An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget

Canturk Isci[†], Alper Buyuktosunoglu[†], Chen-Yong Cher[†], Pradip Bose[†] and Margaret Martonosi[*]

[†]IBM T.J. Watson Research Center
Yorktown Heights

[*]Department of Electrical Engineering
Princeton University

Abstract

Chip-level power and thermal implications will continue to rule as one of the primary design constraints and performance limiters. The gap between average and peak power actually widens with increased levels of core integration. As such, if per-core control of power levels (modes) is possible, a global power manager should be able to dynamically set the modes suitably. This would be done in tune with the workload characteristics, in order to always maintain a chip-level power that is below the specified budget. Furthermore, this should be possible without significant degradation of chip-level throughput performance. We analyze and validate this concept in detail in this paper. We assume a per-core DVFS (dynamic voltage and frequency scaling) knob to be available to such a conceptual global power manager. We evaluate several different policies for global multi-core power management. In this analysis, we consider various different objectives such as prioritization and optimized throughput. Overall, our results show that in the context of a workload comprised of SPEC benchmark threads, our best architected policies can come within 1% of the performance of an ideal oracle, while meeting a given chip-level power budget. Furthermore, we show that these global dynamic *management policies perform significantly better than* static *management, even if static scheduling is given oracular knowledge.*

1 Introduction

Recent microprocessor generations have seen a tremendous increase in transistor density, fueled by advances in semiconductor technology. At the same time, limits in instruction level parallelism (ILP) coupled with power dissipation constraints have caused the high performance microprocessor roadmap to enter the "multi-core" era. Initially, this era of chip multiprocessing (CMP) [35, 40] began with a step-back in core-level frequency (shallower processor pipelines) to allow multi-core (multi-thread) throughput increase at affordable power [15, 16, 18, 29, 41]. If per-thread frequency and performance growth were forever stymied, perhaps future scaling could yield more and more cores (threads) on a die, to get the throughput performance going for a few generations of a core's lifetime. However, the demand for single-thread performance growth is still alive for many crucial application domains; and even without that, growth in the number of cores causes super-linear growth in non-core area and power in order to meet scalable performance targets.

As such, the power dissipation problem did not disappear in the new multi-core regime. In fact, power and peak temperature continue to be the key performance limiters, even as other constraints like chip I/O bandwidth and intra/inter chip data bandwidth and available area for on-chip cache begin to emerge as new (secondary) limiters. Consequently, power and thermal management of multi-core (often organized as CMP) chips has been one of the primary design constraints and an active research area.

Prior research in this area has been primarily limited to studying the efficacy of providing dynamic management of power and thermal hot spots through localized responses provided at the core-level [2, 11] or through global scheduling heuristics [7, 22, 37]. However, the problem of enforcing a chip-level power budget (at minimal performance cost) through a global power manager has not been studied at great depth so far in the literature. The other, related problem of minimizing the power for a given multi-core performance target has similarly not been analyzed in detail. In this paper, we present such an analysis in the context of a generic CMP chip, assuming POWER4/5 [15] class cores.

There are three primary contributions in this work. First, we introduce the concept of a global power manager, that provides a hierarchical feedback-control based mechanism to sense the per-core power and performance "state" of the chip at periodic intervals; it then sets the operating power *level* or *mode* of each core to enforce adherence to known chip-level power budgets. Second, we develop a fast static power management analysis tool to evaluate the benefits and performance costs of various multi-core mode-management policies. Third, we evaluate several different policies for global multi-core (CMP) power management for different objectives, such as prioritization, fairness and optimized chip-level throughput. These simulation-based experiments are performed after factoring in all mode transition overhead costs. Our most important result shows that the best performance-optimizing management policy, *MaxBIPS*, comes within 1% of the performance of an ideal oracular controller, while still meeting the desired power budget. Overall, the range of policies and attendant benefit analyses presented, build key understanding about the fundamental trade-offs in power and performance that the architectural definition of such a global power manager would entail.

The rest of the paper is organized as follows. Section 2 gives a brief overview of our envisioned global management structure. Section 3 describes our experimental methodology and toolset. Section 4 explains our core power-mode defini-

[†]Canturk Isci is a PhD student at Princeton University. This work was done while he was an intern at IBM Research.

tions. Section 5 describes several different global management policies, along with an evaluation of their power-performance benefits. Section 6 provides a generalized summary and interpretation of the results. Section 7 provides a sketch of related (prior) work and Section 8 offers our concluding remarks.

2 Global CMP Power Management: Overview

The main motivation in architecting a global power manager is to exploit the widely known variability in demand and characteristics of the input workloads: both within a given thread, as well as across threads. The variations in application behavior are often repetitive because of loop-oriented execution semantics. The different regions of execution are commonly referred to as *phases* of application behavior [39]. Adaptive responses to such phases for power-efficient computing in a uniprocessor setting have been well studied in the past (e.g. [1]). In a multi-core setting, the global power manager must observe phase states and transitions across all the cores and take appropriate mode-setting actions with the objective of enforcing a chip-level power budget.

Figure 1 shows the conceptual structure of a hierarchical, global power manager. In this vision, we consider each core to have multiple operating modes that we call *power modes*. These modes can be set independently by the hierarchical controller, depending on workload behavior, overall budget constraints and mode-shift constraints. Within each core, we assume a built-in *local* open-loop management function that provides core-level power management actions, without interference from the global controller. For example, baseline dynamic power management techniques like clock-gating, fetch-gating, etc. may be used to provide a baseline power-efficient core functionality. The local monitors do, however, provide periodic per-core power-performance data to the global manager. The global management layer provides per-core mode-setting directives, in coordination with higher-level scheduling and load-balancing directives provided by system software, as indicated in Figure 1.

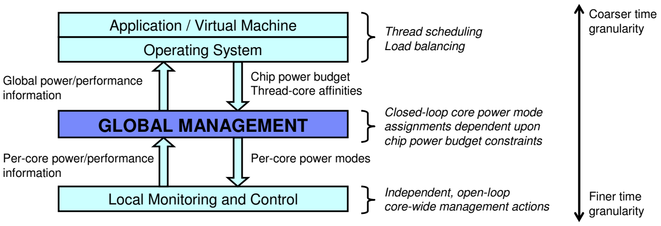

Application / Virtual Machine
Operating System
Thread scheduling
Load balancing
Coarser time granularity
Global power/performance information
Chip power budget
Thread-core affinities
GLOBAL MANAGEMENT
Closed-loop core power mode assignments dependent upon chip power budget constraints
Per-core power/performance information
Per-core power modes
Local Monitoring and Control
Independent, open-loop core-wide management actions
Finer time granularity

Figure 1. Global and local monitoring and control.

For per-core power information, we assume the presence of on-core current sensors, similar to those available in the Foxton controller of the Montecito chip [30, 36]. For performance-related information, we rely upon on-core performance monitoring counter hardware, which are readily available in most microprocessors. Local current sensors provide the core power consumption information, and the local performance counters provide the core performance information (i.e. retired instructions per sampling period) to the global controller. As indicated in Figure 1, such a hierarchical controller is able to provide fast-loop hardware responses to exploit fine-grain workload phase changes, as well as slower-loop software-managed responses, to coarser-grain phase changes. The work described in this paper is limited to a controller that has a single level of hierarchy: local, per-core (built-in) management, working in coordination with a single global manager. In this work, the global power manager periodically monitors the power and IPC of each core and sets the operating modes of the cores for the next monitored interval.

Core Configuration	
Dispatch Rate	5 instructions per cycle
Instruction Queue	256 entries
Reservation Stations	Mem: 2x18, FIX: 2x20, FP: 2x5
Functional Units	2LSU, 2 FXU, 2 FPU, 1BRU
Physical Registers	80 GPR, 72FPR
Branch Predictor	16K entry bimodal 16K entry gshare 16K entry selector
Memory Hierarchy	
L1 Dcache	32KB, 2 way, 128B blocks, 1 cycle latency
L1 Icache	64KB, 2 way, 128B blocks, 1 cycle latency
L2 Ucache	2MB, 4 way LRU, 128B blocks, 9 cycle latency
Memory	77 cycle latency

Table 1. Design parameters for processor model.

This global manager can be implemented as a separate on-die microcontroller with the underlying monitoring and response structure similar to the Foxton controller, it can be a separate helper daemon running on a dedicated core, or it can be a low level, hypervisor-like application interface running on dedicated address space. The choice of implementation depends on implementation feasibility and overhead-granularity trade-offs. As it is beyond the scope of this work, we do not make any explicit assumptions on the implementation choice. The remainder of this paper focuses on defining the functions of global, multi-core power management, as well as presenting the comparative benefits of such an approach.

3 Experimental Methodology

3.1 Simulation Framework and Microarchitectural Model

For our core-level power and performance explorations, we use a detailed cycle-accurate simulator based on IBM's Turandot simulator [33] as part of the Microarchitectural Exploration Toolset (MET). The power statistics are acquired from IBM's PowerTimer power modeling methodology, integrated with Turandot [4]. Baseline Turandot models a detailed, out-of-order, IBM POWER4 like processor architecture with more aggressive clock gating models. In our experiments, we model each core and shared L2 cache with the list of parameters in Table 1. The details of Turandot's POWER4 based architecture can be found in [12].

To evaluate global CMP power management techniques, we also developed a static, trace-based CMP analysis tool integrated with Turandot. This tool provides a fast, scalable interface to develop and evaluate different policies for global power management, as well as providing benchmark statistics for single threaded and static power mode definitions.

This simulation environment uses single threaded Turandot results for each evaluated power mode and performs CMP simulation by simultaneously progressing over Turandot traces for different benchmarks assigned to different cores. We discuss these power modes in detail in the next section. The mode switches are performed simultaneously at all cores, at execution points we refer to as *explore times* (500 μs in our evaluations). After an explore time, all mode switches are performed and we continue CMP simulation until the next explore time. During this period, the simulator simply evaluates the power/performance statistics based on the current mode settings. Our simulation tool updates its simulation statistics every *delta sim time* (50 μs in our evaluations). That is, at each delta sim time, it reevaluates the per-core and overall chip statistics. With this approach, we can explore a large number of cores from 2 to 64, with different power budgets, simulation termination conditions and policy descriptions.

Inevitably, our trace based tool is not cycle-accurate and cannot accurately incorporate shared L2 bus and address conflicts. Therefore, we also validate our power/performance characterizations with respect to a cycle-accurate full-CMP implementation of Turandot. This implementation is similar to previous work by Li et al. [25, 26], where we add time driven L2 and thread synchronization to manage multiple clock domain modes. We also include global dynamic management policies and support for per-core dynamic DVFS to assist individual power mode assignments to each core during execution.

Our experiments show that power variations with full-CMP simulations are relatively small (within 5% of single threaded powers) and the CMP power values are consistently lower. This guarantees that our trace-based assumptions persistently meet slightly lower actual power envelopes. Performance variations with multicore effects are comparatively more significant (on average 9% and up to 30% with highly memory bound applications) leading to lower IPC values due to increased shared cache and bus conflicts. However, more importantly, the variations for each benchmark under different CMP configurations are generally much smaller than the inter-benchmark performance differences. Thus, our derived global management policies still achieve close to optimal power mode choices for different workload combinations with our trace-based simulator. Overall, the policy behaviors for each workload combination as well as the differences across different combinations are consistent between the two approaches. These show that our trace-based evaluations effectively capture the power/performance trade-offs of global power management policies.

3.2 Benchmark Combinations

We analyze 12 workloads from SPEC CPU2000 suite in our experiments. The benchmarks are compiled with the XLC and XLF90 compilers. For two-way and four-way CMP examples, we consider various combinations with different CPU intensities. We group the combinations such that there is some benchmark variability among the groups. We show the selected group configurations and the corresponding CPU intensities in Table 2.

4 Core Power Modes

As we have described, we envision future CMP systems, with multiple power/performance modes for each core. In such a system, global power management optimizes chip power/performance by assigning power modes for each core, based on application behavior. Therefore, a first step in global management is the definition of these power modes.

	Benchmarks	Suites	Aggregate Effect
2-Way CMP	ammp|art	FP|FP	*Low CPU utilization, high memory utilization*
	gcc|mesa	INT|FP	*High CPU utilization, low memory utilization*
	crafty|facerec	INT|FP	*Very high CPU utilization, very low memory utilization*
	art|mcf	FP|INT	*Very low CPU utilization, very high memory utilization*
4-Way CMP	ammp|mcf|crafty|art	FP|INT|INT|FP	*Low CPU utilization, high memory utilization*
	facerec|gcc|mesa|vortex	FP|INT|FP|INT	*High CPU utilization, low memory utilization*
	sixtrack|gap|perlbmk|wupwise	FP|INT|INT|FP	*Very high CPU utilization, very low memory utilization*
	mcf|mcf|art|art	INT|INT|FP|FP	*Very low CPU utilization, very high memory utilization*

Table 2. Benchmark combinations for CMP experiments.

In our work, we explore various power mode choices for a final implementation decision. We evaluate the quality of different modes with a power savings per performance degradation criterion. Specifically, we define three power modes: *Turbo*, *Efficient1(Eff1)* and *Efficient2(Eff2)*. Our target in each power saving mode is to achieve a $\Delta PowerSavings$: $\Delta PerformanceDegradation$ ratio of 3 : 1. Under this target, Turbo represents full-throttle execution, Eff1 represents a medium power savings mode with minimal performance degradation. Eff2 represents high power saving with relatively significant performance degradation. We limit ourselves to only three distinct modes, because defining large number of modes has significant overhead and complexity impacts for the global manager. As we discuss in more detail in Section 5.5, the required state space for global control is linearly dependent on the number of modes. In addition, for a predictive or exploratory mode selection method, the number of required prediction or exploration steps has a superlinear dependence on the number of modes. In Table 3, we summarize the design targets for these modes.

Mode	Power Savings	Performance Degradation
Turbo	None	None
Eff1	15%	5%
Eff2	45%	15%
General Target	**3X**	**1X**

Table 3. Target $\Delta Power : \Delta Performance$ ratios for different power modes.

We consider DVFS as the underlying mechanism for defining the three global management modes, Turbo, Eff1 and Eff2. Although many of the prior mobile or embedded platforms support much more than three DVFS levels, this assumption matches well with the current CMP server platforms. For example two of Intel's recently released CMP server platforms, Sossaman and Woodcrest both support four global (V,f) levels.

In our CMP scenario, application of DVFS means being able to switch different cores to different voltage and frequency domains. This is a major design consideration. With such an implementation, the CMP system becomes a multiple clock and voltage domain processor. Nonetheless, these types of globally asynchronous locally synchronous (GALS) architectures are not uncommon in current research [14, 28, 43]. Considering granularities, DVFS overheads are on the order of microseconds. Therefore, applying DVFS is only feasible with periods of roughly 100K cycles—approximately 100μs for a 1GHz system.

In our exploration, we choose a *linear* DVFS scenario, where both voltage and frequency are scaled linearly. This choice falls within the scope of some of the previous product datasheets [13] for our small range of considered frequency reductions [10]. However, this should be considered as an optimistic bound since the supply voltages are shrinked more aggressively in emerging processor generations [38]. For linear DVFS, we define our modes as follows: **Turbo**: No DVFS (Vdd, f), **Eff1**: $(95\%\ Vdd,\ 95\%\ f)$, and **Eff2**: $(85\%\ Vdd,\ 85\%\ f)$.

A useful property of DVFS is that power and performance behavior can be approximately estimated with simple calculations. Power has a cubic relation to DVFS scaling [5] and performance has a somewhat linear dependency on frequency. Based on these trends, the upper bounds for power and performance for our mode choices are shown in Table 4. Note that the actual performance behavior is expected to be better, as asynchronous memory latencies are not scaled with DVFS.

Power Saving			Performance Degradation		
Turbo	Eff1	Eff2	Turbo	Eff1	Eff2
0	**~14%**	**~38%**	**0**	**<5.3%**	**<17.7%**

Table 4. Estimated power savings and performance degradation with DVFS.

These initial estimates show promising figures to achieve our target 3:1 power/performance trade-off. To validate these, we incorporate DVFS into Turandot. We consider the described Turbo, Eff1 and Eff2 modes for the chosen (V,f) domains and scale memory and L2 access cycle latencies for each mode. We quantify performance degradation with the elapsed execution time for benchmark execution. In Figure 2, we show two corner case benchmarks, sixtrack and mcf, and the overall behavior. To compute overall power/performance relations, we consider normalized execution times—to Turbo case—for each application, and average over the whole suite.

In Figure 2, the power dissipations follow closely with our estimates, while our initial performance degradation estimates prove to be the upper bound that is observed with a highly CPU bound benchmark such as sixtrack. The actual performance degradation ranges among applications depending on their memory boundedness, with mcf drawing the lower bounds in our experiments. Over our whole application pool, both Eff1 and Eff2 power management modes achieve approximately the 3:1 $\Delta PowerSavings : \Delta PerformanceDegradation$ ratio. Computations for power estimates at different modes track very closely with actual behavior. We make use of this fact later in our global power management policy designs to predict different mode behaviors proactively.

5 Global Power Management Policies

In this section we discuss different Global CMP power management policies under a common constraint. We devise policies that are subject to meeting a specific global power budget by adjusting power modes of individual cores. Besides this common constraint, different policies can target at different objectives such as prioritization of cores/benchmarks, balancing power among cores and optimizing system throughput. We describe our evaluation method and focus on optimizing throughput. We also discuss the position of our methods with respect to upper and lower bounds for power management options.

5.1 Evaluation Methodology

We have previously described our experimentation strategy in Section 3. In our analyses we use this framework with the following parameters that reflect the large time granularity of DVFS. We use *delta sim times* of 50 μs and *explore times* at 500 μs. The termination condition is after one of the benchmarks reaches completion. Thus, all cores are utilized for the experimented regions. We consider a nominal Vdd of $1.300\,V$ for Turbo mode. For our selected frequency scaling scenario, Eff1 and Eff2 modes operate at $1.235\,V$ and $1.105\,V$. Based on specifications from previous products and research, we choose a realistic DVFS transition rate of $10mV/\mu s$. Thus, the transition overheads for our three power modes are computed as shown in Table 5.

Transition	ΔV [mV]	Δt [μs]
Turbo → Eff1	65	6.5
Eff1 → Eff2	130	13
Turbo → Eff2	195	19.5

Table 5. DVFS transition overheads for the three power mode transitions.

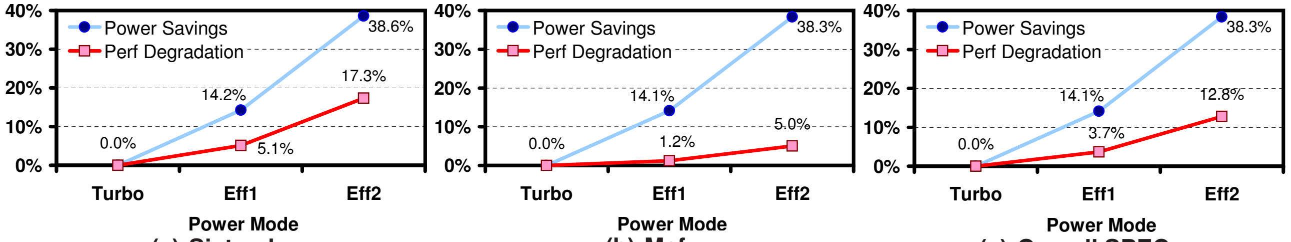
Table 5 shows the transition overheads for the three modes are on the order of 10 μs. Therefore, compared to our 500 μs explore times, these have relatively low overheads ranging from 1 to 4 %. For our evaluation of CMP power management, we choose the following assumptions. We assume there is no benchmark execution during mode transitions. On the other hand, we assume CPU power is still consumed. This is a conservative approach, exploring the worst case corner in terms of power/performance efficiency, as some implementations [3, 5] can continue execution during mode transitions or can reduce power dissipation during stalled execution [8]. Our multiple clock domain implementation imposes additional overheads for synchronization. At each explore time, if there is a mode transition, we find the longest transition cost among all cores and assume all cores are stalled during this period. Penalizing all cores with the largest penalty is not the most efficient approach, but is much more beneficial to keep cores synchronized for following explorations.

5.2 Policy Choices

We have experimented with different policies that can be invoked under different conditions. Here, we first introduce three of the experimented policies for different objectives.

5.2.1 Priority

Our first policy, *priority*, assigns different priorities to different tasks. In our implementation, for a four-core CMP, core4 has the highest priority and core1 has the lowest priority. Therefore, in policy implementation, it tries to run the fourth core



(a) Sixtrack (b) Mcf (c) Overall SPEC

Figure 2. $\Delta PowerSavings : \Delta PerformanceDegradation$ **for DVFS.**

as fast as possible, while preferring to slow down the first core first in case of a budget overshoot. In priority execution, as budget is increased, first core4 then cores 3 to 1 are released from Eff2 to Turbo, until the budget exceeding condition is met. In small budget changes, priority can operate out of order, as the core with the highest priority might not satisfy the budget in its next available mode. In such cases, the first core in priority order that satisfies the budget is moved to the next mode.

5.2.2 PullhiPushLo

The second policy, *pullHipushLo*, tries to balance the power consumption of each core, by slowing down the core that has the highest power in case of a budget overshoot, and by speeding up the lowest power core when there is available power slack. To have fair power distribution, pullHipushLo actually employs a sort of prioritization as well: benchmarks are preferred in their memory boundedness order. Moreover, this policy can also exhibit non-monotonic behavior with different budgets, as different amounts of slack can enable different mode choices that satisfy better power balancing.

5.2.3 MaxBIPS

Our last policy, *MaxBIPS*, targets at optimizing the system throughput, by predicting and choosing the power mode combination that maximizes the throughput at each explore time. MaxBIPS policy predicts the corresponding power and BIPS values for each possible mode combination. Afterwards, it chooses the combination with the highest throughput that satisfies the current power budget. The details of this predictive strategy are discussed in Section 5.5. The MaxBIPS policy exhibits a relatively regular behavior, but once again, this policy inherently assumes a priority. It actually prefers the benchmarks in the order of their CPU-boundedness. Therefore, it performs in a somewhat inverse way of pullHipushLo.

5.3 Chip-Wide DVFS

A simpler alternative to independent per-core power management is applying chip-wide DVFS. Chip-wide DVFS has very appealing features for implementation. As there is no synchronization across cores, it simplifies both OS and hardware implementation. In this case, all cores transition together into Turbo, Eff1 or Eff2 modes at each explore time based on budget constraints.

In Figure 3(a), we show the application of chip-wide DVFS for an 83% budget on a 4-core CMP running ammp, mcf, crafty and art. In Figure 3(c), we show the same experiment with ammp, crafty, art and sixtrack, replacing one memory bound benchmark (mcf) with a CPU bound one (sixtrack). In Figures 3(b) and (d), we show the operation of the same benchmark combinations under the MaxBIPS policy.

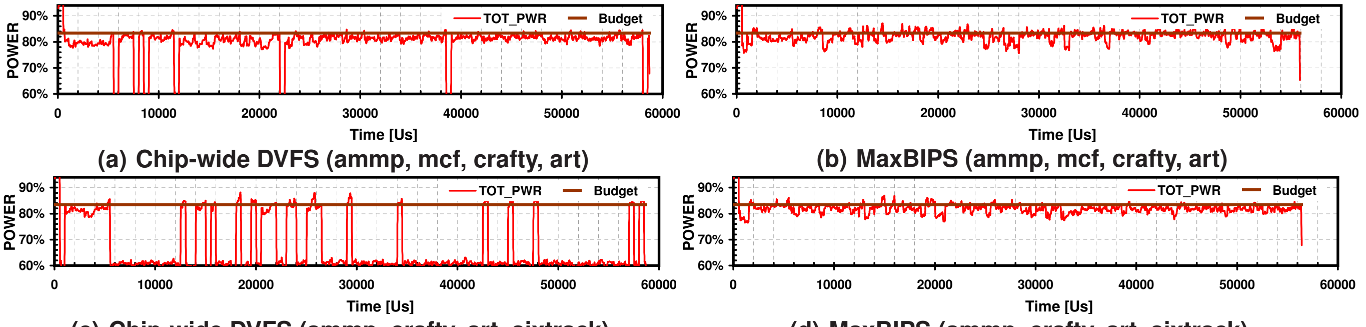
As seen in Figure 3, chip-wide DVFS has completely different implications for different set of benchmarks. In Figure 3(a), it fits the power envelope well, as the total chip power meets around the budget for the first set of benchmarks in Eff1 mode. In Figure 3(c), the same budget results in drastically reduced power/performance behavior due to one change in the benchmark set. In this case, as chip power is usually slightly higher than the target budget, all cores are penalized tremendously to run in Eff2. In comparison, the MaxBIPS policy fits the power budget efficiently regardless of the running benchmark combinations.

In Figure 3, the downside of chip-wide DVFS is clearly conveyed. With CMP systems, you can pay a huge penalty for small budget overshoots. Obviously this kind of monolithic global management has linearly growing negative impact with more aggressive scale-out scenarios. Also a comparison of the two chip-wide DVFS figures shows that global DVFS cannot guarantee generic good performance for a budget with varying workload combinations. This effect is also emphasized with scale-out due to more combinations of variations.

Note that, despite the unpromising results for the presented cases, chip-wide DVFS has certain advantages in terms of implementation simplicity. Especially for small scale-out cases such as a 2-way CMP chip, it might be possible to get comparable results with this approach, probably by introducing more power modes. One drawback of this argument is that the number of modes also need to scale with increasing number of cores on a chip.

5.4 Evaluation of Policies

A straightforward way of representing the effectiveness of a policy is in terms of *policy* and *budget curves*. In a *policy curve*, for each policy we draw the overall system performance degradation—with respect to all Turbo execution—for several target power budgets (shown as percentage of maximum chip power budget for a given configuration). Therefore, the policy which leads to least degradation for a given budget works better in that budget scenario. A *budget curve* plots the percentage of power consumed under a specific policy with respect to the original target budget to fit. This serves as a 'sanity check' to confirm the policies actually fit into the given power budget target. In Figure 4 we show the corresponding policy and budget curves for our three discussed policies and chip-wide DVFS, for the (ammp,mcf,crafty,art) workload combination. As Figure 4(a) shows, for all the given budgets, MaxBIPS performs significantly superior for performance as expected. For each power budget, it achieves the least over-



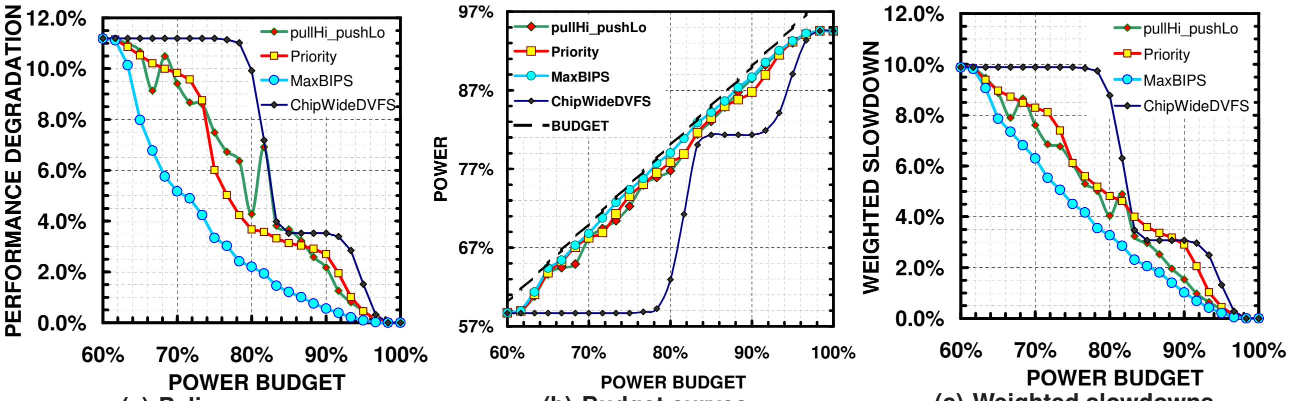
(a) Chip-wide DVFS (ammp, mcf, crafty, art)

(b) MaxBIPS (ammp, mcf, crafty, art)

(c) Chip-wide DVFS (ammp, crafty, art, sixtrack)

(d) MaxBIPS (ammp, crafty, art, sixtrack)

Figure 3. Chip-wide DVFS and MaxBIPS for a fixed target chip budget for two benchmark combinations.



(a) Policy curves

(b) Budget curves

(c) Weighted slowdowns

Figure 4. Policy and budget curves for the experimented policies.

all performance degradation in comparison to other policies. The policy curves clearly show the significantly larger performance degradation with chip-wide DVFS due to the impact of uniform global action. We show the target budget in the budget curves of Figure 4(b) with the dashed line. All policies successfully meet the target power budget for all experimented budgets. While our distributed, per-core management policies achieve chip power consumptions close to the available budget, chip-wide DVFS shows large power slacks under several budgets. This is because, moving all cores together to the next higher power/performance mode exceeds the budget. Therefore, all cores operate at a lower setting, not fully exploiting the available power cap. The two upward steps in the chip-wide DVFS budget curve essentially show the transition of all cores from Eff2 to Eff1, and from Eff1 to Turbo respectively.

Note that the raw performance degradations of Figure 4(a) do not account for any fairness considerations among different applications. Various studies have proposed unified quantitative metrics to incorporate “fairness” into performance gains such as *weighted speedup* [42] and *harmonic mean of thread speedups* [27]. To verify that the policy results are consistent under fairness considerations we show the *Weighted slowdowns* in Figure 4(c) for the three policies and chip-wide DVFS. We compute these from the harmonic mean of individual thread speedups—with respect to all turbo executions—at each power budget and subtracting this from 100% to report the experienced slowdowns. Similar results are obtained with the weighted speedup—using arithmetic mean—method with negligible differences. These weighted slowdown curves also show that MaxBIPS performs better than the other policies for all power budgets. However, the difference is less emphasized as MaxBIPS targets system throughput rather than individual task latencies. On the other hand, priority performs relatively worse than pullHipushLo under this condition as the latter tries to balance power dissipation among cores, thus applying some fairness criterion to mode selections.

Here we also revisit our previously discussed 3:1 $\Delta PowerSavings : \Delta PerformanceDegradation$ target ratio. In Figure 5, we show the achieved power savings and performance degradations by the described policies at each target budget. In each plot, we also show the target 3:1 ratio curve as the dashed straight line. Our per-core DVFS based power modes and experimented policies all show very good $\Delta PowerSavings : \Delta PerformanceDegradation$ ratios, matching the 3:1 ratio. In the case of MaxBIPS, we achieve significantly better than the 3:1 ratio with effective dynamic assignment of power modes.

Finally, we also present an example of how MaxBIPS policy dynamically adjusts core operation modes to manage chip power in Figure 6. In this example, we start with a power budget that is 90% of maximum chip power envelope, which later drops to 70% during execution. This represents a scenario where part of the cooling solution fails or the ambient environment changes during system operation. In Figure 6 (a), we show how each application contributes to the total chip power. MaxBIPS successfully manages the cores to maintain a total power consumption near power budget. In Figure 6 (b), we show the corresponding performance characteristics as percentages of the total chip BIPS for all turbo execution. At certain points, total BIPS exceeds 100% as overall instan-

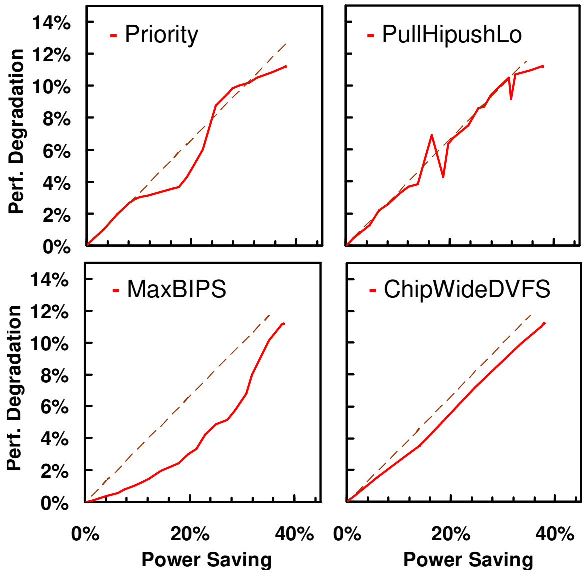


Figure 5. Power saving : performance degradation ratios for the three policies and chip-wide DVFS for the whole range of power budgets.

taneous chip performance for a lower power operating mode at some delta sim times can exceed the average BIPS at full power execution. The small performance degradation due to power budget drop is not immediately observable, but the average BIPS is reduced by 1% and 5% in the two power regions, which coincides with the policy curve values of Figure 4(a) for MaxBIPS.

At each explore time, MaxBIPS computes the best mode combination that meets the budget. However, there can be regions that exceed the budget at short periods due to unprecedented application behavior changes. These are corrected at the next explore time by detecting the budget overshoot. In the following sections, we consider optimizing system throughput for a highly utilized system with fixed power budget constraints and therefore focus mainly on MaxBIPS. We first compare MaxBIPS operation to achievable upper and lower bounds and later on present our generalized results.

5.5 Predicting Different Mode Behavior

In our policy discussions we have inherently assumed that we have the knowledge of power/performance behavior of applications in different modes. Based on this information, policies can choose among different mode combinations based on their merits. However, this knowledge is not directly available. The general approach in many dynamic management schemes is to perform small-scale "explorations" of each mode to deduce the application behavior in these modes, or to somehow predict this behavior from past history. For a heavy-handed adaptation like DVFS, this exploration approach is essentially prohibitive. Overheads lead to diminishing returns. The alternative approach is to assume that a previously seen behavior in a specific mode is the persistent behavior in that mode. However, this has unreliable outcomes, since relying on past history can be misleading with temporally changing application behavior.

As we have previously shown, DVFS has a useful property: an application's behavior at another DVFS setting can be estimated analytically with reasonable accuracy. Therefore, in all our analyses, we actually use this computational approach to estimate behavior in different modes.

We represent the characteristics of each mode in terms of *Power* and *BIPS Matrices*. For an N core CMP system with three power modes, two $Nx3$ matrices can completely characterize all mode behaviors. For the Power Matrix, different power behaviors of modes can be characterized by applying the cubic scaling relation. For the BIPS Matrix, BIPS values can be computed with a linear scaling.

For our original mode definitions, power and BIPS values of different modes can be estimated by scaling with 0.95 and 0.85. For example, if core1 is in Eff1 mode with power $P1_{E1}$ and BIPS $B1_{E1}$, corresponding Turbo and Eff2 power ($P1_T$ and $P1_{E2}$) and BIPS ($B1_T$ and $B1_{E2}$) values can be approximated as:

$$P1_T = P1_{E1}/0.95^3$$
$$P1_{E2} = P1_T \cdot 0.85^3$$
$$B1_T = B1_{E1}/0.95$$
$$B1_{E2} = B1_T \cdot 0.85$$

Note that above BIPS predictions do not account for transition overheads. However, at design time, the cost of switching among modes is well known as well as the BIPS and Power Matrix scalings. Therefore, same parallel BIPS matrix computations can be performed, only with additional scaling factors for BIPS values to incorporate transition costs. For our choice of parameters (with 500 μs explore times, and transition overheads of approximately 7 μs, 13 μs and 20 μs as depicted in Table 5), these scaling factors are 500/507, 500/513 and 500/520. For example, the BIPS after transitioning from Turbo mode to Eff2 mode can now be estimated as:

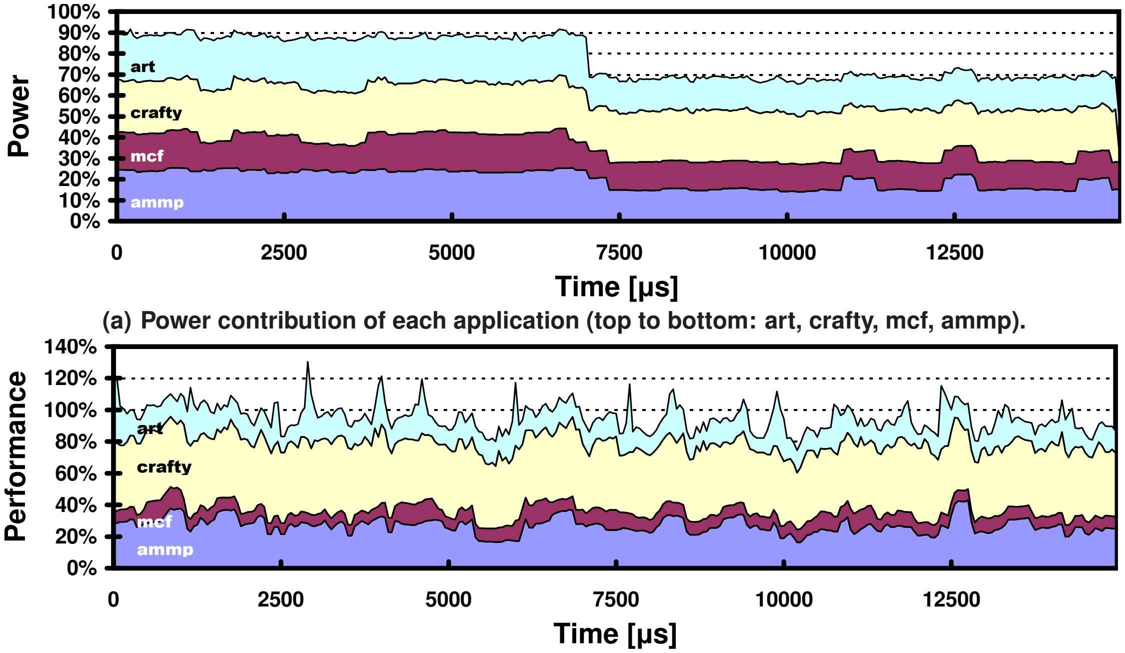
$$B1_{E2} = B1_T \cdot 0.85 \cdot (500/520)$$

As the power modes and number of cores are known in design time, all these relations can be hardwired. Thus, the matrix computation can be done in parallel by the global power management controller choosing which mode defines the properties for the other modes for each core.

Inevitably, these estimations are prone to errors. Over the SPEC suite, this is acceptable with 0.1-0.3% estimation errors for power. For BIPS values, the errors are 2-4%. The higher errors for BIPS are due to the differences in memory-boundedness of applications at different explore periods. For powers, the errors stem from slightly changing utilizations at delta sim cycles in different modes. As the more important estimate is for power—to assure we meet target power budgets, this approach works well in our DVFS scenario. In our experiments we use this approach to choose appropriate power modes for each core dynamically during execution.

5.6 Upper Bounds in Policy Efficiency with Oracle Mode Selection

To assess the efficiency of the policies from an optimal throughput perspective, here we describe the upper bounds for achievable efficiency with oracle knowledge. For the oracle based mode selection, at each explore period, we look at the future execution until the next explore time, i.e., 500 μs forward. We generate our oracle BIPS and Power matrices based



(a) Power contribution of each application (top to bottom: art, crafty, mcf, ammp).

(b) Performance contribution of each application (top to bottom: art, crafty, mcf, ammp).

Figure 6. Execution timeline of (ammp, mcf, crafty, art) with MaxBIPS policy, where power budget drops from 90% to 70% during execution.

on this future knowledge, and then choose the maximum-performance mode combination that satisfies the power budget.

In Figure 7(a), we show the policy curves for the oracle, chip-wide DVFS and MaxBIPS. Priority and pullHipushLo are omitted for brevity, but can be referenced from Figure 4(a). Although oracle performs slightly better than MaxBIPS at all budgets, it is easily seen that MaxBIPS lies within 1% of the oracle. This shows, our MaxBIPS policy, with predictive Power and BIPS matrices performs comparably to a strict oracle for dynamic CMP power management.

5.7 Lower Bounds with Static Mode Assignments

At the opposite corner of global power management lies static mode assignments for each core. That is, for each target budget, we choose a static mode configuration for each core and we do not further alter the core configurations during application execution. This represents a heterogeneous CMP configuration similar to Ghiasi [9]. In our analyses, we consider these separate static mode assignments for each core based on “oracle” knowledge for optimal core-mode-benchmark assignments. We call this *optimistic static management*. For this optimistic case, we look at the overall native executions of each benchmark at each power mode. Then, for each budget, we choose the mode combinations that maximize throughput, while satisfying budget requirements. For example, for one budget, we assume we have 3 Eff2 and 1 Turbo cores, while for another budget, we assume we have 3 Turbo and 1 Eff1 cores. Therefore, the results for static management at each power budget correspond to the highest achievable performance among all possibilities for that budget via static management.

In Figure 7(a), we also show the policy curves for this static case, *Static*. Even under such optimistic assumptions, the static case performs significantly worse than oracle and MaxBIPS, since static core configurations cannot respond to the temporal variations in workload behavior. This shows the potential impact of dynamic management. In addition to the policy curves, we again include the corresponding weighted slowdown curves in Figure 7(b) for the same set of policies, which demonstrate similar outcomes under fairness considerations.

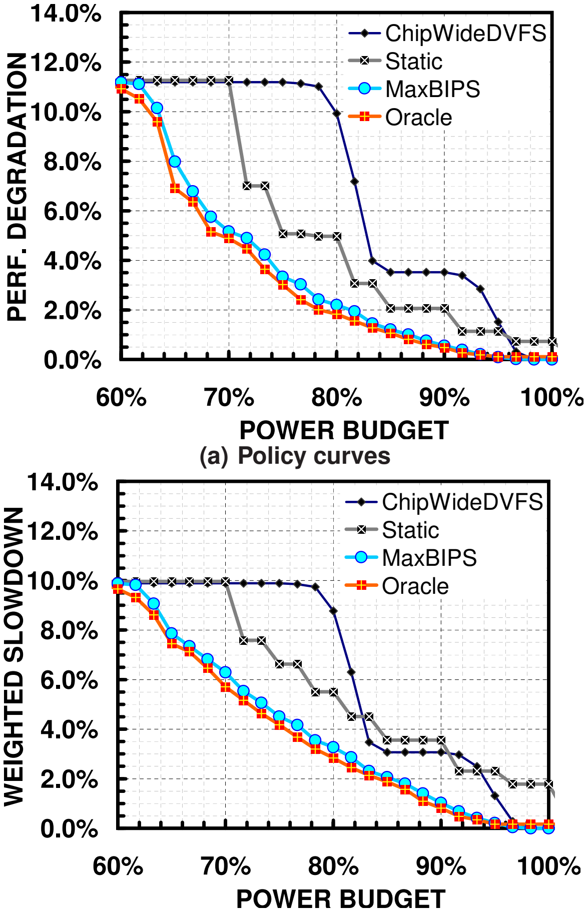
In a static assignment, rescheduling can be done at OS switching granularities. Without oracle knowledge, the OS can realize a bad core-benchmark assignment at the end of a context interval and can switch tasks at the expense of cache affinity. In comparison, a dynamic management approach like MaxBIPS is indifferent to benchmark-core pairings, and initial scheduling decisions have little impact on the management outcomes.

6 General Power Management Results

In the previous section, we have described different approaches to CMP power management and demonstrated the power/performance trade-offs for various policies. We discussed some of the practical implementation and overhead issues and evaluated these with respect to optimal and other alternative power management methods. In this section we generalize our results for a range of workload compositions as well as for different CMP configurations.

6.1 Overall Results

Here we present the outcomes of our CMP power management techniques under different chip power envelopes, for several benchmark combinations with different characteristics.



(b) Weighted slowdowns

Figure 7. Policy curves and weighted slowdowns for MaxBIPS, oracle, optimistic static mode selection and chip-wide DVFS.

Specifically, we look at the four benchmark sets described in Table 2, representing the different corners of workload combinations. We consider three CMP scales for 2, 4 and 8 core processors, all with shared 2MB L2 caches.

In Figures 8 and 9, we show the results for the previously described benchmark combinations, for two and four core CMP configurations respectively. We again show the attained system performance degradation for MaxBIPS policy, Static and ChipWideDVFS methods together with the oracle, in terms of *policy curves*. In Figure 10, we show two examples for an eight core CMP configuration by combining the two pairs of the four-core workload combinations.

In these plots, for each power management approach, we draw the overall system performance degradation—with respect to all Turbo execution— for several target power budgets. Therefore, the method which leads to the least degradation for a given budget can be considered superior in that budget scenario.

The results support our prior conclusions for the MaxBIPS policy, showing the effectiveness of CMP power management with adaptive global policies. In particular, MaxBIPS, which can tailor each core's power mode dynamically with respect to local workload variations under the global power budget constraints, achieves close to optimal power/performance trade offs. MaxBIPS performs significantly better than both static and chip-wide power management approaches under varying chip power budget constraints. For several chip power budgets, the latter two approaches experience 2X or higher performance degradations than the MaxBIPS policy. This difference in power/performance efficiency is emphasized with increased core counts.

Different benchmark combinations show the impact of workload variations on the performance of a CMP power management scheme. Consider high amount of variability across different benchmarks within a given combination, such as Figures 8(a), 9(a), 10(a) and 10(b). These offer better tradeoffs with dynamic management, with less overall performance degradations for a given power budget. On the other hand, workloads with small inter-workload variations experience an almost linear degradation in performance with reduced power budgets. Among these, memory bound combinations experience less slowdown as expected, since their performance is mainly determined by memory.

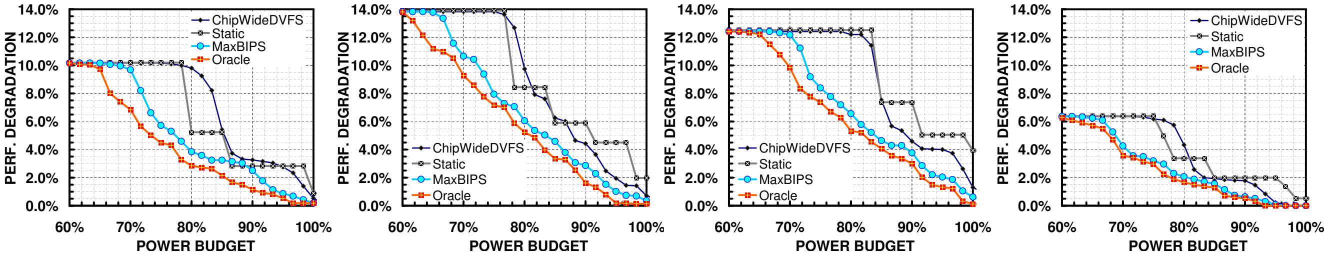
Comparing our MaxBIPS policy to the oracle power management shows the overall effectiveness of our approach. The MaxBIPS global dynamic management policy lies, on average, within 1% performance degradation of the oracle for all benchmark combinations. The performance difference between the oracle and MaxBIPS becomes trivial as we move towards higher CMP scales of 4 and 8 cores per chip. This shows that our described global CMP power management policy—with simple predictive power mode selection techniques—performs comparably to the optimal response for dynamic CMP power management. Effectively, it attains close to optimal power/performance trade-offs.

6.1.1 Trends under CMP Scaling

Comparing our policy results across different CMP scales shows important trends. As we scale out from 2 cores (Figure 8) to 8 cores (Figure 10), we observe three trends. First, the difference between MaxBIPS and the oracle decreases with increasing cores. In the eight core case, our MaxBIPS policy achieves practically the same policy curve as the oracle, while for two cores, the difference is significant. The reason for this is that for fewer cores, MaxBIPS tries to fit into the power envelope with fewer mode combination alternatives to select from. Therefore, at many points, the policy must conservatively underfit the power budget, thus sacrificing some of the attainable performance.

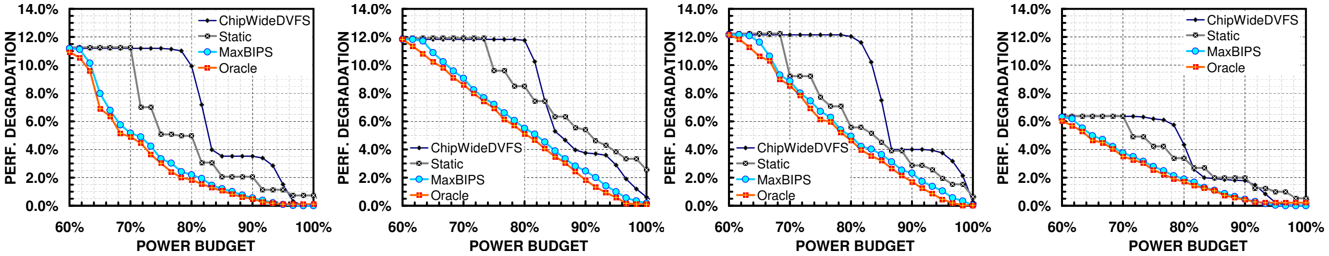
Second, the relative performance degradation achieved by static and chip-wide methods with respect to MaxBIPS increases with the number of cores. As previously mentioned, this is due to the increased inefficiency of chip-wide or static techniques to capture widening inter- and intra-workload variations. Both these trends mean that as the future implementations move to more aggressively scaled CMPs, the benefits and the necessity of our adaptive global power management policies will be similarly emphasized.

Third and last, the trade-offs between static and chip-wide techniques follow different trends under CMP scaling. While, for two core systems (Figure 8) both approaches are comparable, further CMP scales (Figures 9 and 10) favor static per-core power management over chip-wide dynamic manage-



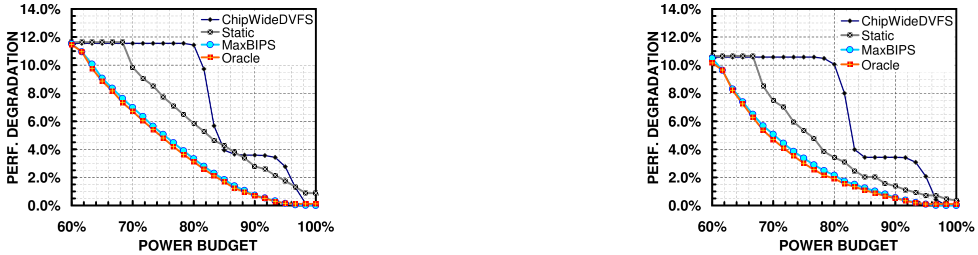
(a) (ammp, art) (b) (gcc, mesa) (c) (crafty, facerec) (d) (art, mcf)

Figure 8. 2-way CMP power management results for different benchmark combinations.



(a) (ammp, mcf, cra., art) (b) (fac., gcc, mesa, vor.) (c) (six., gap, per., wup.) (d) (mcf, mcf, art, art)

Figure 9. 4-way CMP power management results for different benchmark combinations.



(a) (ammp, mcf, cra., art, fac., gcc, mesa, vor.) (b) (six., gap, per., wup., mcf, mcf, art, art)

Figure 10. 8-way CMP power management results for different benchmark combinations.

ment. The static approach addresses inter-workload (spatial) variation, which provides more opportunities over higher CMP scales. Dynamic DVFS addresses intra-workload (temporal) variation over the whole chip. However, it cannot efficiently exploit available power budget slack with the few available power modes, as we have also previously discussed in Section 5.3.

We summarize our observations in Figure 11. In this figure, we show the difference between the achieved performance degradations of the three policies. The degradations are averaged over the active range of power budgets and over all experimented benchmarks. In Figure 11, we also show the results for a single core processor for reference. In the single-core case, MaxBIPS policy becomes identical to chip-wide DVFS. The performance degradation of MaxBIPS converges to oracle as we increase the number of cores. Performance degradation with static power management also decreases with increasing cores. However, it saturates around 2% higher than the oracle and MaxBIPS. Conversely, performance degradation with chip-wide management increases monotonically with increasing cores. The benefits of static and chip-wide approaches are comparable in the 2-4 cores range, but static approach becomes dominant for further scales. Finally, the relative performance efficiency of MaxBIPS is increasingly emphasized over the other two approaches with higher scales, showing the necessity of such per-core adaptive policies for global power management in the next generation CMP configurations.

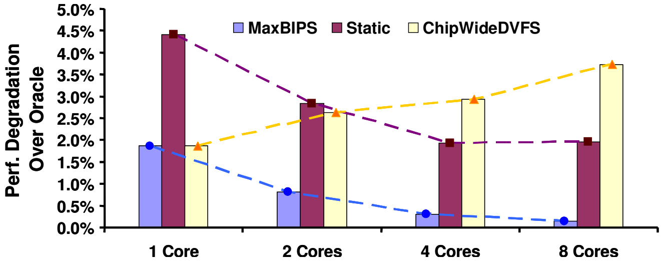


Figure 11. Policy trends with respect to CMP scaling.

7 Related Work

There are many prior studies that focus on the design and control of CMP systems under power, thermal or area constraints [6, 7, 20, 25, 37]. Most of these studies also suggest ways of adapting to workload behavior, under these different constraints. However, most of these management techniques guide independent local actions or OS level scheduling decisions. In contrast, our work focuses on policies that perform power/performance management by adjusting the behavior of individual cores under global chip-wide power budgets with the supervision of global monitoring and control.

One line of prior research considers CMP designs with different core configurations to more efficiently accommodate different application requirements. Kumar et al. [19, 21] propose heterogeneous CMPs in terms of different core complexities. They show that such an approach improves power consumption and can provide better performance-area trade-offs. On the other hand, Ghiasi [9] explores heterogeneity with cores operating at different frequencies. They show that such heterogeneous systems offer improved management of thermal emergencies. These works focus on heterogeneous CMP designs and leverage OS support to assign applications to better suited cores, while our work tries to adjust configurable cores with finer-grained policies to meet chip-wide global budgets.

Oliver et al. [34] describe a multiple clock domain, tile-based architecture that assigns different tile columns to different voltage and frequency domains. This work targets parallelized signal processing applications that are decomposed for execution on different tile columns. Each column can be run at different speeds to meet preset target rates for the applications with the final goal of reducing power. Juang et al. [14] also look at CMPs with dynamically configured voltage and frequencies. However, their work adjusts individual core executions to improve power-performance trade-offs by balancing inter-thread producer-consumer rates. Kotla et al. [17] consider a CMP system with multiple effective frequency domains by throttling cores at different rates. They demonstrate that memory intensive applications can be run on a slower core without significant performance loss. This work also employs some predictive strategies to estimate program behavior at different power modes to help guide an energy-aware scheduler. In comparison to these, our work develops policies that can adapt each core's behavior under fixed power budgets.

Another line of prior work also considers dynamic management under power budget constraints. Merkel et al. [31, 32] present system level methods that adjust processor execution to fit target budgets. They develop an energy-aware scheduling policy to meet uniform budgets for each core to eliminate thermal emergencies. Grochowski et al. [10] discuss latency and throughput trade-offs under chip power constraints. They survey different adaptation techniques and suggest asymmetric cores with DVFS as the most promising alternative. In a related study, Annavaram et al. [2] consider a real implementation example with an asymmetric multiprocessor (AMP). They consider both static and dynamic AMP configurations and improve performance of multithreaded applications under fixed power budgets. These works discuss adaptations in response to the available thread parallelism in applications, while our work investigates core power adaptation policies for fully-utilized CMPs based on predictive power and performance models at different power modes. Li and Martinez [23, 24] investigate methods to identify the optimal operating point on a CMP in terms of number of active cores and DVFS settings for parallel applications. They consider dynamic configurations under limited performance and power constraints and develop analytical models for attainable speedups. They consider application of chip-wide DVFS to manage parallel regions of applications and perform few explorations guided by heuristics to reach an optimal operating point. In contrast, our work focuses on per-core DVFS actions guided by policies that predict different power mode behaviors.

8 Conclusion

This paper proposes and evaluates a new perspective to dynamic power management for CMP systems. This work focuses on the chip-level global monitoring, control and dynamic management of power for the emerging multi-core systems. In our approach, global control is aware of the activity of all the cores in a system. Therefore, it can decide upon good per-core actions to meet global chip constraints.

In our experiments, we employ a fast and scalable static trace-based CMP power/performance analysis methodology to investigate global power management policies. Our analyses show that global management provides a good balance between power/performance under varying workload behavior. Such dynamic global power management can avoid the inefficiency of worst-case designs by reducing the power envelopes or by increasing on-chip core integration, with reasonable single-threaded performance sacrifice. Our policy experiments show the different trade-offs between fairness and throughput. In the experimented policies, the predictability of DVFS-based techniques alleviates any overheads related to the mode explorations. Our mode prediction methodology via the *Power* and *BIPS Matrices* can accurately estimate the application behavior across different operating modes. Our best performing policy, *MaxBIPS*, achieves performance degradations within 1% of a conservative oracle. The discussed dynamic management policies perform significantly better than static power management or chip-wide DVFS, even if these are provided with the future knowledge of the workload behavior.

Our results show the potentials and the applicability of CMP dynamic power management with global monitoring and control, under chip-level power budget constraints. Although there are other aspects and hierarchies of CMP power management that need to be explored, the presented results show the promising opportunities with global dynamic management. We believe as more aggressive scale-out strategies are followed, the benefits of global power management will correspondingly increase, encouraging the incorporation of such techniques in next generation systems.

References

[1] D. Albonesi, R. Balasubramonian, S. Dropsho, S. Dwarkadas, E. Friedman, M. Huang, V. Kursun, G. Magklis, M. Scott, G. Semeraro, P. Bose, A. Buyuktosunoglu, P. Cook, and S. Schuster. Dynamically Tuning Processor Resources with Adaptive Processing. *IEEE Computer*, 36(12):43–51, 2003.

[2] M. Annavaram, E. Grochowski, and J. Shen. Mitigating Amdahl's Law Through EPI Throttling. In *Proceedings of the 32nd International Symposium on Computer Architecture (ISCA-32)*, 2005.

[3] B. Brock and K. Rajamani. Dynamic Power Management for Embedded Systems. In *Proceedings of the IEEE International SOC Conference*, 2003.

[4] D. Brooks, P. Bose, V. Srinivasan, M. K. Gschwind, P. G. Emma, and M. G. Rosenfield. New Methodology for Early-Stage, Microarchitecture-Level Power-Performance Analysis of Microprocessors. *IBM J. of Research and Development*, 46(5/6):653–670, 2003.

[5] L. Clark, E. Hoffman, J. Miller, M. Biyani, Y. Liao, S. Strazdus, M. Morrow, K. Velarde, and M. Yarch. An embedded 32-bit Microprocessor Core for Low-Power and High-Performance Applications. *IEEE Journal of Solid-States Circuits*, 36(11):1599–1608, 2001.

[6] J. D. Davis, J. Laudon, and K. Olukotun. Maximizing CMP Throughput with Mediocre Cores. In *14th International Conference on Parallel Architecture and Compilation Techniques (PACT'05)*, 2005.

[7] J. Donald and M. Martonosi. Techniques for Multicore Thermal Management: Classification and New Exploration. In *Proceedings of the 33th International Symposium on Computer Architecture (ISCA-33)*, 2006.

[8] M. Fleischmann. LongRun Power Management. Whitepaper, Transmeta Corp., 2001.

[9] S. Ghiasi. *Aide de Camp - Asymmetric Multi-Core Design for Dynamic Thermal Management*. PhD thesis, 2004. Dept. of Computer Science, University of Colorado, Boulder, Ph.D. Thesis.

[10] E. Grochowski, R. Ronen, J. Shen, and H. Wang. Best of Both Latency and Throughput. In *Proceedings of the International Conference on Computer Design (ICCD)*, 2004.

[11] S. Heo, K. Barr, and K. Asanovic. Reducing Power Density through Activity Migration. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED), Seoul, Korea*, Aug. 2003.

[12] Z. Hu, D. Brooks, V. Zyuban, and P. Bose. Microarchitecture-level power-performance simulators: Modeling, validation and impact on design. Tutorial. In 36th International Symp. on Microarchitecture, Dec. 2003.

[13] Intel Corporation. *Intel 80200 Processor based on Intel XScale Microarchitecture Datasheet*, Jan. 2003.

[14] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, and D. Clark. Coordinated, Distributed, Formal Energy Management of Chip Multiprocessors. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED'05)*, Aug. 2005.

[15] R. Kalla, B. Sinharoy, and J. Tendler. IBM POWER5 Chip: A Dual-Core Multithreaded Processor. *IEEE Micro*, 24(2):40–47, Mar/Apr 2004.

[16] P. Kongetira. A 32-way Multithreaded SPARC(R) Processor. *Hot Chips 16*, Aug 2004.

[17] R. Kotla, A. Devgan, S. Ghiasi, T. Keller, and F. Rawson. Characterizing the Impact of Different Memory-Intensity Levels. In *IEEE 7th Annual Workshop on Workload Characterization (WWC-7)*, Oct. 2004.

[18] K. Krewell. UltraSPARC IV Mirrors Predecessor: Sun Builds Dual-Core Chip in 130nm. *Microprocessor Report*, Nov 2003.

[19] R. Kumar, K. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction. In *Proceedings of the 36th International Symp. on Microarchitecture*, Dec. 2003.

[20] R. Kumar, N. P. Jouppi, and D. M. Tullsen. Conjoined-Core Chip Multiprocessing. In *Proceedings of the 37th International Symposium on Microarchitecture (MICRO-37)*, 2004.

[21] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance. In *Proceedings of the 31st International Symposium on Computer Architecture*, June 2004.

[22] E. Kursun, C. Y. Cher, A. Buyuktosunoglu, and P. Bose. Investigating the Effects of Task Scheduling on Thermal Behavior. In *Third Workshop on Temperature-Aware Computer Systems (TACS'06)*, June 2006.

[23] J. Li and J. Martinez. Power-Performance Implications of Thread-Level Parallelism on Chip Multiprocessors. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'05)*, 2005.

[24] J. Li and J. Martinez. Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors. In *Proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA-12)*, 2006.

[25] Y. Li, D. Brooks, Z. Hu, and K. Skadron. Performance, Energy and Temperature Considerations for SMT and CMP Architectures. In *11th International Symposium on High Performance Computer Architecture (HPCA-11)*, 2005.

[26] Y. Li, B. Lee, D. Brooks, Z. Hu, and K. Skadron. CMP Design Space Exploration Subject to Physical Constraints. In *12th International Symposium on High Performance Computer Architecture (HPCA-12)*, 2006.

[27] K. Luo, J. Gummaraju, and M. Franklin. Balancing Throughput and Fairness in SMT Processors. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'01)*, Nov. 2001.

[28] G. Magklis, M. Scott, G. Semeraro, D. Albonesi, and S. Dropsho. Profile-based Dynamic Voltage and Frequency Scaling for a Multiple Clock Domain Microprocessor. In *Proceedings of the 30th International Symposium on Computer Architecture (ISCA-30)*, 2003.

[29] C. McNairy and R. Bhatia. Montecito - The Next Product in the Itanium(R) Processor Family. *Hot Chips 16*, Aug 2004.

[30] C. McNairy and R. Bhatia. Montecito: A Dual-Core, Dual-Thread Itanium Processor. *IEEE Micro*, 25(2):10–20, Mar/Apr 2005.

[31] A. Merkel. *Balancing Power Consumption in Multiprocessor Systems*. PhD thesis, Sept. 2005. System Architecture Group, University of Karlsruhe, Diploma Thesis.

[32] A. Merkel, F. Bellosa, and A. Weissel. Event-Driven Thermal Management in SMP Systems. In *Second Workshop on Temperature-Aware Computer Systems (TACS'05)*, June 2005.

[33] M. Moudgill, J.-D. Wellman, and J. H. Moreno. Environment for PowerPC Microarchitecture Exploration. *IEEE Micro*, 19(3):15–25, May/Jun 1999.

[34] J. Oliver et al. Synchroscalar: A Multiple Clock Domain Power-Aware Tile-Based Embedded Processor. In *Proceedings of the 31st International Symposium on Computer Architecture (ISCA-31)*, 2004.

[35] K. Olukotun, B. A. Nayfeh, L. Hammond, K. Wilson, and K.-Y. Chang. The Case for a Single-Chip Multiprocessor. In *Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS VII)*, Oct. 1996.

[36] C. Poirier, R. McGowen, C. Bostak, and S. Naffziger. Power and Temperature Control on a 90nm Itanium-Family Processor. In *IEEE International Solid-State Circuits Conference (ISSCC 2005)*, Feb. 2005.

[37] M. Powell, M. Gomaa, and T. N. Vijaykumar. Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system. In *Eleventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XI*, 2004.

[38] G. Semeraro, G. Magklis, R. Balasubramonian, D. Albonesi, S. Dwarkadas, and M. Scott. Energy-Efficient Processor Design Using Multiple Clock Domains with Dynamic Voltage and Frequency Scaling. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture (HPCA-8)*, 2002.

[39] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically Characterizing Large Scale Program Behavior. In Tenth International Conference on Architectural Support for Programming Languages and Operating Systems, Oct 2002.

[40] L. Spracklen and S. G. Abraham. Chip Multithreading: Opportunities and Challenges. In *11th International Symposium on High Performance Computer Architecture (HPCA-11)*, 2005.

[41] J. M. Tendler, S. Dodson, S. Fields, H. Le, and B. Sinharoy. POWER4 System Microarchitecture. *IBM Journal of Research and Development*, 46(1):5–26, 2002.

[42] D. Tullsen and J. Brown. Handling Long-latency Loads in a Simultaneous Multithreading Processor. In *Proceedings of the 34th Annual International Symposium on Microarchitecture (MICRO-34)*, Dec. 2001.

[43] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark. Formal Online Methods for Voltage/Frequency Control in Multiple Clock Domain Microprocessors. In *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XI)*, 2004.