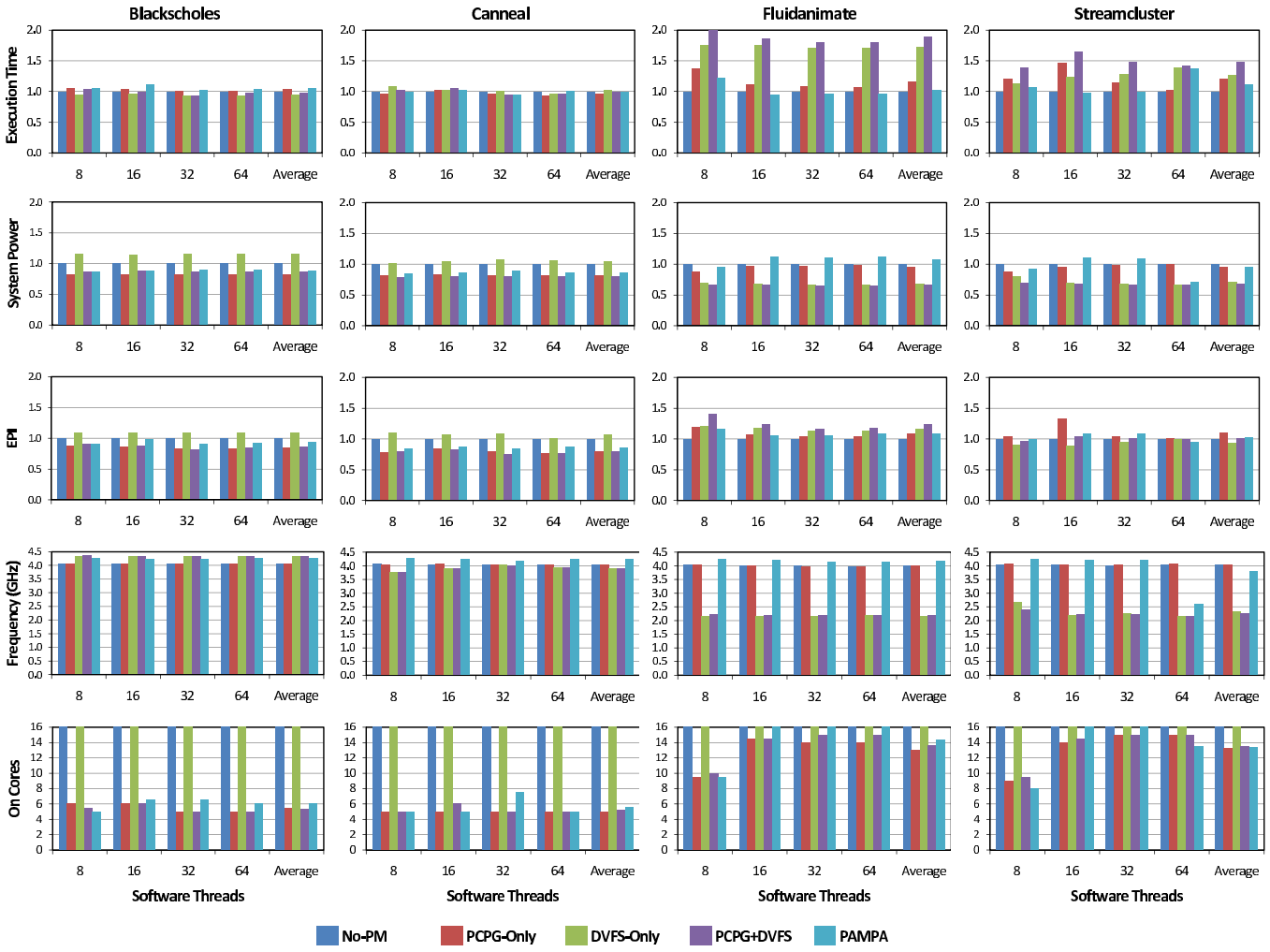


Figure 8: Comparison of four different power management settings ("No-PM", "PCPG Only", "DVFS Only" and 'PCPG+DVFS") against the PAMPA algorithm, for four PARSEC applications executed with 8, 16, 32 and 64 software threads. The results are normalized to the baseline ("No PM").

proxy application [20]. CoMD is a reference implementation of typical classical molecular dynamics algorithms and workloads [11].

Figure 9 presents the total execution time, average system power consumption, energy per instruction (EPI), average processor frequency and average number of turned-on cores when Lulesh and CoMD are executed under the supervision of the evaluated power management settings. The results are normalized to the baseline (No-PM). We evaluate Lulesh with 1, 8, 27 and 64 MPI processes[2] and CoMD with 1, 4, 8 and 16 MPI processes. For Lulesh with 8, 27 and 64 processes, the DVFS-Only and PCPG+DVFS settings exhibit performance degradations larger than 50% with respect to the baseline. This can be explained by observing the frequency, which is reduced to nearly its minimum value by DVFS-Only and PCPG+DVFS. Even though system power is significantly lowered due to the frequency reduction, the execution time increase degrades the EPI, very significantly in some cases (e.g. PCPG+DVFS consumes 75% more energy per instruction for the 8-process Lulesh case than the baseline). In contrast, PAMPA is able to keep frequency at its maximum value given that Lulesh is also a single-thread performance bound application which falls into the Scenario B (Section 5)). At the same time, PAMPA strives to aggressively power gate cores whenever possible (e.g. Lulesh with one and eight processes) to reduce power consumption.

In the case of CoMD, all the power management settings exhibit similar performance levels. In particular, DVFS-Only and PCPG+DVFS do not present the lack of robustness exhibited for the Lulesh application. In terms of efficiency, PAMPA presents EPI values comparable to the most aggressive, decoupled PCPG+DVFS algorithm. Still PAMPA is more effective at keeping frequency at its maximum value which results in a 16% execution time reduction with respect to PCPG+DVFS for the 8-process CoMD case.

Lulesh and CoMD are two more examples of highly parallel applications in which PAMPA provides conflict-free robust operation, while keeping power-performance efficiency levels comparable to the PCPG+DVFS algorithm.

[2]Lulesh requires a cubic number of MPI processes.

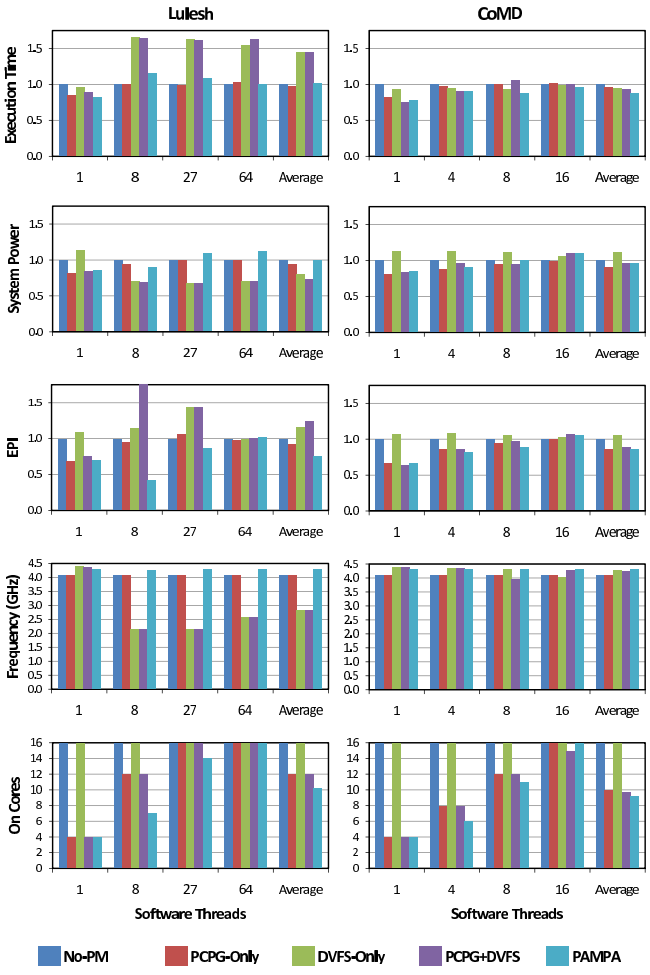


Figure 9: Comparison of four different power management settings ("No-PM", "PCPG Only", "DVFS Only" and 'PCPG+DVFS") against the PAMPA algorithm, for the Lulesh and CoMD applications. The results are normalized to the baseline ("No PM").

8. RELATED WORK

The study of the DVFS and PCPG techniques in the context of CMPs has become more popular in the last years as these "knobs" became available in commercial processors [6, 18, 33]. However, to the best of our knowledge, our work is the first one evaluating these techniques on a real hardware testbed in which *both* DVFS and PCPG are operable. Other works rely on simulations and/or analytical models to estimate the benefit of power management algorithms that use one or more control knobs.

Few prior works address the *coordination* angle of the power management techniques. Among them, Raghavendra *et al.* [28] propose a power management solution for power management coordination in the context of servers and enterprise applications. As in our case, Raghavendra *et al.* also identify the lack of power management coordination as a potentially dangerous problem. Their solution is based on a nested structure of multiple feedback controllers at various levels. Our work is different than Raghavendra *et al.*'s because we focus on processor-level (instead of server-level) power management, PAMPA does not require off-line preprocessing, and our evaluation is fully executed on a real system. Devadas *et al.* [10] tackle the problem of energy management coordination by presenting an algorithm that, first, assigns cores to applications *statically* and, then, manages run-time power consumption by exploiting core idle states. Their results are also based on simulations and focus just on real-time applications. Deng *et al.* [9] present CoScale, a method to coordinate CPU and memory subsystem DVFS under performance constraints. Similar to our work, CoScale relies on execution profiling of core and memory access performance, by using performance counters.

The Advanced Configuration and Power Interface (ACPI) constitutes a standard which enables OS-directed configuration, power management, and thermal management of platforms [2, 31]. ACPI provides support of the power management activities in a system through a set of *states* and the transitions between them. Intel's Sandy Bridge microarchitecture is an example of a chip which power management capabilities can be operated through ACPI. In this case, the OS generates requests —through the available ACPI states— which are eventually honored by the Package Control Unit (PCU) embedded in the chip [30]. ACPI represents the efforts to provide *coordination* of the power management activities in a system insofar as the OS has exclusive control of power management and device configuration. However, in a virtualized environment where multiple OSs share hardware resources (e.g. processors), different OS instances may simultaneously instruct different ACPI state transitions. In some cases, these transitions may be antagonistic and, even worse, fool each other. Therefore, it is mandatory to adopt an extra level or coordination residing as close to the hardware under management as possible. Therefore, the PAMPA algorithm —which is envisioned to be deployed closest to the processors— is not a replacement, but a complement to ACPI. For example, VMware ESX/ESXi is another product intended for hardware virtualized across different OSs. VMware ESX/ESXi already provides power management capabilities [35] and constitutes the system software layer where PAMPA should reside.

HP Power Regulator [14] is an OS-independent power management feature —equivalent to the EnergyScale microcontroller described in Section 2— to configure a server to maximize performance, maximize power savings, or match processor power consumption dynamically as system load changes. One mode, called *HP Dynamic Power Savings* mode, gives frequency control to the Power Regulator firmware on the system board. In this mode, the OS may also be making its own power management decision, decoupled from the Power Regulator control logic.

Regarding power management techniques, it is worth mentioning Ma *et al.*'s work [24], in which PCPG and DVFS are studied in the context of CMPs. Authors argue that PCPG and DVFS have to be actuated in a decoupled way to exploit the different characteristics of the two techniques. Their solution, known as *PGCapping*, operates PCPG and DVFS to meet CMP power constraints as well as to balance the core lifetimes. Even though we also use PCPG and DVFS in the context of CMPs, our work is novel due to the following reasons. First, we do not focus on power optimization solely but on exposing and tackling the drawbacks of the decoupled PCPG and DVFS operation. And second, once we understand these problems, we show that PCPG and DVFS

management has to be done in a coordinated way to improve the synergy of these techniques, which is in opposition to Ma *et al.*'s conclusion. In addition, our work is based on a real PCPG evaluation while Ma *et al.* make use of emulated PCPG. Cochran *et al.* [8] propose to *pack* software threads on to a variable number of cores to fit a given power budget in conjunction with DVFS. In our case, we use PCPG in addition to DVFS, showing an operative way to combine both. We consider that the use of DVFS alone has a limited effectiveness due to the power supply voltage (V_{dd}) becoming more and more close to the threshold voltage (V_{th}) in new generation systems. Lee *et al.* [21] present an analytical model to study the power and performance implications of the use of DVFS and PCPG in CMPs. Isci *et al.* [19] propose the adoption of a global power manager in the context of CMPs which senses per-core power and performance information at run-time. Based on variations in application behavior, the power manager sets particular per-core power levels to fit a power budget. Madan *et al.* [25] present a study of two basic PCPG heuristics and their potential flaws. The heuristics are aimed to reduce power consumption of idle cores. The paper also analyzes possible "holes" (which may produce negative power savings) and proposes a guard mechanism to prevent them. Also related to robustness of power management in multi-core processors, Bose *et al.* [5] provides a broad overview of the potential holes and introduce the idea of guarded power management.

The use of hardware event counters in the context of multi-threaded applications is also leveraged in prior studies. Tam *et al.* [32] propose a mechanism for thread clustering based on data sharing patterns. It is implemented at OS kernel level with information from hardware event counters. Bhattacharjee *et al.* [3] also propose the use of processor counters to dynamically predict *thread criticality*. A critical thread is the slowest thread in an application, which limits its performance. They propose to exploit thread criticality prediction for load balancing and energy saving purposes.

9. CONCLUSIONS

Dynamic power management capabilities are widely available as an integral part of today's processors and systems. These techniques include (but are not limited to) DVFS, core folding and PCPG. To make the most use of these power management knobs, system designers advocate the adoption of *decoupled* power management approaches. The goal is to keep the control algorithms simple and to take advantage of their different transition time overheads. However, the independent operation of the power management knobs can lead to conflicting decisions of the control algorithms which may result in worse power-performance efficiency.

In this work, we argue for the *coordination* of the power management activities in the system to improve the *robustness* of DVFS and PCPG operation. We present a detailed measurement-based evaluation and comparison of product-level power management policies available in a modern multi-core system. We then introduce PAMPA, a *Power-Aware Management of Processor Actuators* algorithm, capable of attaining power-performance efficiency comparable to the most aggressive power management approach, while achieving conflict-free actuation of the power management activities. In contrast to most of the utilization-based power management algorithms deployed in today's systems, PAMPA adopts a novel approach to guide its decisions — it dynamically "detects" situations in which an application may be bound by *single-thread-performance* or *throughput* and actuates the power management knobs accordingly. We evaluate PAMPA on a real AIX/POWER7+ system with DVFS, core folding and PCPG capabilities. In this scenario, PAMPA exhibits power-performance efficiencies for the SPECpower, PARSEC, Lulesh and CoMD benchmarks comparable to the most aggressive power management approach ("PCPG+DVFS"). At the same time, PAMPA avoids excessive performance degradation in cases where the independent DVFS and PCPG operation results in conflicting decisions.

Our work shows that the coordinated operation of power management knobs can benefit the synergy between these techniques and, at the same time, improve the robustness of the power management activities in the system.

Acknowledgments

This work is sponsored by Defense Advanced Research Projects Agency, Microsystems Technology Office (MTO), under contract no. HR0011-13-C-0022. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This document is: Approved for Public Release, Distribution Unlimited.

We thank Mr. David Palframan for his technical support during his internship at IBM.

10. REFERENCES

[1] SPECpower_ssj2008 web site:. `http://www.spec.org/power_ssj2008/`.

[2] ACPI. ACPI overview. `http://www.acpi.info/presentations/ACPI_Overview.pdf`, 2004.

[3] A. Bhattacharjee and M. Martonosi. Thread criticality predictors for dynamic performance, power, and resource management in chip multiprocessors. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 290–301, 2009.

[4] C. Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.

[5] P. Bose, A. Buyuktosunoglu, J. Darringer, M. Gupta, M. Healy, H. Jacobson, I. Nair, J. Rivers, J. Shin, A. Vega, and A. Weger. Power management of multi-core chips: Challenges and pitfalls. In *Proceedings of the Design, Automation Test in Europe Conference*, DATE '12, pages 977 –982, 2012.

[6] A. Branover, D. Foley, and M. Steinman. AMD Fusion APU: Llano. *IEEE Micro*, 32(2):28 –37, March-April 2012.

[7] M. Broyles, C. Francois, A. Geissler, M. Hollinger, T. Rosedahl, G. Silva, J. V. Heuklon, and B. Veale. IBM EnergyScale for POWER7 processor based systems. `ftp://public.dhe.ibm.com/common/ssi/ecm/en/pow03039usen/POW03039USEN.PDF`, 2013.

[8] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda. Pack & cap: adaptive DVFS and thread packing under power caps. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '11, pages 175–185, 2011.

[9] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini. Coscale: Coordinating CPU and memory system DVFS in server systems. In *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '12, pages 143–154, 2012.

[10] V. Devadas and H. Aydin. Coordinated power management of periodic real-time tasks on chip multiprocessors. In *Proceedings of the International Conference on Green Computing*, GREENCOMP '10, pages 61–72, 2010.

[11] Exascale Co-Design Center for Materials in Extreme Environments (ExMatEx). CoMD: A classical molecular dynamics mini-app. `http://exmatex.github.io/CoMD/doxygen-mpi/index.html`.

[12] M. Floyd, M. Allen-Ware, K. Rajamani, B. Brock, C. Lefurgy, A. Drake, L. Pesantez, T. Gloekler, J. Tierno, P. Bose, and A. Buyuktosunoglu. Introducing the adaptive energy management features of the POWER7 chip. *IEEE Micro*, 31(2):60–75, March-April 2011.

[13] M. Floyd, M. Ware, K. Rajamani, T. Gloekler, B. Brock, P. Bose, A. Buyuktosunoglu, J. C. Rubio, B. Schubert, B. Spruth, J. A. Tierno, and L. Pesantez. Adaptive energy-management features of the IBM POWER7 chip. *IBM Journal of Research and Development*, 55(3):8:1–8:18, May-June 2011.

[14] HP. HP Power Regulator. `http://h18013.www1.hp.com/products/servers/management/ilo/power-regulator.html`.

[15] W. Huang, C. Lefurgy, W. Kuk, A. Buyuktosunoglu, M. Floyd, K. Rajamani, M. Allen-Ware, and B. Brock. Accurate fine-grained processor power proxies. In *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '12, pages 224–234, 2012.

[16] IBM Corp. Virtual processor management within a partition. `http://pic.dhe.ibm.com/infocenter/aix/v6r1/topic/com.ibm.aix.prftungd/doc/prftungd/virtual_proc_mngmnt_part.htm`.

[17] Intel Corp. First the tick, now the tock: Next generation Intel microarchitecture (Nehalem). `http://www.intel.com/content/dam/doc/white-paper/intel-microarchitecture-white-paper.pdf`.

[18] Intel Corp. Intel® Xeon® processor E5 family. `http://www.intel.com/content/www/us/en/processors/xeon/xeon-processor-5000-sequence.html`.

[19] 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 *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '06, pages 347–358, 2006.

[20] I. Karlin, J. Keasler, and R. Neely. Lulesh 2.0 updates and changes. Technical Report LLNL-TR-641973, August 2013.

[21] J. Lee and N. S. Kim. Optimizing throughput of power- and thermal-constrained multicore processors using DVFS and per-core power-gating. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, pages 47–50, 2009.

[22] C. Lefurgy, X. Wang, and M. Ware. Server-level power control. In *Proceedings of the 4th International Conference on Autonomic Computing*, ICAC '07, pages 4–, 2007.

[23] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, and J. B. Carter. Active management of timing guardband to save energy in POWER7. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '11, pages 1–11, 2011.

[24] K. Ma and X. Wang. PGCapping: exploiting power gating for power capping and core lifetime balancing in CMPs. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, PACT '12, pages 13–22, 2012.

[25] N. Madan, A. Buyuktosunoglu, P. Bose, and M. Annavaram. A case for guarded power gating for multi-core processors. In *Proceedings of the 17th International Symposium on High-Performance Computer Architecture*, HPCA '11, pages 291–300, 2011.

[26] A. Mericas, B. Elkin, and V. R. Indukuru. Comprehensive PMU event reference - POWER7. `http://www.power.org/documentation/comprehensive-pmu-event-reference-power7/`.

[27] P. R. Panda, A. Shrivastava, B. Silpa, and K. Gummidipudi. *Power-Efficient System Design*. Springer, 1st edition, 2010.

[28] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No "power" struggles: coordinated multi-level power management for the data center. In *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIII, pages 48–59, 2008.

[29] K. Rajamani, F. Rawson, M. Ware, H. Hanson, J. Carter, T. Rosedahl, A. Geissler, G. Silva, and H. Hua. Power-performance management on an IBM POWER7 server. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '10, pages 201–206, 2010.

[30] E. Rotem. Power management architecture of the 2nd generation Intel Core™ microarchitecture, formerly codenamed Sandy Bridge. In *23rd Hot Chips Symposium*, August 2011.

[31] B. Steigerwald, C. Lucero, C. Akella, and A. Agrawal. *Energy Aware Computing: Powerful Approaches for Green System Design*. Intel Press, 1st edition, 2011.

[32] D. Tam, R. Azimi, and M. Stumm. Thread clustering: sharing-aware scheduling on SMP-CMP-SMT multiprocessors. In *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, pages 47–58, 2007.

[33] S. Taylor. POWER7+™: IBM's next generation POWER microprocessor. In *24th Hot Chips Symposium*, August 2012.

[34] A. Vassighi and M. Sachdev. *Thermal and Power Management of Integrated Circuits*. Springer, 1st edition, 2006.

[35] VMware. Host power management in VMware vSphere®5. `http://www.vmware.com/files/pdf/hpm-perf-vsphere5.pdf`, 2010.