Shared Resource Aware Scheduling on Power-Constrained Tiled Many-Core Processors

Sudhanshu Shekhar Jha
Universitat Politécnica de Catalunya, Spain

Wim Heirman
Intel Corporation
Belgium

Ayose Falcón
HP
Spain

Jordi Tubella
Universitat Politécnica de Catalunya, Spain

Antonio González
Universitat Politécnica de Catalunya, Spain

Lieven Eeckhout
Ghent University
Belgium

ABSTRACT

Power management through dynamic core, cache and frequency adaptation is becoming a necessity in today's power-constrained many-core environments. Unfortunately, as core count grows, the complexity of both the adaptation hardware and the power management algorithms increases. In this paper, we propose a two-tier hierarchical power management methodology to exploit per-tile voltage regulators and clustered last-level caches. In addition, we include a novel thread migration layer that (i) analyzes threads running on the tiled many-core processor for shared resource sensitivity in tandem with core, cache and frequency adaptation, and (ii) co-schedules threads per tile with compatible behavior.

1. INTRODUCTION

Industry-wide adoption of chip multiprocessors (CMPs) is driven by the need to maintain the performance trend in a power-efficient way on par with Moore's law [17]. With continued emphasis on technology scaling for increased circuit densities, controlling chip power consumption has become a first-order design constraint. Due to the end of Dennard scaling [6] (slowed supply voltage scaling), we may become so power-constrained that we are no longer able to power on all transistors at the same time — *dark silicon* [8]. Runtime factors such as thermal emergencies [2] and power capping [10] further constrain the available chip power. Owing to all the above factors, power budgeting on many-core systems has received considerable attention recently.

Most existing power management schemes use a centralized approach to regulate power dissipation based on power monitoring and performance characteristics. Unfortunately, the complexity and overhead of centralized power management increases in quadratic/logarithmic fashion with core count [7]. We therefore propose a *two-tier hierarchical power manager* for tile-based many-core architectures; each tile consists of a small number of cores and a shared L2 cache within a single voltage-frequency domain. The two-tier power manager first distributes power across tiles, and then across cores within a tile. The architecture also provides support for core, cache and frequency adaptations to avoid core gating at moderate to stringent power budgets.

Tiled many-core processors pose an interesting challenge when it comes to hardware adaptation and scheduling. Changing frequency and reconfiguring the shared L2 cache affects all threads running in the tile. It therefore becomes important to *migrate* threads, such that threads with *compatible* behavior are co-scheduled onto the same tile. Since the execution behavior varies over time, periodic re-evaluation and dynamic thread migration is also required. We therefore classify threads based on their sensitivity to both cache and frequency dynamically at runtime. We propose DVFS and Cache-aware Thread Migration (*DCTM*): a scheduler running on top of the two-tier hierarchical power manager to ensure an optimal co-schedule for all threads running on the power-constrained tiled many-core processor while accounting for the effects of hardware adaptation.

2. MOTIVATION

2.1 Limitations of a Centralized Approach

In the context of power management in many-core processors, prior works [5, 12, 16, 18] have relied on a central entity to manage power using one or more micro-architectural techniques to trade off performance at high to moderate power budgets. At stringent power budgets, neither of power management schemes like DVFS nor core adaptation nor cache resizing *in isolation* can provide a viable solution. As a result, prior work [13, 14] had to resort to core gating at stringent power envelops. The Performance Monitoring Unit (PMU) keeps track of a core's activity and controls the micro-architectural configuration in response to requests made by the Global Power Manager (GPM); the GPM combines information from all cores and performs the global power/performance optimization, see *Centralized Approach* in Figure 1. But as core count continues to grow, the centralized approach becomes inviable (quadratic/logarithmic complexity). In future many-core processors [1], a centralized GPM (even with logarithmic complexity) would be a severe bottleneck. Because a centralized power manager does not scale favorably towards large many-core processors and fine-grain hardware adaptations, we propose *two-tier hierarchical* power management (see Section 3) — *first contri-*

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CF'16 May 16-19, 2016, Como, Italy

© 2016 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4128-8/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2903150.2903490

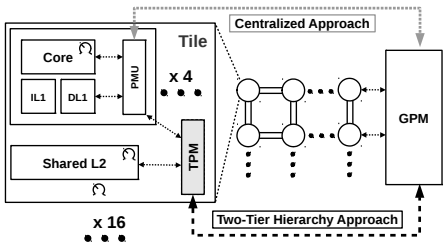

Figure 1: Generic tiled many-core architecture with centralized (top) versus hierarchical (bottom) power management.

bution in this paper.

2.2 Cache-aware Thread Migration (Cruise)

When threads are co-scheduled on a multi-core processor with a shared last-level cache (LLC), conflicting thread behavior can lead to suboptimal performance. Jaleel et al. [11] propose Cruise: a hardware/software co-designed scheduling methodology that uses knowledge of the underlying LLC replacement policy and application cache utility information to determine how best to co-schedule applications in multi-core systems with a shared LLC. Cruise monitors the number of LLC accesses per kilo instructions (APKI) and miss rate (MR) for each application. Application classification based on these metrics along with co-scheduling rules then optimize overall system performance[1]. Cruise assumes that all cores run at the same clock frequency. In other words, it does not take DVFS sensitivity into account. This is a limitation as LLCT and (especially) LLCFR applications, being mixed compute- and memory-bound, may be quite sensitive to frequency. We overcome this limitation by proposing DCTM (see Section 4) — *second contribution in this paper.*

3. TWO-TIER HIERARCHICAL POWER MANAGEMENT

The *Centralized* approach (Section 2.1) is inappropriate for large-scale many-core processors, for two reasons. First, it assumes per-core DVFS adaptation which is infeasible for many-core processors as it requires per-core on-chip voltage, which would incur fairly high chip area overhead [3]. Second, the runtime complexity and overhead of a *Centralized* approach increases considerably with core count.

To address these two limitations, we group cores per tile and add an intermediate layer for power management, the *Tile Power Manager (TPM)*; see *Two-Tier Hierarchy Approach* in Figure 1. Chip power is managed via a hierarchical power manager with a GPM steering the per-tile TPMs. This organization reduces the runtime overhead of the power manager dramatically. We observe that the overhead increases substantially with core count. However, when considering a tiled architecture and a two-tier hierarchical power manager, we are able to significantly reduce the runtime overhead of the power manager. In other words, by keeping the GPM relatively simple and passing more functionality to the TPMs, we avoid GPM to be a bottleneck at high core count. Moreover, as all TPMs can work in parallel, the complexity of the two-tier approach equals $O(G)$ + $O(T_c log T_c)$, with T_c denoting the number of physical cores per tile, and G the complexity of the GPM (constant in our case).

[1]Due to lack of space, we are unable to describe the application classification — CCF, LLCT and LLCFR/LLCF — and scheduling rules. Please refer to the Cruise paper [11].

4. DVFS AND CACHE-AWARE THREAD MIGRATION

A tiled many-core processor architecture with hierarchical power management, as we just established in the previous section, poses a new challenge as threads running on the same tile share the L2 cache (LLC) and a common clock frequency. In other words, and in contrast to Cruise, threads running on the same tile not only share the LLC but also share a common clock frequency. Therefore, it is important to take both cache size sensitivity and frequency sensitivity into account when mapping threads to tiles, i.e., the thread migration layer needs to be aware of the sensitivity to both DVFS and LLC size.

4.1 DVFS and LLC Sensitivity Analysis

To understand an application's sensitivity to clock frequency and LLC size, we set up the following off-line analysis. We run simulations with 55 SPEC CPU2006 application traces for 750 million instructions to observe the performance sensitivity with respect to both LLC and frequency settings. Figure 2 plots application performance sensitivity to frequency changes, expressed as the ratio between its performance reduction and the reduction in frequency that was applied. Applications are clustered by their LLC-aware classification type (following Cruise [11]), and plotted in ascending order of sensitivity within each cluster. We categorize applications into the following DVFS-aware classes, according to their performance sensitivity to DVFS:

- High Sensitivity (HS, $> 66\%$): These applications are highly sensitive to DVFS. The performance of these applications is severely affected when migrated to a tile running at low frequency, whereas performance improves significantly if they can be migrated to a higher-frequency tile. These applications are generally compute-bound.
- Moderate Sensitivity (MS, 35–66%): These applications are moderately affected by DVFS. Applications with a mix of compute-bound and memory-bound operations are grouped in this category.
- Low Sensitivity (LS, $< 35\%$): These applications degrade slightly when running at a low DVFS setting. It is therefore beneficial to reduce frequency as much as possible to save power. These applications are typically memory-bound.

When co-scheduling applications, the application categorization based on LLC usage (see Cruise, Section 2.2) needs to work in tandem with the DVFS sensitivity categorization as just described. Hence, combining the LLC and DVFS classifications, we have 3×3 categories of applications. Not all combinations occur in practice though, as there is some correlation between LLC and DVFS behavior; for instance, CCF applications are almost always compute-bound and hence have high DVFS sensitivity (HS). Figure 2 identifies five categories: LLCT with LS and MS, LLCFR with MS and HS, and CCF with HS.

4.2 DCTM Scheduling Rules

DVFS and Cache-aware Thread Migration (DCTM) leverages these classifications to steer scheduling of threads to tiles. The power manager will then assign the appropriate adaptation per tile (for frequency and LLC size) and per core (for core configuration). Intuitively speaking, DCTM maps threads with the same classification onto the same tile. Tiles with only LS threads will naturally be configured to run at low frequency (saving power without sacrificing per-

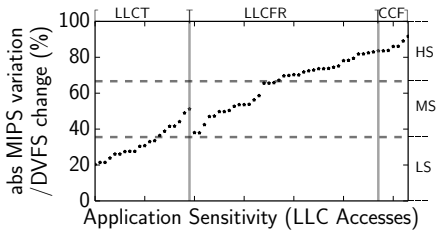

Figure 2: Application classification based on LLC and DVFS sensitivity.

formance much), while tiles with only HS threads preferably use a larger fraction of the total power budget to run at a higher frequency and boost overall system performance. In contrast, mixing LS, MS and HS threads on a single tile leads to a suboptimal situation: either the tile is set to run at low frequency, penalizing performance for the HS threads; or it runs at high frequency which accommodates the HS threads, but wastes power as it does not improve performance of the LS threads. We create the following scheduling rules for DCTM:

1. Co-schedule LLCT-LS applications on the same tile.
2. Co-schedule LLCT-MS applications on the same tile.
3. Co-schedule CCF-HS applications on tiles with LLCT-MS applications to account for performance impact due to shared LLC contention.
4. Co-schedule the remaining LLCFR-MS and LLCFR-HS applications on the remaining tiles. If possible, co-schedule them with LLCFR-HS applications that are in the CCF-HS category to avoid performance degradation due to shared LLC contention.

4.3 Putting It All Together

The DCTM power manager runs at two time scales. The coarse-grain timescale, at 20 ms in our setup, groups threads to tiles using the DCTM scheduling rules as just described in the previous section. One solution to classifying workloads in terms of LLC and DVFS sensitivity may be to employ sampling, i.e., by running a workload's performance at different frequency settings and different LLC sizes for short durations of time. The limitation is that it incurs significant overhead as we would need to monitor performance for various combinations of LLC size and frequency setting. Instead, we leverage the simple, yet effective analytical performance models proposed in [12] to estimate the performance impact of clock frequency and LLC size on overall performance.

The fine-grain timescale, at 1 ms in our setup, distributes power across tiles: the GPM distributes power across all tiles, and within each tile, the TPM regulates the hardware adaptations as per the allocated power. Our processor architecture allows three adaptations: core adaptation, LLC resizing, and per-tile DVFS, as we will describe in more detail in Section 5. The first fine-grained time slice (1 ms) assumes no power capping, and runs each thread at the maximum configuration (largest core configuration, largest LLC size, highest frequency). We compute the performance of each tile as a ratio of total system performance, i.e., per-tile MIPS divided by chip-wide MIPS. The GPM distributes the total available power budget across all tiles for the next time slice per the MIPS ratios of the tiles in the previous slice, i.e., a high-performance tile is given a larger fraction of the available power budget. The intuition is that compute-intensive tiles need a larger fraction of the total power, boosting overall system performance. Once total power is distributed across the tiles, the TPMs then decide on the optimal configuration for the core, LLC and DVFS setting in each tile. TPM steers adaptation using the performance/power models proposed in [12], with the goal of optimizing performance within the available power budget.

Parameter	Values			
Core adaptations				
ROB size	16	32	64	128
Reservation station entries	4	8	16	32
Load queue entries	6	12	24	48
Store queue entries	4	8	16	32
DVFS adaptations per-tile				
Frequency (GHz)	0.8	1.0	1.2	—
Voltage (V)	0.7	0.75	0.8	—
Shared LLC adaptations per-tile				
Cache ways	4	8	12	16
Capacity (KB)	512	1024	1536	2048

Table 1: Micro-architectural adaptations.

Component	Parameters
Core configuration	
Core type	4-way issue OOO, 128-entry ROB
Load/store queue	48 load entries, 32 store entries
L1-I cache	32 KB, 4-way, 3 cycle access time
L1-D cache	32 KB, 4-way, 3 cycle access time
Tile configuration	
Tile size	4 cores
Core count	64, 128, 256
Tile count	16, 32, 64
L2 cache (per-tile)	2048 KB, 16-way, 10 cycle access time
L2 prefetcher	stride-based, 8 independent streams
Coherence protocol	directory-based MESI, distributed tags
Network on Chip	mesh 16×1, 16×2, 16×4 32 GB/s/link
Main memory	8, 16, 32 controllers 80 ns latency, 128 GB/s total
Chip wide configuration	
Frequency-Vdd	1.2 GHz @ 0.8 V
Technology	22 nm
TDP	100 W, 190 W, 350 W

Table 2: Tile-based many-core architecture.

5. EXPERIMENTAL SETUP

Performance simulator. We use the Sniper multi-core simulator [4], version 6.0, and added support for dynamically changing core and cache parameters. The core adaptation and DVFS transitions combined take 2 µs during which no computations can be performed — a conservative approach.

Power consumption. McPAT version 1.0 is used to estimate static and dynamic power consumption [15] for a 22 nm technology. Power savings incurred by reconfiguration are modeled by running McPAT with the modified target parameters (Table 1). Running McPAT along with the performance simulation allows us to emulate the behavior of hardware energy counters at simulated time slices of 1 ms.

Adaptive Micro-Architecture. To keep all the cores active even at stringent power budgets, we incorporate core micro-architectural adaptation, LLC adaptation and DVFS adaptation simultaneously, thereby providing various operational points in our adaptive tiled many-core processor. The adaptive core/tile configuration is expressed as a tuple $[core, f_t, llc_t]$, denoting that the core is configured as *core*, running at frequency f_t and llc_t cache ways enabled for the given tile t (see also Table 1).

Workloads. We run a 64-thread workload using SPEC CPU2006 benchmarks; 29 programs in total, which along with all reference inputs leads to 55 benchmarks with 9 random. We replicate the workload by 2× and 4× for the 128-core and 256-core setups, respectively. We run the simula-

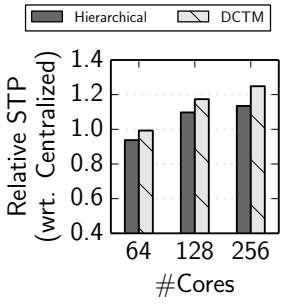

Figure 3: STP (normalized to *Centralized*) for *Hierarchical* and *DCTM* at 60% power budget vs. core count.

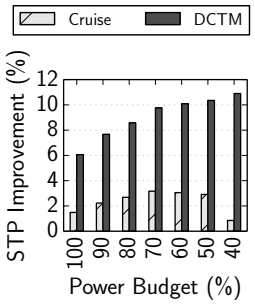

Figure 4: STP improvement (percentage) for *DCTM* and *Cruise* over *Hierarchical* for the 256-core setup.

tion for 200 ms to keep total simulation time within feasible limits. When a benchmark completes before this time, it is restarted on the same core. We quantify weighted speedup [19] or system throughput (STP) [9] which quantifies the aggregate throughput achieved by all cores in the system.

6. EVALUATION

We now evaluate DCTM on our power-constrained tiled many-core architecture. Unless mentioned otherwise, results are obtained using fine-grained hardware adaptation at 1 ms intervals, while thread migration is performed at 20 ms intervals. Each experiment fixes the available power budget to a fraction of the chip's nominal power consumption (see TDP in Table 2). We quantify performance in terms of system throughput (STP), which includes power management overhead.

6.1 Hierarchical vs. Centralized Power Management

We first evaluate the scalability of two-tier hierarchical power management versus a centralized approach. We consider the following power management policies: (i) *Centralized* — centralized power management with per-core DVFS; (ii) *Hierarchical* — our two-tier hierarchical power manager, with random mapping, and per-tile DVFS; and (iii) *DCTM* — our two-tier hierarchical power manager with DVFS and LLC aware thread migration.

Figure 3 quantifies relative STP (normalized to the *Centralized* approach) for the SPEC CPU workload as a function of core count at a 60% power budget. The *Centralized* approach is quite effective at 64 cores. The overhead of the centralized power manager is limited, and the ability to exploit per-core DVFS yields a performance benefit over the *Hierarchical* approach with per-tile DVFS. At larger core counts however, the overhead of the centralized power manager is not offset by the benefit from per-core DVFS, yielding a performance benefit for the *Hierarchical* approach (by 7% on average). The results also show that being able to migrate threads such that compatible threads co-execute per tile, as done using *DCTM*, yields a substantial performance benefit over random thread assignment with *Hierarchical*.

6.2 Two-Tier Approach: Performance vs. Power Budget

The application's sensitivity to DVFS could provide better performance than just considering LLC sensitivity. To illustrate this, Figure 4 shows the STP improvement (as percentage) of a 256-core setup at different power budgets for *Cruise* and *DCTM*, relative to the *Hierarchical* performance. Both *Cruise* and *DCTM* employ a two-tier hierarchical power manager. Figure 4 shows the STP improvement (as a percentage) for the 256-core setup at different power budgets for *Cruise* and *DCTM*, relative to *Hierarchical*. The bottomline is that *DCTM* outperforms *Hierarchical* by 10.1% on average. *DCTM* outperforms DVFS-agnostic *Cruise* by 6.7% on average. For average SPEC CPU, *DCTM* shows an increasing trend at increasingly smaller power budgets. The reason is that the workload includes a wide range of applications with varying characteristics, which can be efficiently exploited using both DVFS and LLC sensitivities.

7. CONCLUSION

An integrated and scalable many-core power management is clearly needed as we move towards increasingly tighter power budgets. In this work, we leverage a two-tier hierarchical power manager due to its low overhead and high scalability on a tiled many-core architecture with shared LLC and per-tile DVFS at fine-grain time slices. We leverage DVFS and cache-aware thread migration (DCTM) to ensure optimum per-tile co-scheduling of compatible threads at runtime over the two-tier hierarchical power manager. Based on our evaluations, we show that DCTM outperforms Cruise [11] by 6.7% on average for the multi-program SPEC CPU workload. Compared to a centralized power manager, DCTM improves performance by 14.1% on average while using 4× less on-chip voltage regulators.

References

[1] S. Borkar. Thousand core chips: A technology perspective. In *DAC*, 2007.
[2] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA*, 2001.
[3] E. A. Burton et al. FIVR – Fully integrated voltage regulators on 4th generation Intel® Core™ SoCs. In *APEC*, 2014.
[4] T. E. Carlson et al. An evaluation of high-level mechanistic core models. *ACM TACO*, 2014.
[5] Q. Deng et al. CoScale: Coordinating CPU and memory system DVFS in server systems. In *MICRO*, 2012.
[6] R. Dennard et al. Design of ion-implanted MOSFET's with very small physical dimensions. *ISSCC*, 1974.
[7] T. Ebi et al. TAPE: Thermal-aware agent-based power economy multi/many-core architectures. In *ICCAD*, 2009.
[8] H. Esmaeilzadeh et al. Dark silicon and the end of multicore scaling. In *ISCA*, 2011.
[9] S. Eyerman and L. Eeckhout. System-level performance metrics for multiprogram workloads. *IEEE MICRO*, 2008.
[10] X. Fan et al. Power provisioning for a warehouse-sized computer. In *ISCA*, 2007.
[11] A. Jaleel et al. Cruise: Cache replacement and utility-aware scheduling. In *ASPLOS*, 2012.
[12] S. S. Jha et al. Chrysso: An integrated power manager for constrained many-core processors. In *CF*, 2015.
[13] J. Leverich et al. Power management of datacenter workloads using per-core power gating. *CAL*, 2009.
[14] J. Li and J. F. Martinez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *HPCA*, 2006.
[15] S. Li et al. McPAT 1.0: An integrated power, area, and timing modeling framework for multicore architectures. *HP Labs*, 2009.
[16] K. Meng et al. Multi-optimization power management for chip multiprocessors. In *PACT*, 2008.
[17] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 1965.
[18] P. Petrica et al. Flicker: A dynamically adaptive architecture for power limited multicore systems. In *ISCA*, 2013.
[19] A. Snavely and D. M. Tullsen. Symbiotic jobscheduling for simultaneous multithreading processor. In *ASPLOS*, 2000.