Understanding the Thermal Implications of Multicore Architectures

Pedro Chaparro, José González, *Member*, *IEEE Computer Society*,
Grigorios Magklis, *Member*, *IEEE Computer Society*, Qiong Cai, and Antonio González, *Member*, *IEEE*

Abstract—Multicore architectures are becoming the main design paradigm for current and future processors. The main reason is that multicore designs provide an effective way of overcoming instruction-level parallelism (ILP) limitations by exploiting thread-level parallelism (TLP). In addition, it is a power and complexity-effective way of taking advantage of the huge number of transistors that can be integrated on a chip. On the other hand, today's higher than ever power densities have made temperature one of the main limitations of microprocessor evolution. Thermal management in multicore architectures is a fairly new area. Some works have addressed dynamic thermal management in bi/quad-core architectures. This work provides insight and explores different alternatives for thermal management in multicore architectures with 16 cores. Schemes employing both energy reduction and activity migration are explored and improvements for thread migration schemes are proposed.

Index Terms—Multicore architectures, dynamic thermal management, activity migration, dynamic voltage, frequency scaling.

1 Introduction

Dynamic thermal management (*DTM*) has been a hot topic in recent years. The purpose of DTM is to achieve high-performance computing while maintaining the chip below a safe temperature [3]. DTM is critical because of reliability reasons [34], leakage power [2], [10] (that depends exponentially on temperature), and cooling costs [2], [14], [26]. Current microprocessors are configured to slow down or, even, enter sleep mode when an emergency temperature is reached, which hurts performance.

Microarchitectural temperature control techniques are primarily classified into two categories. In the first category, temperature is controlled by reducing the amount of energy dissipated (that is, converted to heat). Usually, this implies some performance degradation. Dynamic voltage and frequency scaling (*DVFS*) has proven to be a highly effective DTM method because it achieves quadratic energy reduction while only degrading performance linearly. In the research community, DVFS schemes are usually implemented using proportional, integral, derivative (PID) controllers [8], [11], [32].

In the second category, temperature is controlled by distributing the processor activity over the chip area. High activity for a particular chip area means high localized heat dissipation. By constantly relocating the heat source (activity), total temperature can be reduced because average temperature for each particular part of the chip is reduced [16]. Many works use this second approach: migration at granularity of functional unit [16], [32], pipeline [24], cache bank [7], execution clusters [5], and, more recently, thread migration (TM, also known as core hopping) [11], [27].

The advent of multicore designs [1], [15], [18], [19] opens a new scenario for DTM. Recent works have studied DTM in designs of at most four cores [11], [27]. Designs with more cores present new opportunities for thermal management. As far as we know, no studies have addressed the issue of controlling temperature for systems with a higher number of cores.

This work presents an in-depth study of different DTM techniques that can be used in multicore designs. We also perform sensitivity analysis for most of the parameters of the different techniques, and we propose some improvements.

This paper is structured as follows: Section 2 provides the reader with insight on the different issues that arise when considering multicore designs. Section 2 also details what previous works have proposed in this area and what the novelty of our work is. Section 3 explains our simulation infrastructure, whereas Section 4 evaluates the techniques and analyzes the results. Finally, Section 5 concludes the paper.

2 Background and Related Work

In this section, we present several previous studies in the area of DTM, and we provide an in-depth analysis of the behavior of previously proposed mechanisms.

Powell et al. [27] proposed a TM and thread assignment mechanism for a quad-core design with simultaneous multithreading (SMT) capabilities. Donald and Martonosi [11] analyzed the spectrum of TM, DVFS, and possible combinations of the two, similarly to this work. However, there are important differences between their work and ours as will be explained later. Constantinou et al. [9] analyzed the implications of migrating threads among cores and concluded that not invalidating the branch predictor reduces the penalty of resuming execution with cold structures. Li et al. [22], [23] studied the design space of multicore systems. Their conclusion was that the design decisions are very different if thermal constraints are

- *The authors are with the Intel Barcelona Research Center, Intel Labs, UPC, Ed. Nexus II Jordi Girona, 29-3ª, 08034 Barcelona, Spain. E-mail: {pedro.chaparro.monferrer, pepe.gonzalez, grigorios.magklis, qiongx.cai, antonio.gonzalez}@intel.com.*

Manuscript received 1 Sept. 2006; revised 31 Jan. 2007; accepted 23 Feb. 2007; published online 11 Apr. 2007.
Recommended for acceptance by R. Iyer and D.M. Tullsen.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-0259-0906.
Digital Object Identifier no. 10.1109/TPDS.2007.1092.

Published by the IEEE Computer Society

introduced in the first stages of the design rather than at a later stage to limit a thermally blind design. In particular, they also predict that simple energy-efficient cores are the preferred choice for designs with large number of cores. Finally, Li and Martínez employ 16 cores in their study of energy efficiency [21].

2.1 Stop and Go Policies

Stop and go [3] policies are the simplest policies that can be used to reduce peak temperature. The underlying mechanism is as simple as "stopping" a core whenever its temperature reaches a threshold. As a safeguard, the threshold is set below the critical temperature. As soon as the core returns to a safe temperature, execution is resumed. *Stop&go* has been shown to be the least effective thermal management technique [11]. This work uses *stop&go* as a backup mechanism for critical situations, where other DTM mechanisms fail.

2.1.1 Core Stopping Mechanisms

Stopping a core can be achieved in different ways. The simplest is to clock gate the core (Scheme A). This stops dynamic energy dissipation, but since V_{dd} is not changed, the core continues dissipating leakage energy. The main advantage is that both the architectural and the microarchitectural states are preserved.

A second and more aggressive option is to save the architectural state and to cut the voltage supply of the core (Scheme B). In this scheme, a core dissipates no power at all and achieves faster cool downtimes, but its state is lost (that is why we need to save the state before powering off). According to previous studies [20], powering up a core that is completely shut down requires around 1,000 cycles. If the DTM control interval is long enough, the penalty is negligible. However, powering on/off too often or too fast may cause electrical problems because of current swings generated by the core.

There is an intermediate solution in which the context is kept in the core (Scheme C). This could be achieved if the core, while idle, is supplied by a "sleep" voltage, much lower than the nominal voltage, which would allow to preserve the state inside the core [12]. This way, the core dissipates much less energy and requires no context saving/restoring.

This work will assume as a base stop mechanism Scheme B—V_{dd} cut—because it is the most effective. We think it is realistic to assume a *fast context save* and restore mechanism with minimal impact.

2.1.2 Cooldown Interval Length

The combination of the thermal emergency temperature and the length of the idle interval define the performance of stop-based schemes. The reason is the charge/discharge exponential behavior of temperature (Fig. 1). The cooldown curve (dashed line) decreases exponentially in time. This means that, the more we wait, the less we benefit for being idle. The ideal interval would be the smallest one that the system could support (in our case, the one between two consecutive thermal sensor reads).

In previous studies [11], the cooldown interval tends to be relatively large. However, as we showed, long idle intervals penalize the performance of *stop&go* schemes excessively. For the rest of this work, we will assume that execution can be stopped and resumed in the granularity of a single thermal sensor interval (167μs as explained later) since this is the most optimistic assumption for *stop&go*.

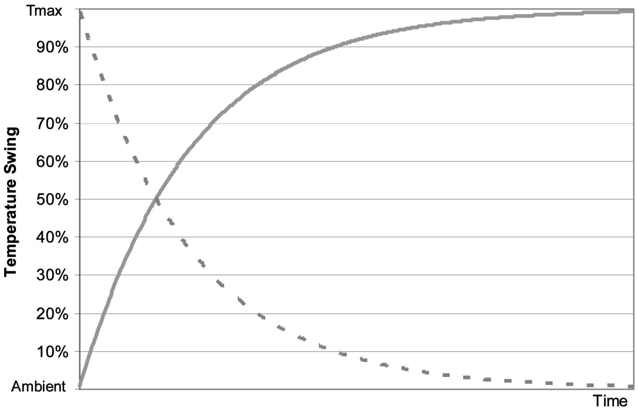


Fig. 1. Cooldown (dashed line) and heat-up (solid line) curves for a single RC of 1 ms (95 percent of the curve is achieved in 3×RC).

2.1.3 Thermal Emergency Temperature Impact on Stop&Go

Another interesting parameter is the ratio of cooldown and heat-up times. Assume that a core, which has been consuming power P, is stopped when it reaches temperature T. Then, it remains stopped for an interval of time dt, and its temperature is decreased by Td degrees. Now, let us assume the opposite case: A core (that is, exceeding temperature T) is stopped until it reaches temperature T. Then, its execution is resumed (consuming the same power P). After dt cycles, its temperature will increase Ti degrees.

It can be demonstrated that if T is located in the upper half of the curves in Fig. 1, then $|Td| > |Ti|$ and if T is located in the lower half, then $|Td| < |Ti|$ (in the middle point $Td = -Ti$).

In practice, this means that the point in which the thermal emergency is set determines the ratio of the heat up over the cooldown times. Since the thermal emergency temperature is set by physical constraints, the point in the curve where the emergency is set depends on the curve's shape (determined by the thermal properties of the system) and amplitude (power dissipated by the system). If the thermal emergency is set at the lower part of the curve then heat up will happen faster than cooldown.

However, this reasoning ignores some important issues. The first is the global heating which alters the upper boundary of the curves and, thus, their slope. The second is the effect of lateral conductivity between cores. Finally, the most important one is leakage. If leakage is a significant percentage of total power, then the temperature-leakage feedback loop increases dramatically the slope of the heat-up curve. This implies that an analysis must be done for each particular design.

In our case, cooldown times are faster than heat-up times (for the base temperature limit). In the different configurations considered in this work, it takes more time to heat up back to the original temperature than it takes to cooldown after a stop. Fig. 2 shows this phenomenon. In the figure, we can see the temperature evolution of each core of a 16-core system running random applications.

If cooldown is faster than heat up, then a single stop interval keeps a core under the thermal limit for the current

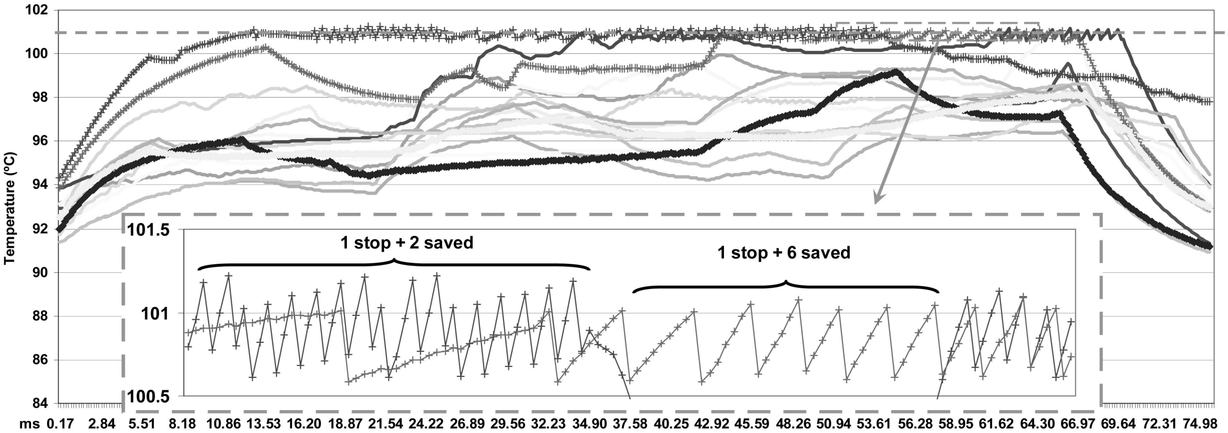


Fig. 2. Heat-up and cooldown times in a 16-core microarchitecture.

(stopped) interval and for several of the subsequent intervals until temperature is increased again (this is shown in Fig. 2). For instance, if in a thermally unconstrained execution, 50 percent of the intervals were over the limit and the average cooling ratio happens to be 1:10 (one cooldown interval requires 10 heat-up intervals to reach the original temperature), then the number of intervals over the threshold (and the performance loss for a thermally constrained execution) after applying *stop&go* will be roughly *50 percent/11 = 4.5 percent*.

2.2 Decision Granularity of DTM Control Mechanisms

The RC is a time metric that measures how fast temperature increases and decreases. The RC at the die level is typically between 1 ms and 10 ms [33]. During this interval, the core temperature may increase significantly (in 3×RC the temperature swing is 95 percent of the total). In our configuration, the approximate RC values are 3.75 ms at the die level, 225 ms at the heat spreader level and 100 s for the sink-ambient convection path. The maximum temperature variation seen in a core after the thermal model initialization is around 0.5°C (the exact value depends on the configuration). This implies that to have efficient DTM control, decisions must be made at a fine granularity [27].

On the other hand, some of the previous studies implement TM at the operating system level, with migration decisions made at intervals of tens of milliseconds [11]. In order to analyze the impact of the frequency of migration decisions on performance, we perform a sensitivity analysis of this parameter.

For all of the above reasons, we have decided to restrict our analysis to hardware-only DTM techniques with a base control interval of 167μs.

2.3 Thread Migration Policies

TM requires a mechanism to transfer architectural state (and optionally, microarchitectural state too) from one core to another. First, the migrating thread is stalled, whereas its state sent through the interconnection network to the other core, where it is restored to resume the execution. This implies flushing some structures as, for instance, the TLB to preserve correctness in the address translation.

When evaluating the benefits of DTM schemes that utilize TM the assumptions, we make about that migration costs are very important: An artificially large migration cost may result to excessive performance loss, and an unrealistically low migration cost may give misleading results.

Fig. 3 shows performance (normalized to maximum) of two thermal control schemes in our base configuration: *Stop&go* with and without TM. It can be seen that, for high migration costs, the total execution time is increased although less stops are required. On the other hand, if the migration cost is small enough, TM is able to reduce total execution time.

Even though the TM cost should be analyzed for each particular design, it is expected to be small compared to a stop interval. Our base analysis assumes a total cost of 10,000 cycles (that is, 2 percent of an idle interval of 167 μs at a reference frequency of 3 GHz). This assumption is consistent, and even conservative compared to measured average context switch cases published elsewhere [25], [29].

2.4 Thermal Impact of the Number of Cores

As pointed out in [22], when the number of cores increases, the temperature of a core depends more on global rather than local heat dissipation. The reason for this phenomenon is that the heat spreader and the heat sink gather the heat generated by all cores. As a consequence, the temperature in both spreader and sink is increased, and this affects the temperature in all cores. This effect is similar to increasing the ambient temperature in a single-core chip. The result is that as the

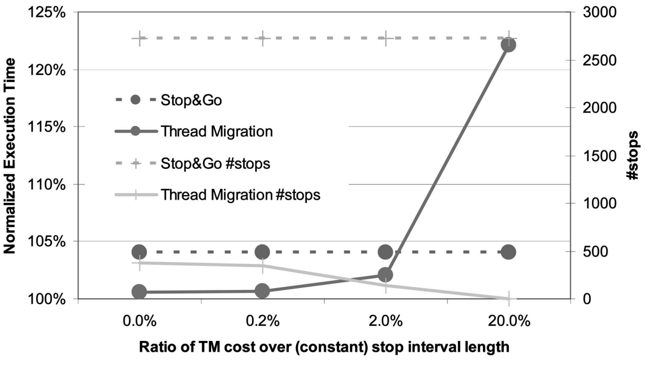


Fig. 3. Impact of TM cost.

number of cores increases, the weight of a core to the local temperature decreases relative to global heating [23].

On the other hand, the total amount of heat that can be removed is limited. This means that cores cannot be added "for free" unless the power per core is reduced. This drives multicore designs toward the integration of simpler and more power-efficient cores [22]. Although designs with two and even four cores already exhibit these problems, designs with 16 or more cores are far beyond simple multicore designs and present further challenges.

Floorplanning is also a differentiating factor. A 4-core design can employ a relatively simple and homogeneous floorplan. The variety of core locations is very limited so that the same thread in different cores is likely to have a low variability in its thermal behavior. However, the scenario for 16 or more cores is much broader. Potentially, the number of *unique locations*[1] is equal to the number of cores (if their internal structure is not symmetrical). In a fully symmetrical floorplan, a quadcore design would have only one unique location, whereas a 16-core design would have four.

Our work is different from previous studies because of the following reasons:

- This is the first work that studies the thermal behavior of a 16-core CMP architecture using a detailed thermal model. This scenario increases the likelihood of thermal emergencies. Also, the complexity of the thermal map allows for new combinations of TM and voltage/frequency scaling.
- So far, DTM for multicore studies has ignored lateral heat dissipation across cores resulting to significant loss of accuracy. Lateral dissipation has been ignored because of complexity reasons in the thermal model. However, lateral dissipation is known to be an important parameter in heat modeling [32]. In addition, a new generation of materials (Si28 [28]) may increase the lateral dissipation of silicon exacerbating this error. Our work models a full system including all lateral heat paths.
- Previous works assumed some parameters such as maximum temperature, thermal solution, and so forth, to be fixed. In this work, we analyze the sensitivity of the different thermal management schemes to the value of these parameters.
- We evaluate the performance of the thermal management schemes under two different types of scenario. First, we assume that the processor has to run 1,000 traces randomly chosen among 296. Second, we assume a scenario consisting of two parallel applications running hot and cold traces. The first scenario represents a system running multiple independent applications. The second scenario represents a system running two parallel applications, one of them composed by cold threads and the other composed by hot threads.
- We propose improvements over existing TM schemes, as well as propose new ones.

3 Simulation Infrastructure for Multicore Architectures

This work assumes a system including 16 identical cores. Simulating a multicore environment is very challenging and even more so if the length of the simulations must be long enough to obtain a rich variety of thermal behaviors. The way that we have used to handle simulation complexity is to simplify the problem by compacting cycle-by-cycle events and concentrate on events that change the average system behavior in the long run.

TABLE 1
Core Configuration

ISA	IA32	Technology	45nm
Fetch	6k *micro-ops* trace cache, 2-way 5 cycles fetch-to-dispatch 3 *micro-ops*/cycle		
ROB	64 entries 3 *micro-ops* committed/cycle		
Issue Queue	8 entry prescheduler queue 32 entry issue queue 1 int + 2 int/fp + 1fp/@ + 1store issue per cycle		
Register File	96 integer 96 FP		
Data Cache	16kB, 4-way 2 cycles hit 48 entries Load/Store queue, 1 op/cycle		
L2	512kB, 8-way 24 cycles hit, ≥175 miss 1.6GHz FSB		

3.1 Performance Metrics

Any metric used to measure the effectiveness of thermal management should reflect the user's perception about the time consumed by his/her *workload* (a set of *applications* to be executed).

In this work, total execution time—our metric—is computed as the sum over all cores of the total time it took each core to complete all the applications it was assigned. This way, we account for the total amount of work done per core including *wasted intervals.* A wasted interval is a time interval during which a core is idle due to the triggering of a thermal emergency mechanism.

3.2 Thermal Emergency Temperature

Designs are typically optimized for the common case application. This means that, usually, the maximum allowed temperature is set so that the *average application* does not loose performance because of DTM. Applications with higher than average power consumption suffer penalizations. However, performance degradation because of thermal limitations is no longer a rare event. In our experiments, we set our base thermal limit so that 23 percent of the thermal samples collected during a thermally unrestricted simulation are above it (the 77th percentile of the temperature samples).

3.3 Power and Performance Traces

A database has been built with 296 traces of 10 M instructions each. These traces represent a wide variety of applications (multimedia, SPEC, kernels, databases, and so forth). To build the database, each trace is executed independently in a power/performance simulator of an IA32 core (whose details are shown in Table 1). Every 10,000 instructions, the simulator outputs power/performance statistics that describe the behavior of the application during this reconfiguration interval. Furthermore, in order to allow for dynamic voltage scaling, each trace is simulated for all voltages and frequencies shown in Table 2 and is added to the database separately.

The power model used in our simulator is based on assigning energy cost per usage of every core's structure similarly to Wattch [4]. Energy per event is computed using

[1] Two cores belong to the same *unique location* if they cannot be matched by transforming the floorplan using rotations and axial symmetry.

TABLE 2
Voltage/Frequency Level Information

Level	V_{dd}	Freq.	Application power (W)		
	(V)	(GHz)	Avg.	Max.	Min.
0	0.7	1.7	0.62	1.23	0.33
1	0.725	1.8	0.70	1.40	0.37
2	0.75	1.9	0.79	1.58	0.42
3	0.775	2.1	0.92	1.86	0.48
4	0.8	2.2	1.02	2.08	0.53
5	0.825	2.3	1.13	2.31	0.59
6	0.85	2.4	1.24	2.56	0.64
7	0.875	2.5	1.37	2.83	0.71
8	0.9	2.6	1.49	3.11	0.77
9	0.925	2.7	1.63	3.41	0.84
10	0.95	2.8	1.77	3.73	0.91
11	0.975	2.9	1.93	4.07	0.99
12	1	3.0	2.08	4.43	1.07

an enhanced version of Cacti [30]. Application power consumption is depicted in Table 2.

3.4 Performance and Power Simulation

The multicore simulator generates a workload to be executed by selecting randomly 1,000 applications among the 296 available traces (traces can be repeated). Each core requests a new trace to be run as soon as it finishes its current trace. This is repeated until the 1,000-application workload is exhausted.

The multicore simulator makes thermal related decisions at the granularity of 167μs. For each reconfiguration interval, power and performance metrics are estimated by reassembling the proper information from the database. Each core computes the number of instructions completed in this interval of time at the current frequency by looking up that information in the database.

At the end of every interval, the control mechanism is activated, and it decides the new configuration for the next interval (thread allocation, cores that will be stopped, and voltage/frequency levels for all cores). This environment allows for extremely fast simulation while preserving reasonable accuracy.

3.5 Thermal Simulation

Our thermal model is based on the RC duality between electrical and thermal phenomena. Such dynamic compact models have been used in the recent years for microarchitectural research [17], [32]. In these models, an RC circuit containing nodes (points where temperature is measured) and connections (modeling heat flow between nodes) is built to express the thermal behavior of the microprocessor. A thermal RC circuit can be solved similarly to an electrical RC circuit. More details on the thermal model can be found in [6].

3.5.1 High-Resolution Thermal Modeling

Solving an RC circuit with many nodes is very expensive. This is especially important for multicore systems where the number of nodes increases linearly with the number of cores unless the amount of nodes per core is reduced. For this reason, the thermal simulation of a multicore system is a challenge even for a small number of cores.

The solving cost can be decreased by reducing the number of nodes per core. However, modeling different nodes inside a core is important since hot spots typically arise in a small part of the core and do not affect homogeneously the area of the core. This is the reason why reduction of resolution is often discarded.

Previous studies overcome the complexity problem by removing lateral heat dissipation among cores. This approach ignores the lateral dissipation on the core boundaries, which could be as significant or, even more, as lateral dissipation inside a core (especially with the introduction of new materials [28]). In this work, in order to be more accurate, we utilize full in-core resolution and full lateral dissipation. A multinode heat spreader and a multinode die layer are modeled as well, where each core is modeled by nine different nodes, at least. This is required in order to capture the thermal heterogeneity inside the core. In addition, lateral dissipation is accounted for, both inside cores and among cores. To achieve reasonable solving times for the thermal model the solver has been extremely optimized.

3.5.2 Leakage Model

Leakage is modeled using an analytical equation similar to that used in [35] that relates leakage to temperature and voltage. Leakage is recomputed every time temperature or voltage is updated. Although simulations are long enough to converge into the long-term temperatures, initial temperatures are set to representative values before starting the simulation. This allows for a faster warm-up period. The thermal circuit is fed by the average power of all applications and the leakage-temperature loop is run until simulation converges or the thermal limit is reached.

4 Multicore Dynamic Thermal Management

For the different TM techniques evaluated, several thermal sensors per core are assumed [31]. We also assume that thermal sensors can be read at every reconfiguration interval (that is, every 167 μs). Results are provided in groups: TM schemes, DVFS schemes, and combined TM-DVFS schemes. All techniques are run on top of *stop&go* to cover for the cases that the DTM mechanism is unable to control temperature.

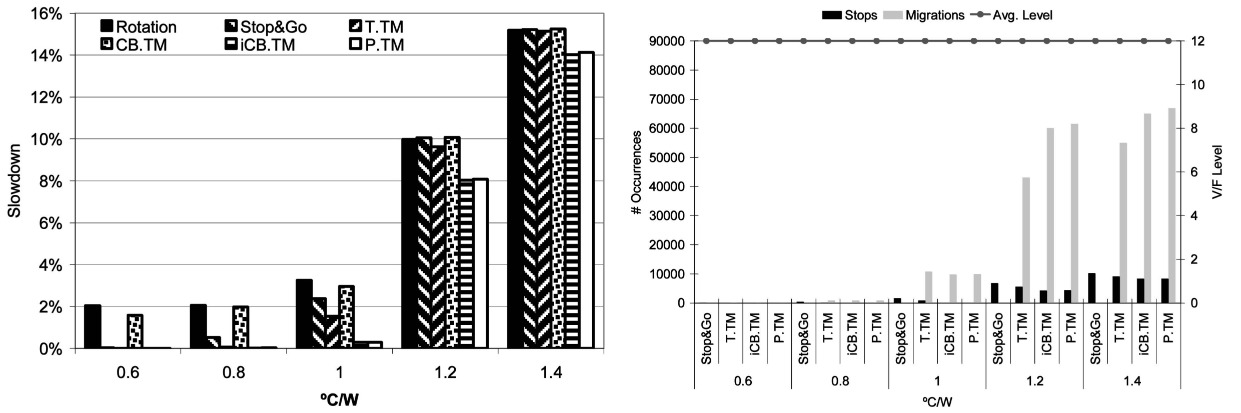
For each experiment, along with performance numbers, we provide with details on the number of *idle* intervals and TMs, as well as the average voltage and frequency level for each configuration. The performance of the DTM mechanisms is expressed as the slowdown with respect to an unrealistic thermally unlimited microarchitecture (so lower number is better). Note that the baseline we use is ideal (and unrealistic) since it assumes that the silicon can support any temperature.

In order to analyze the sensitivity of the DTM techniques with respect to the thermal solution, the results are presented for several thermal solutions (whose thermal resistance ranges from 0.6°C/W to 1.4°C/W). Experiments for thermal solutions ranging from 0.2°C/W to 0.6°C/W and from 1.4°C/W to 2°C/W have been also carried out, but they are not shown in the paper since they do not provide any insight. A small value in the thermal resistance indicates a good thermal solution (*low-R*), whereas a high value means a cheap thermal solution (*high-R*).

The base thermal limit is set to 102.5°C. This means that both proactive and reactive techniques set their goal to 102°C (the security margin is 0.5°C). We have also performed sensitivity analysis for several parameters: maximum temperature, TM cost, and stop length.

4.1 Thread Migration Schemes

Fig. 4a shows the penalty for *stop&go* and *stop&go* augmented with different TM policies. Fig. 4b shows the



(a) (b)

Fig. 4. (a) TM policies performance and (b) TM details (*Rotation* and *CB.TM* do not provide any insight and are not shown to make the graph simpler).

number of idle intervals (*stops*) and migrations, as well as the average voltage/frequency level of the execution for each particular combination of thermal solution and TM policy. The TM policies that we evaluate are the following (assuming N cores, labeled from 0 to $N-1$):

- *Stop&Go*. No TM.
- *Rotation*. At the beginning of a reconfiguration interval, the thread in core i is migrated to core $(i+1)\ mod\ N$.
- *T.TM* (Temperature-based). At the beginning of a reconfiguration interval, the cores are ordered by temperature. The thread on core i is swapped with the thread on core $N-i-1$ (that is, the thread on the hottest core is swapped with the thread on the coldest core, the thread on the second hottest core with the one on the second coldest core, and so on).
- *CB.TM* (Counter-based). TM proposed in [11]. The cores are ordered according to temperature imbalance (delta between the highest and the second highest hotspots inside each core). Then, the list is traversed top-down. For each core in the list, the thread in the system that is predicted to maximize temperature reduction is migrated to that core.
- *iCB.TM*. Improved *CB.TM*. Cores are ordered by their temperature and not by the delta between their highest and the second highest hotspots. Migration is also disabled if temperature is more than 1°C under the limit.
- *P.TM* (Power-based). At the beginning of a reconfiguration interval the cores are ordered by their temperature (increasingly) and the threads by their power consumption in the last interval (decreasingly). Then, the thread in position i is assigned to core i (that is, the most power-hungry thread is assigned to the coldest core). Migrations are disabled if temperature is more than 1°C under the limit.

Rotation, *stop&go*, and *T.TM* represent the "naive" techniques. *CB.TM* was proposed in [11], whereas *iCB.TM* and *P.TM* constitute part of our proposal in this paper.

For the *iCB.TM* and *P.TM* schemes, migrations are disabled if the temperature difference with respect to the thermal limit is below 1°C. This improves performance, since migrations that, at the end, would not provide any performance benefit are avoided in low-R environments. This happens naturally in *T.TM* and *stop&go*. This has not been applied in *Rotation* to exemplify the impact of useless migrations in low-R configurations (Fig. 4a). Other migration cancellation schemes were also tested, but without any significant improvement.

For high-R thermal solutions, both *iCB.TM* and *P.TM* perform the best. The reason is that both are able to avoid emergencies by rearranging threads the best.

4.2 Dynamic Voltage and Frequency Schemes

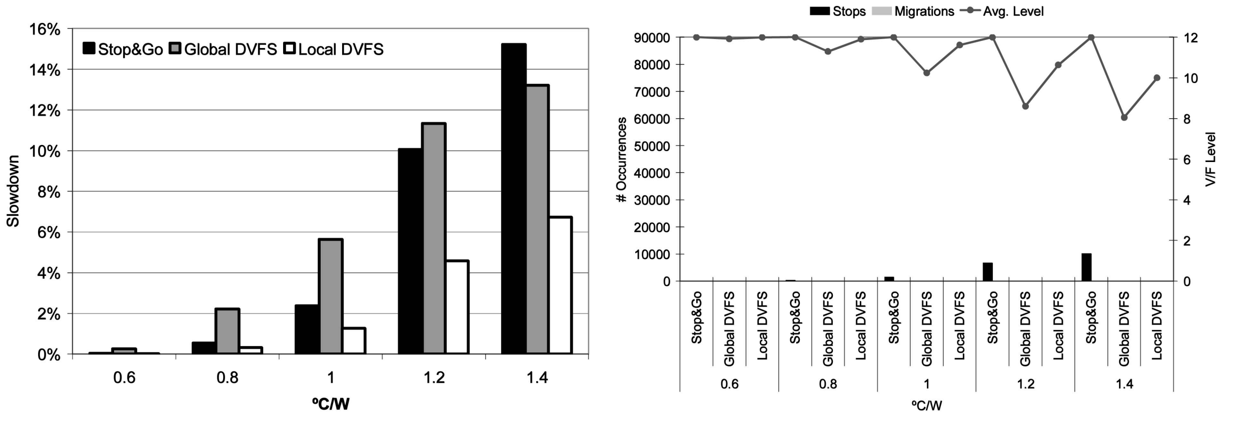
Fig. 5a shows the penalty of full-chip DVFS (Global), per-core DVFS (Local), and *stop&go*. Fig. 5b shows how different DVFS schemes change the average voltage/frequency levels and the number of idle intervals. We assume that the chip continues execution while changing voltage and frequency [13]. A PID controller (configured similarly to the one in [11]) is used to decide the voltage/frequency level per interval.

As expected, scaling down the voltage/frequency of the whole chip when a core is overheated harms total performance; it is better to just stop the hot core. However, if the thermal solution is of low quality (that is, high-R), most of the cores will trigger emergencies. In this case, it is better to scale down the whole chip, including cores that are running under the thermal limit than doing *stop&go*. This way, global heating is reduced, which helps to achieve faster cooldown times (the heat spreader and the heat sink are cooled faster).

The best performing scheme in all cases is Local DVFS that scales down only the overheating cores. However, Local DVFS is significantly more complex to implement than Global DVFS, so Global DVFS should not be discarded. For instance, Local DVFS is very likely to require more (internal) voltage regulators, it also requires each core on a different voltage island, and it complicates the power delivery network.

4.3 Combined Schemes

TM and DVFS are orthogonal techniques that can be combined to achieve additional benefits. Fig. 6a shows the performance penalty of different TM, and DVFS combinations. Fig. 6b gives further metrics for each technique (number of idle intervals, migrations and average voltage/frequency level). For comparison purposes, the performance penalty of

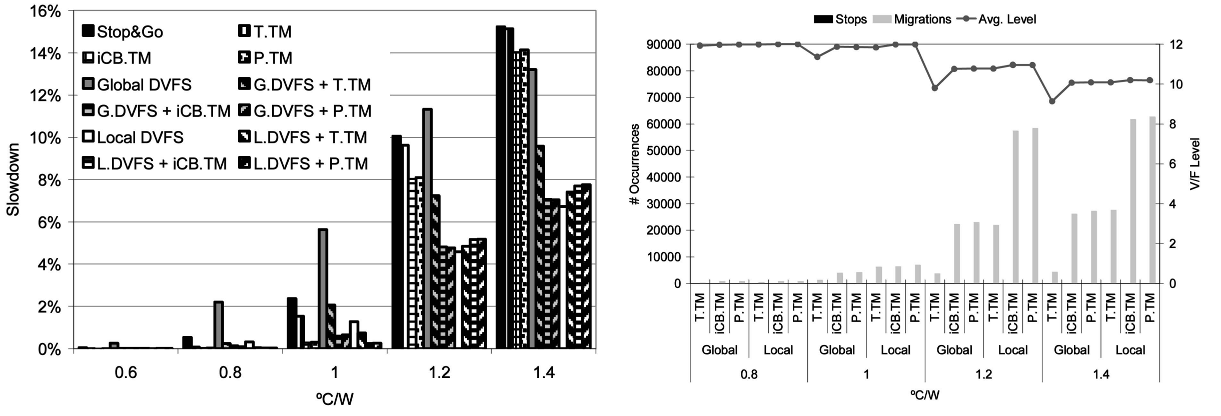


(a) (b)

Fig. 5. (a) DVFS penalty and (b) DVFS details.

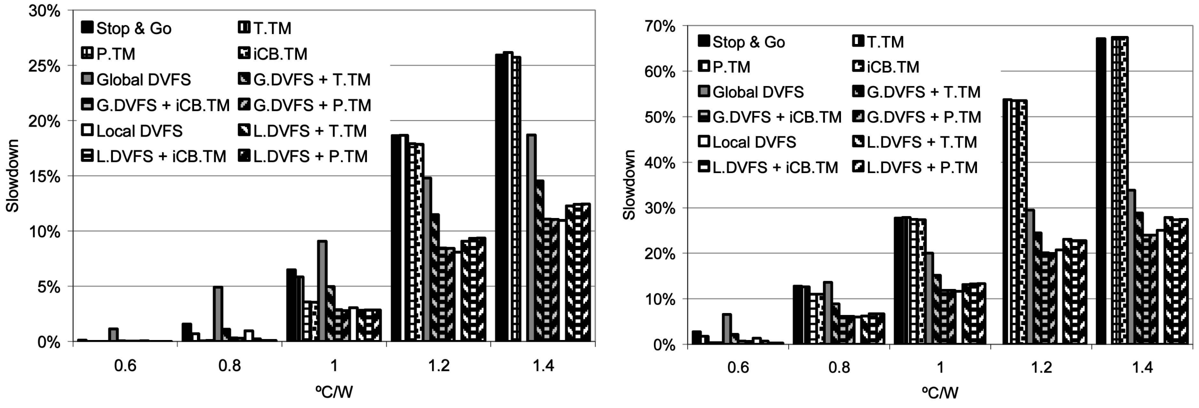
stop&go and of the noncombined TM and DVFS techniques are shown. Different interesting conclusions can be reached from the results:

- In most cases, Global DVFS is the worst technique because it slows down the whole chip.
- For cheap thermal solutions (high-R configurations), when all cores are likely to trigger stops, Global DVFS performs better than any TM scheme and *stop&go*. For these techniques, the number of stops is really large and affects most of the cores. Since doing DVFS is better than stopping and most of the cores are stopping, it is better to DVFS all of them together.
- TM only helps Local DVFS for high-performance thermal solutions. The small number of migrations allows for a slightly higher average core voltage/frequency level resulting in overall better performance. In these cases, *iCB.TM* is the best TM scheme to use with Local DVFS.
- For cheap thermal solutions (high-R), TM actually harms the performance of Local DVFS. The reason is the penalty of migrations that offsets any benefit that comes from DVFS. The less harming TM is *T.TM* because of the significantly smaller number of migrations.
- In high-R scenarios, TM achieves better temperature reductions combined with Global DVFS than with Local DVFS. The number of TMs is higher with Local DVFS than with Global DVFS. There is a subtle reason for that: let us assume an order for the cores from the core with the highest temperature to the core with the lowest temperature. The number of TMs is highly dependent on the number of changes on this order. The only reasons why this order changes when using Global DVFS are context switches and changes in the application's power (due to phase changes). With Global DVFS, there are no changes of the order among cores due to voltage/frequency scaling, since all cores change simultaneously and by the same amount. However, with Local DVFS, voltage/frequency is changed independently for each core. Therefore, in this case, apart for context switches and application power consumption, DVFS is also a source of changes in the order of the cores. This is the reason that we have more TMs with Local DVFS.
- TM and Global DVFS achieve an excellent synergy. TM compensates the lack of core-level granularity of Global DVFS.



(a) (b)

Fig. 6. (a) Performance of combined (TM + DVFS) techniques and (b) execution details. To make the graph easier to read, the right-hand graph does not show results for 0.6°C/W (they are the same as 0.8°C/W).



(a) (b)

Fig. 7. Performance penalty of combined schemes for reduced thermal limits. (a) $100.5^\circ\mathrm{C}$. (b) $96^\circ\mathrm{C}$.

- The performance of Global DVFS+TM is very close to Local DVFS. It provides a good trade-off between the reduced number of migrations and the average core frequency/voltage level (slightly lower than the one in Local DVFS).

In summary, for low-R configurations (expensive thermal solutions), the best technique is Local $\mathrm{DVFS} + iCB.TM$, whereas for high-R configurations (average and cheap thermal solutions), the best scheme is Local DVFS. An important conclusion already discussed in [11] is the competitive performance of Global DVFS + TM.

4.4 Sensitivity Analysis

4.4.1 Maximum Temperature

This section evaluates the sensitivity of DTM techniques to the thermal limit. Reducing this limit below the "crash limit" (temperature beyond which the chip suffers a failure) may be interesting for durability and reliability reasons. Also, the maximum allowed temperature may be constrained in handheld devices (even laptops) for the users' comfort and for usability reasons.

Lowering the thermal threshold creates a scenario in which more applications are likely to exceed the limit (similarly to high-R). In this case, a 2°C difference increases the change of triggering an emergency from 22.5 percent to 58.75 percent.

Fig. 7a shows the performance when the maximum allowed temperature is set to $100.5^\circ\mathrm{C}$ ($2^\circ\mathrm{C}$ under the previous limit) and Fig. 7b when the maximum temperature is set to $96^\circ\mathrm{C}$. Note that, different scales had to be used among previous figures, Fig. 7a and Fig. 7b. The performance penalty is roughly doubled for a limit of $100.5^\circ\mathrm{C}$ and quadrupled for a limit of $96^\circ\mathrm{C}$. In these scenarios, the relative difference between Local and Global techniques is reduced. The reason is that in this particular environment, most of the cores trigger an emergency. Local algorithms make more sense when only few cores reach a thermal emergency at the same time. If a large number of cores trigger an emergency, the potential is reduced, as can be seen in the graphs.

With a limit of $100.5^\circ\mathrm{C}$, the best performing techniques remain the same. However, the difference among them is reduced. When we set the limit at $96^\circ\mathrm{C}$, it can be seen that the effect previously described is exacerbated. The relative performance of the Global and Local techniques is even closer. Actually, Global DVFS+TM outperforms all Local DVFS schemes in high-R configurations. This indicates that having a fine-grain scheme is not useful in extreme environments (for example, a low-cost thermal solution with a very small thermal limit). In that case, the average frequency is very similar between Local and Global.

4.4.2 Dissipating Leakage on Idle Intervals

In our previous analysis, we assumed that during idle periods a core dissipates neither leakage nor dynamic power. To accomplish this, cores must save the context in order to go to the idle state. However, it may not be feasible to power down a core due to current swings in the power delivery network. For this reason, here, we analyze the performance of the different DTM schemes when leakage is dissipated during idle periods (that is, cores remain powered on).

Fig. 8 presents the additional performance penalty of various TM schemes in the base configuration. Since DVFS schemes successfully eliminate most stops (that is, idle periods), we do not expect their performance to change when we assume leakage during idle periods. As the thermal solution is less aggressive (high-R), the higher number of stops (which are less effective because of leakage) results in a lower performance (2.5 percent extra penalty in $1.4^\circ\mathrm{C/W}$). As expected, techniques with a higher ability to avoid stops are slightly less impacted.

4.4.3 Thread Migration and Stop Costs

In intervals where *stop&go* is triggered, the processor remains idle for a certain amount of time. The length of such intervals determines what is the penalty associated to a thermal emergency. Recall that *stop&go* is used as a backup mechanism when the main DTM policy fails to control peak temperature, and the hot spot of a particular core is reaching the thermal limit. Therefore, assuming a low penalty (1 reconfiguration interval) for *stop&*go tends to reduce the difference between ineffective DTM techniques

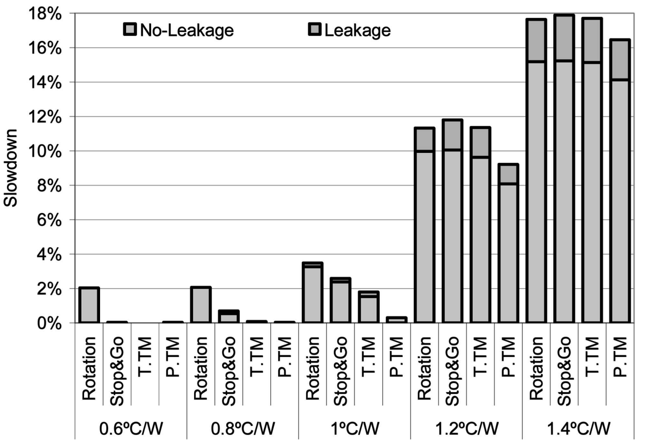


Fig. 8. Extra TM performance penalty for cores dissipating leakage in idle state.

(the ones that make an extensive use of *stop&go* as backup mechanism) and the effective ones.

This has been experimentally tested, as shown in Fig. 9a. For small stop intervals, the performance penalty of *stop&go* is only 1 percent (for 1°C/W) where Local DVFS is almost 0 percent. This is a small difference, taking into account that for larger stop intervals, the performance penalty of *stop&go* increases, whereas Local DVFS remains insensitive to this parameter. For worse thermal solutions and/or lower thermal thresholds, the difference between the two techniques grows even larger. The worse the thermal solution, the faster the difference grows.

When TM is utilized for thermal management, it is important to find out the relationship between the TM cost and the benefit that it provides. The TM cost is somewhat hard to quantify since it is very dependent on the microarchitecture (for example, the amount of state that must be transferred depends on number of registers, including control registers, and the internal state of the CPU). Fig. 9b shows the sensitivity study of TM cost when Global DVFS is employed. It is clear from that figure that increasing the TM cost reduces its benefit. However, on one hand, Global DVFS cannot effectively reduce the temperature of a particular hot core with no impact on the rest of the processor. On the other hand, TM is very effective when the temperature of a small number of cores wants to be changed, since it acts in a per-core basis. Therefore, the combination of Global DVFS with TM gets important benefits even for high TM costs due to their synergistic effect.

Fig. 9c shows a sensitivity study of TM cost when Local DVFS is used. For low migration cost, TM helps Local DVFS control the temperature in low-pressure environments. If the migration cost grows, performance actually degrades.

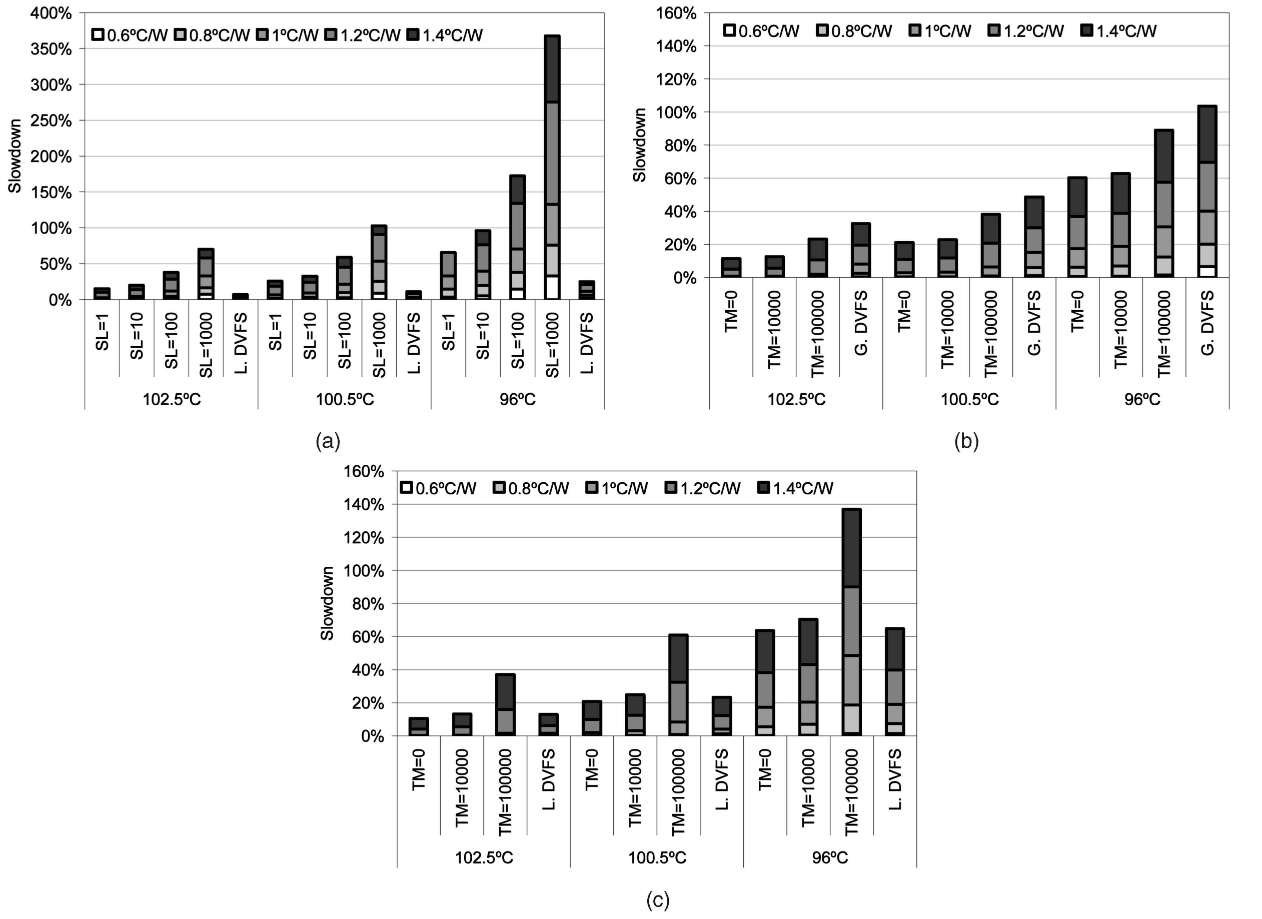


Fig. 9. Sensitivity analysis of (a) stop length using (of $167\mu s$ intervals), (b) TM cost in cycles for global DVFS, and (c) TM cost in cycles for local DVFS. (a) *Stop&go* versus local DVFS, (b) global DVFS without iCB.TM, and (c) local DVFS without P.TM.

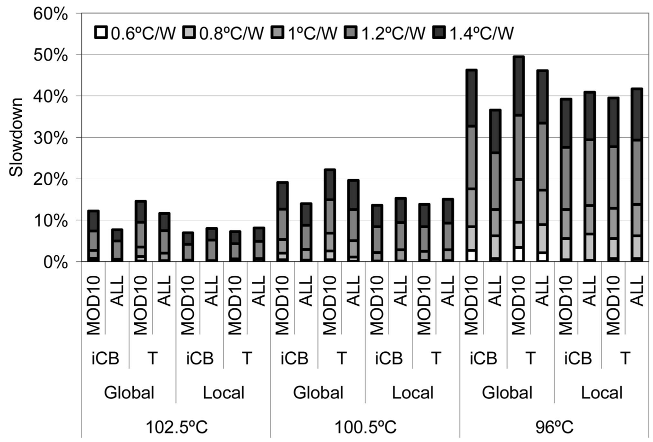


Fig. 10. TM reconfiguration interval comparison for different DVFS+TM schemes and thermal configurations.

Even for bad thermal solutions and low migration cost, performance can degrade. This is because faster temperature swings that lead to more migrations that results to a reduction in performance.

4.4.4 Reconfiguration Interval

The reconfiguration interval is an interesting parameter that has to be tuned when implementing interval-based DTM mechanisms. If it is too short, DTM decisions may not have enough time to become effective. If it is too long, it may result to slow reaction times that penalize performance. In addition, each DTM scheme has a different optimal interval length.

Fig. 10 compares TM decisions made at every interval (*ALL*) and every 10 intervals (*MOD10*) for Local and Global DVFS augmented with iCB.TM, and T.TM. DVFS decisions are made at the minimum granularity. Reconfiguring every 10 intervals was experimentally found the best TM reconfiguration latency. The results prove that different techniques may need different reconfiguration intervals. Global DVFS is penalized by the longer reconfiguration interval, whereas Local DVFS benefits slightly.

4.4.5 Homogeneous Workloads

This section briefly comments our initial evaluation of the performance of DTM techniques for a particularly interesting scenario: We assume that the processor is running 16 threads, eight of them "cold" and eight of them "hot." This approximates a system running two parallel applications with different power profiles. In this scenario, the performance penalty of a *stop&go* approach might be as high as 19 percent. However, if power-based TM is applied, most of the idle periods could be removed, reducing the performance penalty to 5 percent—using the original thermal parameters (see beginning of Section 4) and a maximum temperature of 93°C. An exhaustive analysis of the thermal behavior of collaborative applications is left as part of our future work.

5 Conclusions

Multicore designs offer a new and rich scenario for DTM. TM schemes, as well as dynamic voltage and frequency scaling techniques seem the most promising features to boost performance while controlling the operating temperature.

In this work, we analyze the impact of the wide range of parameters that influence the performance of DTM for multicore designs. In particular, our work explores different thermal solutions, TM and idle costs, application combinations, as well as stop policies. We show that the following hold:

- Different system parameters result to different optimal management schemes and/or scheme configurations.
- Local DVFS is the most effective scheme in almost all configurations.
- TM with realistic cost adds only a modest performance improvement to Local DVFS.
- Global DVFS+TM performs quite well compared to Local DVFS. For High-R configurations (cheap thermal solutions), it can even do better. This is very important due to the simpler nature of Global DVFS.

Acknowledgments

This work was partially supported by the Spanish Ministry of Education and Science and FEDER funds, under grant TIN2004-03072.

References

[1] AMD Multi-Core Technology, http://multi-core.amd.com, 2006.
[2] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23-29, 1999.
[3] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," *Proc. Seventh Int'l Symp. High-Performance Computer Architecture*, Jan. 2001.
[4] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proc. 27th Int'l Symp. Computer Architecture*, pp. 83-94, 2000.
[5] P. Chaparro, J. González, and A. González, "Thermal-Aware Clustered Microarchitectures," *Proc. Int'l Conf. Computer Design*, Oct. 2004.
[6] P. Chaparro, J. González, and A. González, "Thermal-Effective Clustered Microarchitectures," *Proc. First Workshop Thermal Aware Computer Systems*, June 2004.
[7] P. Chaparro, G. Magklis, J. González, and A. González, "Distributing the Frontend for Temperature Reduction," *Proc. 11th Int'l Symp. High Performance Computer Architecture*, Feb. 2005.
[8] P. Chaparro, G. Magklis, J. González, and A. González, "Using MCD-DVFS for Dynamic Thermal Management Performance Improvement," *Proc. 2006 Intersociety Conf. Thermal and Thermomechanical Phenomena in Electronic Systems (ITHERM '06)*, June 2006.
[9] T. Constantinou, Y. Sazeides, P. Michaud, D. Fetis, and A. Seznec, "Performance Implications of Single Thread Migration on a Chip Multi-Core," *SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 80-91, 2005.
[10] V. De and S. Borkar, "Technology and Design Challenges for Low Power and High Performance," *Proc. 2000 Int'l Symp. Low Power Electronics Design*, pp. 163-168, 2000.
[11] J. Donald and M. Martonosi, "Techniques for Multi-Core Thermal Management: Classification and New Exploration," *Proc. 33rd Ann. Int'l Symp. Computer Architecture*, June 2006.
[12] K. Flautner, N. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy Caches: Simple Techniques for Reducing Leakage Power," *Proc. 29th Int'l Symp. Computer Architecture*, June 2002.
[13] S. Gochman et al., "The Intel Pentium M Processor: Microarchitecture and Performance," *Intel Technology J.*, vol. 7, no. 2, May 2003.
[14] S. Gunther, F. Binns, D.M. Carmean, and J.C. Hall, "Managing the Impact of Increasing Microprocessor Power Consumption," *Intel Technology J.*, vol. Q1, 2001.

[15] L. Hammand, B.A. Nayfeh, and K. Olukotun, "A Single-Chip Multiprocessor," *Computer*, pp. 79-85, Sept. 1997.

[16] S. Heo, K. Barr, and K. Asanović, "Reducing Power Density through Activity Migration," *Proc. 2003 Int'l Symp. Low Power Electronics and Design,* 2003.

[17] W. Huang, E. Humenay, K. Skadron, and M.R. Stan, "The Need for a Full Chip and Package Thermal Model for Thermally Optimized IC Designs," *Proc. 2005 Int'l Symp. Low-Power Electronics Design,* pp. 245-250, Aug. 2005.

[18] IBM Cell Processor, http://researchweb.watson.ibm.com/journal/rd/494/kahle.pdf, 2006.

[19] Intel Core 2 Duo, http://www.intel.com/products/processor/core2/index.htm, 2006.

[20] 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," *Proc. 36th Int'l Symp. Microarchitecture,* Dec. 2003.

[21] J. Li and J.F. Martínez, "Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors," *Proc. Int'l Symp. High-Performance Computer Architecture,* Feb. 2006.

[22] Y. Li, B.C. Lee, D. Brooks, Z. Hu, and K. Skadron, "CMP Design Space Exploration Subject to Physical Constraints," *Proc. 12th Int'l Symp. High Performance Computer Architecture,* Feb. 2006.

[23] Y. Li, K. Skadron, Z. Hu, and D. Brooks, "Performance, Energy, and Thermal Considerations for SMT and CMP Architectures," *Proc. 11th Int'l Symp. High Performance Computer Architecture,* pp. 71-82, Feb. 2005.

[24] C.H. Lim, W.R. Daasch, and G. Cai, "A Thermal-Aware Superscalar Microprocessor," *Proc. 2000 Int'l Symp. Quality Electronic Design,* Mar. 2002.

[25] Linux Kernel Archive, "Context-Switch Cost on Linux/ia32," http://uwsg.ucs.indiana.edu/hypermail/linux/kernel/0001.2/1207.html, 2006.

[26] J. Moore, R. Sharma, R. Shih, J. Chase, C. Patel, and P. Ranganathan, "Going beyond CPUs: The Potential of Temperature-Aware Solutions for the Data Center," *Proc. 2004 First Workshop Temperature-Aware Computer Systems (TACS-1) Held in Conjunction with ISCA-31,* June 2004.

[27] M.D. Powell, M. Gomaa, and T.N. Vijaykumar, "Heat-and-Run: Leveraging SMT and CMP to Manage Power Density through the Operating System," *Proc. 11th Int'l Conf. Architectural Support for Programming Languages and Operating Systems,* Oct. 2004.

[28] R. Prasher, R. Mahajan, and C-P. Chiu, "Thermal Management of High Performance Silicon," *Proc. Tutorial at the Intersociety Conf. Thermal and Thermomechanical Phenomena in Electronic Systems (ITHERM '06),* 2006.

[29] J.D. Regehr, *Using Hierarchical Scheduling to Support Soft Real-Time Applications in General-Purpose Operating Systems,* chapter 10, http://www.cs.utah.edu/~regehr/papers/diss/doc-wwwch10.html, 2006.

[30] P. Shivakumar and N.P. Jouppi, "CACTI 3.0: An Integrated Cache Timing, Power and Area Model," WRL Research Report 2001/2, 2001.

[31] B. Sinharoy, "POWER5 Architecture and Systems," *Proc. Int'l Symp. High Performance Computer Architecture, Keynote Presentation,* Feb. 2004.

[32] K. Skadron, M.R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-Aware Microarchitecture," *Proc. 30th Ann. Int'l Symp. Computer Architecture,* Apr. 2003.

[33] K. Skadron and M. Stan, *A Quick Thermal Tutorial.* Univ. of Virginia, 2005.

[34] L.-T. Yeh and R.C. Chu, *Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods and Design Practices.* ASME Press, 2002.

[35] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotleakage: A Temperature-Aware Model of Subthreshold and Gate Leakage for Architects," Technical Report CS-2003-05, Dept. Computer Science, Univ. of Virginia, Mar. 2003.

Pedro Chaparro received the MS degree in computer science and computer engineering in January 2003 from the Universitat Politècnica de Catalunya (UPC) in Barcelona. He started the PhD degree in September 2004 in the Computer Architecture Department at the UPC. He joined the Intel-UPC Barcelona Research Center, which is part of the Intel Microprocessor Technology Labs, in 2002. His PhD thesis and his work at the Intel-UPC lab both focus on power and thermal-aware microarchitectures. Recently, his research also focuses on reliability issues.

José González received the PhD degree in computer science in 2000 from the Universitat Politècnica de Catalunya (UPC) in Spain. Then, he moved to the Universidad de Murcia, when he became an associate professor in 2001. He joined Intel in 2002 in the new Microprocessor Technology Lab in Barcelona, where he is currently working. His research is focused on processor microarchitecture, especially targeted to power-efficient multicore systems. He has published more than 30 papers in journals and conferences, and he has been a member of the IEEE Computer Society since 2000.

Grigorios Magklis received the BSc degree from the Computer Science Department, University of Crete in 1999 and the PhD degree in computer science from the University of Rochester, New York, in 2005. Since 2003, he has been working as a senior research scientist at the Intel-Universitat Politècnica de Catalunya (UPC) Barcelona Research Center, which is part of the Intel Microprocessor Technology Labs. His research is focused on microarchitecture and application analysis, with special emphasis on power efficiency and adaptability. He has published more than 20 papers in journals and conferences. He has been a member of the ACM since 1999 and of the IEEE Computer Society since 2002.

Qiong Cai received the PhD degree in computer science in 2006 at the University of New South Wales, Australia. He joined Intel-Universitat Politècnica de Catalunya (UPC) Barcelona Research Center as a senior research scientist in 2005 and has been working in projects related to power and thermal-aware microarchitectures.

Antonio González received the MS and PhD degrees from the Universitat Politècnica de Catalunya (UPC), Barcelona. He joined the faculty of the Computer Architecture Department at UPC in 1986 and became a full professor in 2002. He is the founding director of the Intel-UPC Barcelona Research Center, started in 2002, whose research focuses on new microarchitecture paradigms and code generation techniques for future microprocessors. He has published more than 200 papers, has given more than 80 invited talks, has filed more than 20 patents, and has advised 13 PhD thesis in the areas of computer architecture and compilers. He is an associate editor of the *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *ACM Transactions on Architecture and Code Optimization*, and the *Journal of Embedded Computing*. He has served on more than 100 program committees for international symposia in the field of computer architecture, including ISCA, MICRO, HPCA, PACT, ICS, ICCD, ISPASS, CASES, and IPDPS. He has been program (co)chair for ICS 2003, ISPASS 2003, MICRO 2004, and HPCA 2008, among other symposia. He is a member of the IEEE.