Improving Execution Time of Parallel Programs on Large Scale Chip Multiprocessors with Constant Average Power Processing

Kramer Straube*, Christopher Nitta†, Raj Amirtharajah *,Matthew Farrens†, Venkatesh Akella*
*Department of Electrical & Computer Engineering, University of California, Davis
†Department of Computer Science, University of California, Davis
kkstraube@ucdavis.edu, cjnitta@ucdavis.edu, farrens@cs.ucdavis.edu, akella@ucdavis.edu

***Abstract*—In this paper we propose a microarchitectural technique called Constant Average Power Processing (CAPP) that reduces the execution time of parallel programs by dynamically detecting the power slack at runtime and directing it to specific core(s) that are the bottleneck at any given time. The key insight of this work is that by sensing the current, communicating it to the global controller and adjusting the cores' frequencies, it is possible to maintain a constant power level in a distributed and scalable manner. We evaluate the potential benefits and scalability of the proposed technique on a set of synthetic benchmarks and compare the results with related work such as Running Average Power Limit (RAPL).**

I. Introduction

In CMOS technology today there is a significant gap between the peak clock frequency and the nominal clock frequency of a core. This gap gets wider as the number of cores on a die increases, because the fixed power input has to be shared by a larger number of cores - this means the cores themselves have to be operated at a lower nominal frequency. Though a core cannot sustain higher frequencies for all possible instruction mixes, certain power headroom can be used to provide elevated frequencies. This is the approach used by Intel's Running Average Power Limit (RAPL), as described in [15] and [3]. Here, an on-die microcontroller and associated firmware called the Package Control Unit (PCU) estimates the power consumption of each core based on the instruction mix and uses that information to boost the clock frequency of the cores up to the power limit. RAPL controls the hardware through firmware which results in adaptation interval anywhere from tens of microseconds to milliseconds which misses some fine grained load imbalances. Furthermore, the centralized PCU used in RAPL limits the scalability of the approach, especially when the number of cores is large.

The objective of the work presented here is to overcome these drawbacks and develop a power management architecture and implementation strategy that (a) operates at a time scale on the order of hundreds of nanoseconds due to a fully hardware-based implementation, and (b) features a distributed implementation in order to improve scalability.

We call this proposed power management method Constant Average Power Processing (CAPP), which maximizes the *performance* of a chipscale multiprocessor by measuring the current draw from the power rails to determine the available slack (the difference between the power target and the actual power draw). This slack is used when calculating the appropriate voltage to put on the global voltage rail - the more (or less) slack there is, the higher (or lower) the voltage will be. This voltage is read by a local controller at each core, which can employ a variety of local metrics such as IPC, queue sizes, local temperature, etc. to determine the per core voltage (and frequency) to use. This value can be higher than the nominal voltage if the controller determines that there is sufficient slack and there is work to be done, or it may be lower if the controller decides the local processor has little or nothing to do.

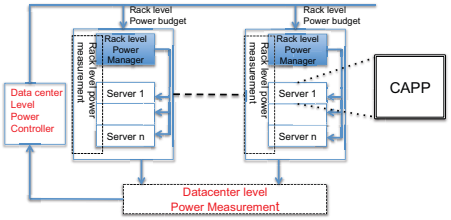

Fig. 1: High Level Overview of Multi-tier Power Limiting and Power Shifting Scheme in a Datacenter.

CAPP is a pure hardware-based approach with a control interval of hundreds of nanoseconds to allow the detection of fine grain load imbalances and dynamically divert power from cores that are stalled to cores that are active. The goal of CAPP is not to *minimize power*, but rather to *maximize performance* by constantly consuming the target amount of power. In CAPP there is an explicit attempt not to leave *any* power unused from the power budget, if there is an opportunity to use it to improve the performance. In this sense, CAPP acts as a faster hardware-based decentralized version of RAPL.

We begin by describing the high level architecture and principle of operation of CAPP in Section II and our evaluation methodology in Section III. Next, we evaluate the performance of CAPP on a set of synthetic benchmarks to verify that it works as intended, as well as the potential benefits on future applications. Finally, we close the paper with related work, and conclusions and some directions for future work.

II. CAPP Framework Details

Datacenters or warehouse-scale computers [2] are emerging as an important class of computers that require high-

1063-6404/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCD.2017.113

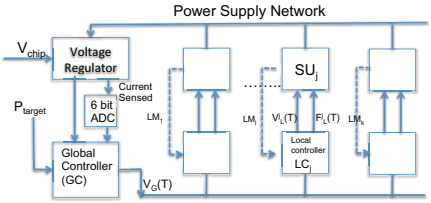

Fig. 2: CAPP: High Level Architecture

performance computing with a constant power limit. We consider a multi-tier power provisioning and management scheme in a typical datacenter as shown in Figure 1. The main datacenter-level power controller monitors the power usage and sets a power budget for each rack (a cluster of computers). The rack/cluster-level power manager sets the power target (P_{TARGET}) for a computing device in the server. As stated previously, the goal of CAPP is to deliver the **maximum** performance from each server for the given power target.

CAPP works by measuring the server's global voltage rail current draw $I(t)$ (i.e., the total current drawn by all of the computing elements) through a Voltage Regulator (VR) integrated with current sensing such as [1]. The current sensing circuitry detects and amplifies the sense voltage and provides that value to the CAPP global voltage controller, which uses that information to calculate the ideal global supply voltage that will maintain the target power level.

There are 3 main components to CAPP - the current sensing circuitry, the global controller (GC), and the local controller (LC_j) at each system unit (SU_j). The local controller can be a passive passthrough (one level controller CAPP), or it can use a variety of metrics such as IPC (two level controller CAPP). A system unit can be a core, a cluster of cores, or a special function unit such as a GPU or accelerator - we will focus on a homogeneous chip-scale multiprocessor where the system unit is a processor core.

Every power control epoch (T), the following set of closed loop actions occur: Changes in activity in system unit SU_j cause a change in the current draw $I(t)$ sensed by the VR and detected by the GC, which results in the GC setting the target voltage ($V_G(T)$) for the next epoch. The local controllers use a combination of ($V_G(T)$), their own local metric vector (LM_j), the settings of some runtime/programmer-visible registers, and a set of metrics such as the IPC to set their local voltage ($V_j(T)$) and frequency ($F_j(T)$).

One of the main benefits of CAPP is that it is not a one-size-fits-all scheme - each local controller can choose its own metrics and make local decisions based on its recent behavior or workload projections.

a) Assumptions and Implementation Issues: The duration of the epoch (T) is chosen so that the system is stable, i.e. all voltages have a chance to settle to their new values. Hence T depends on the actual resistance, capacitance, and inductance of the power supply network. As explained in Section III, we use a conservative value of 1 microsecond or higher in our other simulations. The proposed decentralized voltage controlled system makes the system units *asynchronous* with respect to each other in terms of clocking, which could result

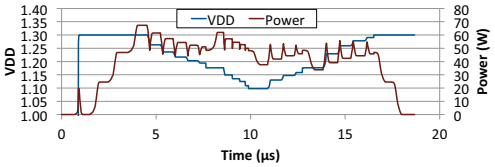

Fig. 3: Simulated supply voltage (blue) and power (red).

in timing violations if not properly addressed. We make two reasonable assumptions to avoid these problems - first, we assume that there is a local voltage guardband that ensures correct operation, and second we assume an *adaptive* clock scheme [10]. Adaptive clocking uses a local oscillator, such as an inverter ring, that operates at the same voltage as the rest of the system unit. This oscillator provides the clock signals used throughout the system unit - thus, the clock frequency will slow down when the voltage drops ensuring no timing errors occur.

b) Proof of Concept - Hardware Verification: In order to evaluate a CAPP implementation using a realistic global voltage controller cycle time (∼300 ns) we performed a Cadence Spectre simulation with high-current Verilog-A "core" models and a full implementation of the global controller. Figure 3 shows the supply voltage changes over the course of the simulation. Each "core" turns on in sequence and then turns off in sequence. Drops (and subsequent rises) in the supply voltage are seen for each of the "core" activity changes. These drops take place over several steps due to the choices of the Proportional Integral Derivative (PID) coefficient used within the global controller. Figure 3 shows the power during this simulation. Despite moderate spikes at voltage transition times due to the charging of the capacitors in the power supply network, the controller enforces the power target of 50 watts when possible. It exceeds the power target for short durations when the "cores" turn on, but quickly readjusts to match the power limit in the steady state (within an acceptable error margin). Overall, this simulation demonstrates that the hardware of CAPP functions as expected. The global voltage controller raises and lowers the global voltage in response to changes in the activity of the "cores", and these voltage changes keep the total power near the power target.

In summary, CAPP can be viewed as a simple, scalable, and more flexible (because the local controller at each system unit can be customized to meet its unique requirements) framework to maximize performance given a power constraint.

III. Evaluation Methodology

In order to evaluate the performance impact of CAPP, we modified Sniper version 6.0 [6] so that we could do full closed-loop power simulations. Sniper is a multi-core simulator that uses instruction instrumentation and instruction intervals to accelerate simulation. Because Sniper does not execute an entire program in cycle level accurate mode, there is a maximum execution time variation of approximately 3%.

In order to evaluate the power consumed when using CAPP, we had to do closed-loop power simulations over periods as short as 300 nanoseconds. We extracted the power model for our target processor from McPAT version 1.3 [11] and

TABLE I: Breakdown of delays for CAPP transitions

Component	Transition time (ns)
Voltage Regulator	36-226
Sensing Circuitry	50-60
Controller	10-30
Power Supply Network	3-15
Total	**99-331**

embedded it within Sniper. These changes were verified using McPAT and match to within 2% (this is below the variability of the Sniper execution).

We created Python scripts to implement the local controllers. The CAPP Python script calls the Sniper embedded power model to calculate the chip-level power, which the global voltage controller model within the Python script uses to determine future behavior.

The local voltage controllers use thresholds specific to the particular metric in order to determine whether to increase, decrease or maintain a voltage ratio. This voltage ratio is multiplied by the global voltage and matched to the highest frequency that can be run at that voltage. For example, a local voltage controller on Hi-IPC would raise the ratio if the IPC is above 0.6 and would lower the ratio if the IPC is below 0.3. For this paper, we used IPC as the local control metric for the 2 level CAPP with the thresholds of 0.6 and 0.3.

We used a Nehalem model for the cores in the Sniper simulations. The frequency of the cores can range from 2 GHz to 800 MHz at voltages from 1.2 volts to 0.8 volts (respectively). The configuration details were taken from the pre-existing configuration files within Sniper. Based on early simulation feedback, we allow the maximum global voltage to go up to 1.5 Volts - this allows the global voltage controller to push the frequencies of the cores upward (despite their ratios) when additional power is available.

In order to calculate a reference case for CAPP, we used a fixed frequency configuration of the same Sniper Nehalem-based system. We selected a frequency of 1.5 GHz, which was in the middle of the core's possible frequencies, to provide a reasonable margin for increasing frequency. Then, we measured the maximum power used by the fixed frequency system for a range or workloads. We also ran simulations using the maximum allowable frequency for the Nehalem cores, according to the processor model provided with Sniper (2 GHz). Running at this speed highlights the maximum possible performance that could be achieved on a particular workload.

Once the baseline and ceiling configurations were selected, we defined the parameters for CAPP, starting with the global voltage controller cycle time. Based on both individual component simulations and references from the literature, we calculated the roundtrip time of a CAPP update to the global voltage passing through the entire system (the range of possible times are shown in Table I). The voltage regulator delay is derived from [5], while the operational amplifier delay was measured using Cadence Spectre and the ADC delay was obtained from Murmann's ADC survey spreadsheet [13]. The controller delay was approximated by implementing similar logic and evaluating the delay using Spectre.

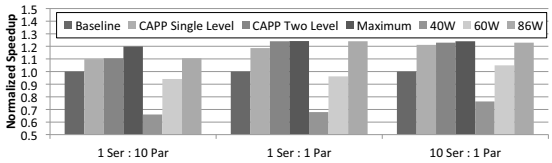

Fig. 4: Normalized execution time of CAPP, the maximum fixed frequency possible, and varying power vs. varying sequential to parallel ratios.

We calculated a range of 99 to 331 nanoseconds for the CAPP control cycle time, so we used a very conservative control cycle time of 1 microsecond for our performance evaluations. This ensures that the performance evaluation of CAPP reflects a realistic implementation by using a pessimistic control cycle time.

IV. Performance Analysis of CAPP

One of the main motivations for CAPP is to detect and overcome serial bottlenecks continuously to reduce the execution time and improve scalability of parallel programs. Serial sections are a problem, because according to Amdahl's law Equation, Speedup = 1 /(X + (1-X)/N) where X is the time spent on the serial portion that cannot be sped up, and N is the number of cores. In order to be able to evaluate CAPP's capability we need a controlled workload, where we can vary the duration and distribution of critical sections. Thus, we constructed a parameterized synthetic benchmark with a set of parameters to control different aspects of the execution, such as the length of the critical and parallel sections.

We wrote the synthetic benchmark in C++. The benchmark has each core loop through a small kernel a set number of times, separated by locks. The kernel behavior contains integer operations, memory operations and a conditional branch.

When modeling a critical section, all cores run the kernel for a set number of loops. Then, the lock begins and the first core to get the lock completes a the kernel workload while all other cores stop. Each other thread completes the serial section under the lock until they are all done.

Figure 4 shows the execution time of CAPP using single level and two level control strategies along side fixed maximum frequency runs on a synthetic parallel workload with a multiprocessor size of 8 cores. The execution times have been normalized to the base fixed frequency 1.5GHz cores. The results are grouped by the ratio of sequential to parallel code that exists in the synthetic benchmark. The power target of the CAPP systems is that of the peak power utilized by the fixed 1.5GHz system. As one might expect, as the sequential code increases the performance shifts toward that of the maximum obtainable (this is due to the fact that there is power slack in the system). The other point to note is that while even the simple strategy improves performace, the two level strategy consistently performs as well or better than the simple single level strategy. This implies that local knowledge can help further improve the global performance on a particular workload.

Figure 4 also shows normalized execution time for an 8 core CAPP with varing degrees of sequential and parallel time by

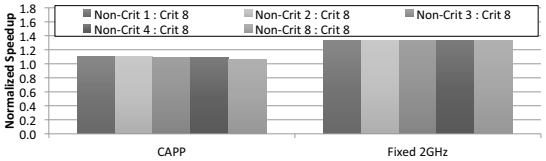

Fig. 5: Normalized execution time of CAPP and MAX for varying balance of non-critical to critical work.

power target. As one would expect, the performance increases with both higher sequential time and also with a higher power target. What is interesting is that CAPP is capable of about the same performance as the baseline with a power target of only 60W. The baseline fixed frequency utilizes a peak of 86W, but CAPP can obtain similar performance with ∼70% of the peak power.

Figure 5 shows the balance of non-critical to critical work varying on a synthetic benchmark. This represents an imbalanced parallel workload, and demonstrates that as the work becomes more imbalanced the relative performance of CAPP improves. (The Fixed 2GHz values are maximum values, and are not obtainable due to excess power consumption.)

V. Related Work

Power management in processors has been an active area of research for more than a decade. See [16] for a detailed survey of power management schemes, especially DVFS and its variants. As mentioned before, classic DVFS is different from what is proposed here since the goal of classic DVFS is to minimize power consumption while satisfying a performance constraint, while the goal of CAPP is maximizing performance while staying with an average power budget. However, it is important to note that the local controllers in CAPP can choose to minimize power if they so desire (for example, when they have no work to do). They don't have to run as fast as the voltage set by the global controller. Adrenaline [7], which attempts to reduce the tail latency of Memcached queries by voltage boosting, Rubik [9] which does fine grain voltage scaling and boosting to reduce variability in latency in datacentric workloads, both seek to maximize performance through voltage changes in specific alternate scenarios. The use of voltage/frequency islands (VFIs) is proposed in [12], [14], which break a multiprocessor chip up into various independent domains. These VFIs are controlled to minimize the energy of the total system while maintaining a certain quality of service (QoS) standard. Juang et al minimize the energy-delay product (EDP) of a chip multiprocessor design by reducing the power and slightly increasing the runtime [8]. A coordinated and stable scheme for controlling DVFS results in reduced frequencies without significant loss of performance. Ellsworth et al use dynamic scheduling to enforce a system-level power limit in an over-provisioned data center. [4]

VI. Conclusions and Future Work

The technique proposed here is purely hardware. In the future we plan to extend this approach to allow the OS to guide the local controllers, to set their voltage and frequency based on execution history of applications or user-hints. In this paper we selected a the IPC metric to guide the local controller in order to demonstrate the feasibility of the approach. However, there may be other metrics that are more appropriate for detecting fine grain critical sections.We intend to explore the design space of these control strategies in the future.

References

[1] "Dual channel pwm controller with integrated driver for imvp8 cpu core power supply," http://www.richtek.com/assets/product_file/RT3606BC/DS3606BC-00.pdf, accessed: 2016-09-30.

[2] L. A. Barroso, J. Clidaras, and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition*, ser. Synthesis Lectures on Computer Architecture, M. D. Hill, Ed. Morgan & Claypool Publishers, August 2013.

[3] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "Rapl: memory power estimation and capping," in *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*. IEEE, 2010, pp. 189–194.

[4] D. A. Ellsworth, A. D. Malony, B. Rountree, and M. Schulz, "Dynamic power sharing for higher job throughput," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015, p. 80.

[5] W. Godycki, C. Torng, I. Bukreyev, A. Apsel, and C. Batten, "Enabling realistic fine-grain voltage scaling with reconfigurable power distribution networks," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2014, pp. 381–393.

[6] W. Heirman, T. Carlson, and L. Eeckhout, "Sniper: Scalable and accurate parallel multi-core simulation," in *8th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems (ACACES-2012)*. High-Performance and Embedded Architecture and Compilation Network of Excellence (HiPEAC), 2012, pp. 91–94.

[7] C.-H. Hsu, Y. Zhang, M. A. Laurenzano, D. Meisner, T. Wenisch, J. Mars, L. Tang, and R. G. Dreslinski, "Adrenaline: Pinpointing and reining in tail queries with quick voltage boosting," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 2015, pp. 271–282.

[8] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, and D. W. Clark, "Coordinated, distributed, formal energy management of chip multiprocessors," in *ISLPED'05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005.* IEEE, 2005, pp. 127–130.

[9] H. Kasture, D. B. Bartolini, N. Beckmann, and D. Sanchez, "Rubik: Fast analytical power management for latency-critical systems," in *Proceedings of the 48th International Symposium on Microarchitecture*. ACM, 2015, pp. 598–610.

[10] B. Keller, "Opportunities for fine-grained adaptive voltage scaling to improve system-level energy efficiency," Master's thesis, EECS Department, University of California, Berkeley, Dec 2015.

[11] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2009, pp. 469–480.

[12] A. K. Mishra, S. Srikantaiah, M. Kandemir, and C. R. Das, "Cpm in cmps: Coordinated power management in chip-multiprocessors," in *2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2010, pp. 1–12.

[13] B. Murmann, "Adc performance survey 1997-2015," *http://www. stanford. edu/˜ murmann/adcsurvey. html.[1][2][6][7][3] This work Fs (MS/s)*, vol. 50, no. 100, p. 50, 2015.

[14] U. Y. Ogras, R. Marculescu, P. Choudhary, and D. Marculescu, "Voltage-frequency island partitioning for gals-based networks-on-chip," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 110–115.

[15] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, Mar. 2012.

[16] M. Själander, M. Martonosi, and S. Kaxiras, "Power-efficient computer architectures: Recent advances," *Synthesis Lectures on Computer Architecture*, vol. 9, no. 3, pp. 1–96, 2014.