# Improving Provisioned Power Efficiency in HPC Systems with GPU-CAPP

Kramer Straube*, Jason Lowe-Power †, Christopher Nitta†, Matthew Farrens†, Venkatesh Akella*
*Department of Electrical & Computer Engineering, University of California, Davis
†Department of Computer Science, University of California, Davis
Davis, USA
kkstraube@ucdavis.edu, jlowepower@ucdavis.edu, cjnitta@ucdavis.edu, farrens@cs.ucdavis.edu, akella@ucdavis.edu

***Abstract*—In this paper we propose a microarchitectural technique called GPU Constant Average Power Processing (GPU-CAPP) that improves the power utilization of power provisioning-limited systems by using provisioned power as much as possible to accelerate computation on parallel workloads. GPU-CAPP uses a flexible, decentralized control to ensure fast response times and the scalability required for increasingly parallel GPU designs. We use GPGPU-Sim and GPUWattch to simulate GPU-CAPP and evaluate its capabilities on a subset of the Rodinia benchmark suite. Overall, GPU-CAPP enables speedup by an average of 26% and 12% over equivalent fixed frequency systems at two power targets.**

***Keywords*-power capping; GPU; power management; performance maximization;**

## I. Introduction

With explosive growth in data in every discipline there has been an insatiable demand for high performance computing in both traditional scientific applications and large scale data analytics. GPUs have become an integral part of high performance nodes because of their ability to harness scalable data parallelism in a wide range of applications. However, power consumption continues to be a serious concern as we march towards exascale computing systems. As Barosso et. al [2] point out, managing power in high performance computing facility is challenging because of a variety of reasons - not only is the capital cost of provisioning power quite substantial but the operating costs of a high performance computing facility are also dominated by the cost of power. In addition, researchers have found that computation is quite **bursty** in nature - periods of high activity followed by periods of inactivity [13]. Also, power is not proportional to the actual activity in the processors because even if a processor is idle (i.e., not turned off) its power consumption could be quite significant. Power capping and power shifting [17], [23], [3], where power is managed as a dynamic resource that is allocated to different clusters of compute nodes (racks) based on the characteristics of the workload and projected demand, so that the servers are operating at optimal power efficiency and minimizing the power cost is an important technique to address the power management challenges in a high performance computing facility.

Broadly speaking the objective of this work is to extend power capping to the chip-level, specifically to GPUs and to maximize the performance that can be achieved with a given power budget. The goal is for the average power to be as close to the allowable peak power as possible. One way to accomplish this is with over provisioning as shown by Pakti et. al [20]. While this will successfully use some of the excess available provisioned power, it will also cost significantly more due to the increased hardware costs. Provisioning power is a significant cost at all levels from the datacenter down to a single chip. Thus, our aim is to improve the utilization of this provisioned power at the level of a single GPU.

The issue of using less than the available provisioned power is important in the case of GPUs. Figure 1 shows the total power over time on a simulated GTX 480 running the LU Decomposition benchmark from the Rodinia suite running at the nominal frequency of 700 MHz. The peak power of a GPU workload exceeds the average power by almost 4X! If the supercomputer was provisioned based on the aggregate peak power across all GPUs, there would be a large amount of wasted power provisioning. To simplify the description of this excess provisioned power over time, we call the percentage of the available power over time that the system uses the *Provisioned Power Efficiency* (PPE) as shown below. Provisioned Power Efficiency is the average power divided by the maximum provisioned available power.

$$PPE = \frac{AveragePower}{SystemProvisionedPower} \tag{1}$$

Having a low provisioned power efficiency leads directly to lost money in designing the high performance computing system because it needs to accommodate the maximum power draw. Figure 2 shows the provisioned power efficiency for our baseline GPU running at its peak frequency of 700 MHz. This figure shows that on average most workloads use less than 50% of the provisioned power. The reason for this inefficiency is that the system must be configured for the worst of the worst case across all time for all workloads.

Our goal is to use very fine-grained dynamic frequency and voltage scaling (DVFS) at the GPU chip level to smooth the curve in Figure 1 and bring the average power and peak power together increasing the PPE. At the chip level level, more power management techniques are available to modify the peak and average power draw. Additionally, the

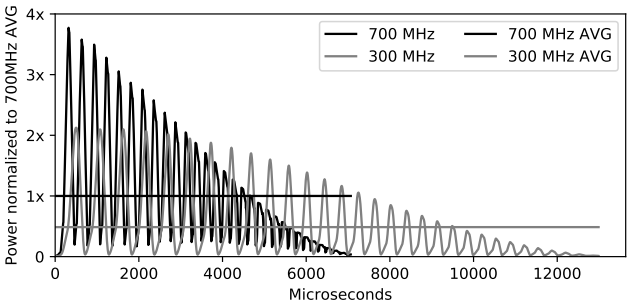



Figure 1: Nomalized power over time of 300 MHz and 700 MHz GPU running LU Decomposition.

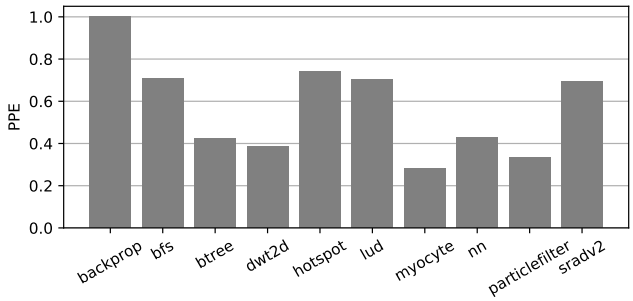



Figure 2: Optimal provisioned power utilization at 700 MHz across several rodinia benchmarks.

provisioned power at the chip level is also a significant portion of the cost of the chip due to limited specifications of the voltage regulator and the number of pins used for power supply. By increasing the chip-level PPE the entire datacenter's PPE will also increase since designers can reduce the required guardbands.

Intel's technique called Running Average Power Limit (RAPL) attempts to keep the system at a power limit over a given period of time but has several key issues that prevent its usage in GPUs [7]. RAPL uses a centralized scheme where subcomponent (such as GPU streaming multiprocessors) metrics are centralized in a single controller and all decisions regarding power management are made there. As the number of subcomponents scales, RAPL becomes untenable due to the amount of information that must reach the central controller from all of the subcomponents. The only way for all of that information to reach the controller is either through some kind of bus or through separate wires. A bus would quickly become congested with all of the simultaneous access requests from many streaming multiprocessors (SMs), and using dedicated wires would require significant amounts of area, power and specialized routing. Alternatively, system-level techniques operating in software could control the system's power draw. These techniques cannot react quickly enough (˜100$\mu$s) to fast spikes in power and can violate the provisioned power limit as seen in Figure 1.

To improve the provisioned power efficiency, we propose GPU Constant Average Power Processing (GPU-CAPP). GPU-CAPP overcomes the bandwidth and latency limitations of RAPL and other DVFS implementations by using the power delivery network for global communication. GPU-CAPP controls the overall power usage of the chip through a two-level hardware implementation. The first level occurs at the on-chip voltage regulator and senses the current power usage and controls the global voltage to maximize the provisioned power efficiency. The second level occurs at each SM and uses the global voltage along with local metrics to assign voltage and frequency settings to the SM to maximize its power efficiency.

We implemented GPU-CAPP using a validated GPU power model (GPUWattch [18]) and evaluated a large design space for potential local controllers. We found the best local controller should use a dynamic active warp target based on the current power usage to control the local voltage and frequency of each GPU SM. We evaluated GPU-CAPP across a subset of the Rodinia benchmark suite at two different power targets. We find that if the system is provisioned for a low power target (23 W), the speedup gained by GPU-CAPP over a similarly provisioned fixed frequency system is 26.5%. If the system is provisioned for a mid-range power target (43 W), the average speedup of GPU-CAPP is 12%. GPU-CAPP improves the PPE from 61% to 88.8% at 23 W and from 54.4% to 65.6% at 43 W. These results show that GPU-CAPP allows the user to get a higher performance for the same cost of system provisioning.

## II. GPU-CAPP Framework Details

GPU Constant Average Power Processing (GPU-CAPP) uses a two-level decentralized control schema to optimize the performance under a power target. We adopt the design from Straube et. al [26] to work for GPUs instead of CPUs and extend its behavior for several GPU-specific features. We cover the design here to clarify the functionality in the GPU setting. The major differences that between the CPU domain and the GPU domain are threefold. First, the number of streaming multiprocessors (SMs) in a GPU far outnumber the number of cores in a CPU. The additional SMs can utilize local control and will allow for even larger swings between the average and maximum power. The increased number of control nodes (aka SMs or cores) highlights the scalability requirements for the GPU setting, exacerbating any scalability issues with various designs. Second, the program behaviors between CPU programs and GPU programs are different due to the focus on data parallel workloads for the GPU to extract maximum benefit. These different workloads will change the power and performance behavior of the system. Finally, the interaction of the observable metrics at each SM with the future power behavior is

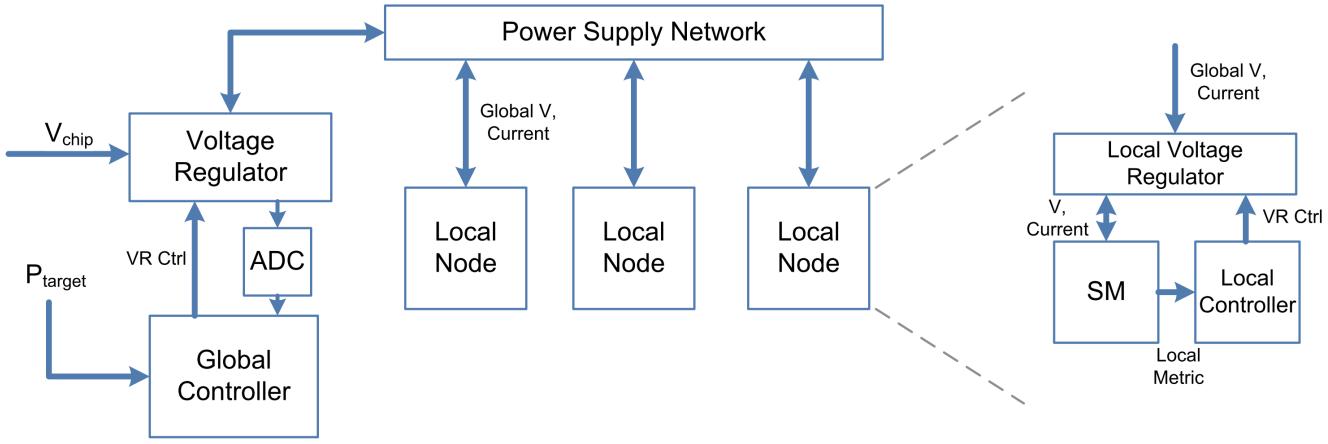



Figure 3: CAPP: High Level Architecture

different from its CPU counterpart due to the pipeline and program designs for the GPU domain. These factors make the potential impact of a GPU implementation larger than that of a CPU implementation.

The key reason GPU-CAPP is more scalable is that all global communication occurs through the power supply network. The use of a local and global controller allow for per-SM and per-chip control behaviors respectively without affecting the scalability of the approach in the GPU implementation. The global controller maintains the power target of the entire chip. The local controller optimizes the power usage at each SM to reduce inefficient power usage. Figure 3 shows the high-level design of GPU-CAPP. In this paper, we adapt the original algorithm by replacing the system units (SUs) with SMs, modifying the global controller and changing the local controllers for interfacing with the SMs.

The general flow of GPU-CAPP begins with an activity change at one or more SMs. This causes an increased current draw which is propagated back to the global Voltage Regulator (VR) through the power supply network. The global VR senses this current and the global controller receives the new current value with the global voltage. The global controller calculates the new global voltage and sets the global VR to output this new global voltage. The global VR settles to the new global voltage. The local VRs and local controllers get the new global voltage, convert that voltage to a local voltage and output the new local voltage and local frequency. The SM is set to the new local voltage and frequency, altering the power to better meet the power target.

### A. *Global Controller*

The global controller uses the current measurement on the global voltage supply rail along with the last applied voltage obtained from the VR with integrated current sensing such as [24]. The global controller includes and analog to digital converter, control logic, and the defined power target (stored in a register). The control logic within the global controller converts the input current and voltage measurements to a power measurement. This power measurement is converted to a power error with the power target. A Proportional-Integral-Derivative (PID) controller uses the power error to determine the new global voltage. The controller was tuned manually with a single benchmark and then subsequently validated across the entire benchmark suite. The global controller feeds back the new global voltage setting to the voltage regulator. This method of control is reactive to the power changes as they occur and will exceed the power target for very short periods of time (<20 microseconds). However, the global controller reacts quickly to reduce the voltage to bring the power back to the power target. These short times above the power target are allowable because the capacitance built into the system will filter out the extra power draw for short periods of time.

The implementation of the PID controller allows for several design decisions. We tuned the proportional and integral coefficient using a step response function and a single benchmark. Through successive runs, we first tuned the proportional coefficient to its maximum stable value by increasing it until the output became unstable. Then, it was backed off and the integral component was increased until the power constraint was within a 5% guardband. This step insured that the integral component properly controls the steady state error. Since the system current draw as a function of the voltage changes over time, we also used a single benchmark (hotspot) to further tune the PID controller coefficients. By adding to the integral coefficient, these activity oscillations affected the global voltage less. We found that the derivative component was not needed for this application. We added a feed-forward component to the PID controller to center the PID behavior within the desired range of voltages. The integral error was clipped to

prevent a large amount of integral error from building up and unduly influencing the system over a long period of time. The PID tuning is specific to the system design. A new system architecture would require the PID to be re-tuned. Process variations could be handled through different PID coefficients or guardbands and binning the resulting silicon into various bins satisfying the power characteristics. The PID controller would still control the system sufficiently with expected process variations but may be slightly more aggressive or conservative compared to the median system unit.

On a production device, PID tuning would follow an algorithmic procedure. First, the proportional constant would be increased until the voltage response was sufficiently fast (with some extra guardband). Then, the integral constant would be increased until the worst case power virus did not violate the power target.

### *B. Local Controller*

The local controller interfaces with the per-SM voltage regulator (or DC-DC converter) to control the local voltage and frequency. The local controller improves the power efficiency of the SM by controlling the voltage and frequency of the SM based on a chosen metric. When the SM is not at full utilization, the local controller can drop the local voltage, slowing down the SM and freeing power for other more critical SMs. To accomplish this task, the local controller uses a small amount of control logic, an interface with the SM and an interface with the local VR. The local controller uses a metric such as instructions per clock (IPC) from the SM to determine how much of the global voltage to use. For a GPU, there are additional metrics that are available at each SM for the local controller such as the number of active warps. Thermal effects are not modeled in this paper because we assume that the system is limited by the provisioned power not thermal effects but could be handled by the local controller.

We evaluated several possible local controller designs based on three possible metrics. The first local controller design is a threshold-based design using IPC as the primary metric. This means that when the IPC exceeds the upper threshold of 0.8, the ratio of the global voltage used by the local controller to define the local voltage would increase by 0.05 (to a maximum of 1.0). When the IPC fell below the lower threshold of 0.6, the ratio would decrease by 0.05 (to a minimum of 0.7). These values were determined experimentally based on the fixed frequency runs of a subset of the benchmarks. The local voltage equals the global voltage (controlled by the global controller) multiplied by the local controller ratio. The voltage determines the allowable frequency that the SM can run. This kind of controller is a static local controller using IPC. We chose to examine IPC as a control metric because high IPC indicates that the SM is bottlenecked by the compute units, not uncontrolled units such as memory. Frequency increases in the case of high IPC often translate to additional completed instructions and faster overall execution.

We also implemented a local controller using the number of active warps. The threshold-based design is similar but uses different upper and lower threshold values. A high number of active warps allows the SM to time multiplex the warps to prevent any single one from being bottlenecked by memory. Therefore, frequency increases often result in faster program execution. A lower number of active warps may not be able to fully utilize the SM so the frequency can be lowered with reduced performance impact.

The last static local controller we implemented was based on the number of inactive warps. While this initially seems unintuitive based on the explanation for the active warp-based local controller, the number of inactive warps can indicate that the SM is the lagging SM of the parallel execution due to increased memory time or other outside latencies. The increased memory time, for example, may have resulted from cache misses, coherence traffic or page faults. To speed up the execution of the entire program, the lagging SM is the bottleneck and needs to be accelerated. The competing factors between the active and inactive warp controllers can manifest differently depending on the specific benchmark behavior.

We implemented a dynamic controller for each of these static controllers that attempts to get the global voltage to equal to a desired value. Each of these controllers use a static-based design that has fixed thresholds but that limits the adaptability of the local controller to various benchmark behaviors. By allowing the target value to adjust, the local controller thresholds dynamically adjust for the different behaviors of the various benchmarks. To determine whether to alter the local voltage via the local controller voltage ratio, the local controller measures if the metric exceeds the desired target by a fixed percentage (5% in this implementation). Attempting to get the global voltage to a target value allows the controller to adapt to the current power. When the voltage is above the target voltage, the metric target will increase to be further discerning between the SMs. This method pushes the power to the SMs that have the highest relative metric values without global communication. These local controllers were created for the IPC, active warp and inactive warp metrics.

### *C. Effects of the Power Supply Network*

Central to the way that this scheme functions are the characteristics of the power supply network. Based on the resistance, capacitance, and inductance of the system, the time for the global voltage to propagate from global VR to the per-SM VRs and the time for the current changes to propagate from the SMs to the global VR can change. These times are critical to determining how fast the global controller can operate and change the global voltage.

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** |
| **GPU-CAPP Control Period** | **1,000** |

The frequency of the GPU-CAPP control code execution is determined by the hardware details shown in Table I. The voltage regulator delays originate from the Raven voltage regulator implementation [16]. We selected this VR for its quick response time to voltage changes. The sensing circuitry, power supply network and controller delays are determined from Cadence Spectre simulations. The power supply network delays are design specific and may change depending on the design. We used the model presented by Gupta et. al [9]. To ensure that these numbers are not too aggressive, we use a conservative control cycle time of 1 microsecond.

### D. Assumptions and Implementation Issues

Throughout this design, we make several assumptions about the system and the design of the nodes. The proposed design of GPU-CAPP is decentralized and global voltage control is independent of the control of the SMs themselves. Since the global voltage can change without warning the SMs, there is a risk of timing violations if the changes are not adequately addressed. There are two ways to design the system to prevent these issues. First, a voltage guardband can be applied to the system to ensure that the local controller has sufficient time to react as the global voltage drops. Second, the SMs can use *adaptive clocking* as described by Keller [15]. Adaptive clocking uses a local oscillator that operates at the local voltage and is used for clocking the SM to prevent timing violations. In our specific implementation, we assume the use of adaptive clocking but we provide the alternative strategy for future potential implementations.

## III. Evaluation Methodology

To evaluate the GPU implementation of GPU-CAPP, we used GPGPU-Sim. GPGPU-Sim simulates the functional capabilities of the parallel thread execution and the timing behavior of the compute portion of the GPU [1]. It uses a low-level virtual machine by Nvidia using a CUDA intermediate code format to functionally evaluate the program. The timing portion of GPGPU-Sim is a cycle-level simulator focusing on the SMs, caches, interconnect network, memory partition and graphics DRAM. Each of these components have one or more detailed models to capture detailed behaviors and model them appropriately. The main definition of our GPU target system is shown in Table II.

Table II: Details of GTX 480 Configuration

| Component | Detail |
|---|---|
| SMs | 15 |
| Cores per SM | 1 |
| L1 Cache Size | 16 kB |
| Shared Memory Size | 48 kB |
| L2 Cache Size | 768 kB |
| Maximum Frequency | 700 MHz |
| Minimum Frequency | 100 MHz |

While GPU-CAPP is a general approach to power capping in GPUs, we defined the specific implementation that we would use to evaluate the idea. The GTX480 architecture forms the base of the GPU because its power was characterized in detail in GPUWattch. To measure the power, we use the GPUWattch energy model that is integrated with GPGPU-Sim [18]. This model accounts for both dynamic and static power draws based on the activity factors and system configuration of the GPU.

From this setup, we modified the internal simulator code to support GPU-CAPP functionality. The GPU-CAPP control code runs together with the power model to properly update the voltage and frequency settings of each SM. The power model provides the total power and the simulator core provides the IPC metric information for each simulator cycle. We modified the GPUWattch code to support dynamic voltage and frequency scaling (DVFS).

For all results in this paper, we selected power targets to demonstrate the behavior at different areas of the frequency curve. The fixed frequency system uses the frequency that does not violate the power constraint across all benchmarks. Since the fixed frequency implementation provides a steady baseline against which all schemes can be compared, we consider this to be a fair comparison.

## IV. Performance Analysis of GPU-CAPP

To evaluate the performance of GPU-CAPP, we ran a subset of the Rodinia benchmark suite on the GTX480 with an expanded voltage and frequency set [5]. These benchmarks were chosen as a representative set of the entire suite to show the capabilities of GPU-CAPP on GPGPU applications.

For each benchmark, a power target was set to a preselected value. We collected baseline values of power and execution time for a range of frequencies from 100 MHz to 700 MHz. The top frequency option shows the best possible execution time that GPU-CAPP could reach if it was not bounded by power. We present the results for two different experiments. We assume the system is configured to run a set of workloads and the GPU-CAPP peak provisioned power is set to the predefined value. The maximum possible frequency that can be used as a baseline is the highest frequency which does not violate the power constraint for all benchmarks. We evaluate the speedup and provisioned

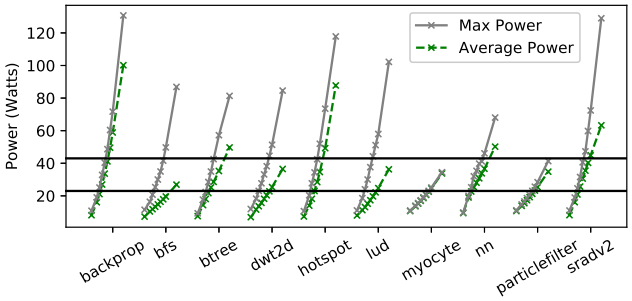



Figure 5: Average and maxiumum power for each fixed frequency (100 MHz–700 MHz at 50 MHz granularity).

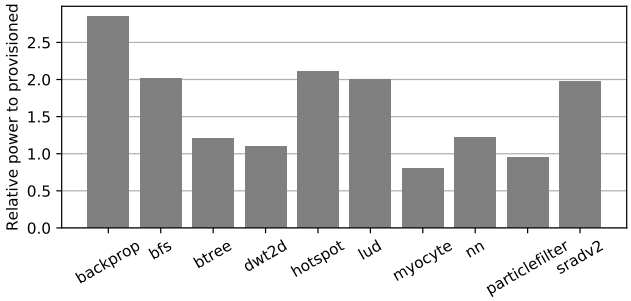



Figure 6: Normalized maximum power of 700 MHz relative to high power target (43 W).

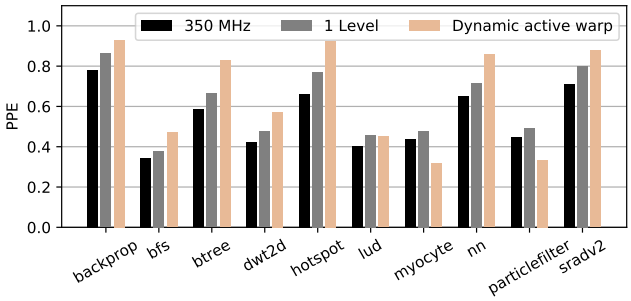



Figure 7: Provisioned power efficiency for the 43W power target.

power efficiency of a variety of GPU-CAPP implementations at a single power target. We also evaluate the impact of the 1 level GPU-CAPP implementation at both a lower power configuration to understand how GPU-CAPP behaves at various power levels. These power levels correlate to two different cost points in provisioning these GPUs and the relative speedup shows the improvement over fixed frequency. We also compare the speedup achieved relative to the 700 MHz performance to see how we compare with the maximum performance despite the additional power constraint. The results for each of these cases provide insights into the capabilities of GPU-CAPP.

### *A. GPU-CAPP Controller Evaluation Results*

The high power configuration of the system uses a power limit of 43 Watts. Under this constraint, the maximum possible frequency for the fixed frequency system is 350 MHz. We tested the 1 Level GPU-CAPP implementation and all of the 2 Level GPU-CAPP implementations described in Section III.

Figure 4 shows the speedups achieved by the various GPU-CAPP implementations over the set of Rodinia benchmarks when using the 23W power target. The overall speedup available is less due to the reduced frequency gap between the baseline system (350 MHz) and the maximum system frequency (700 MHz). Figure 5 shows the power characteristics of the fixed frequency runs. The curves represent the average and maximum power for each frequency point across the benchmark suite. The different power behavior of the benchmarks with respect to frequency illustrates how some benchmarks can gain more speedup under a power constraint than others. The larger the gap between the average power and the maximum power, the more potential for speedup under a power limit because on average there is more excess allocated power to utilize. The two horizontal lines show the two power constraints that we evaluate in this paper. To determine the appropriate fixed frequency system for comparison, we took the fixed frequency setting that did not have a maximum power above the specified power for any benchmark. With this methodology, we determined that 350 MHz was the high power setting and 200 MHz was the lower power setting. The 1-Level GPU-CAPP implementation with no local controllers achieves an average 5.0% speedup. The local controllers can achieve further average speedups of up to 12%. The optimal local controller was the dynamic target active warp local controller. These speedups are less than the low power results because of the reduced available speedup. The best local controller captures 85.2% of the speedup at 700 MHz. Figure 6 shows the maximum power at 700 MHz. These values exceed the power target so 700 MHz cannot be used and is not viable for this system but is is the highest possible frequency and shows the maximum unconstrained performance. GPU-CAPP only reduces the speedup by 14.8% relative to 700 MHz despite reducing the required power provisioning by 184%.

Figure 7 shows the provisioned power efficiency relative to the high power target for 350 MHz, 1-Level GPU-CAPP and 2-Level Warp GPU-CAPP. The 350 MHz provisioned power efficiency shows the base PPE of the system without a dynamic power management scheme. This varies across benchmarks as expected. 1-Level GPU-CAPP improves the average provisioned power efficiency from 54.4% at 350 MHz to 61.1%. The efficient local controllers generally improve the power efficiency in the high power case. For example, the dynamic target active warp local controller-based GPU-CAPP implementation achieved 65.6% provisioned

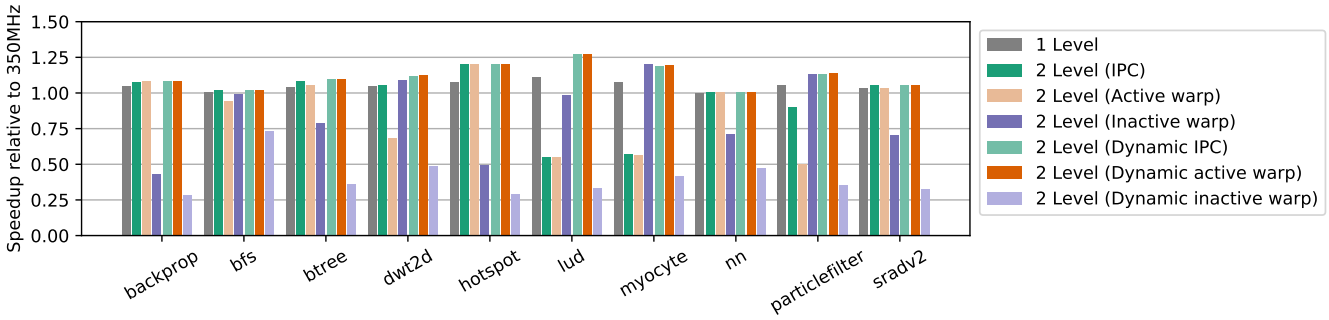
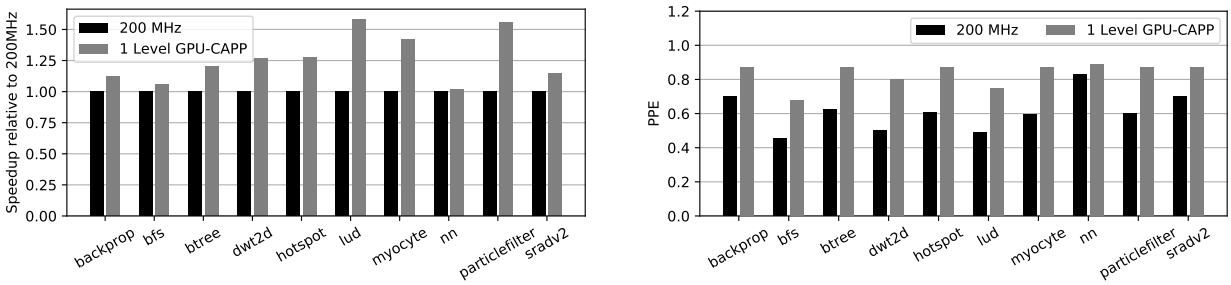



Figure 4: Speedup GPU-CAPP relative to 350MHz. "1 Level" referes to GPU-CAPP without a local controller.




(a) Normalized execution time relative to the 200MHz.

(b) Provisioned power efficiency relative to 23W power target.

Figure 9: Results with Low Power Target (23 W)

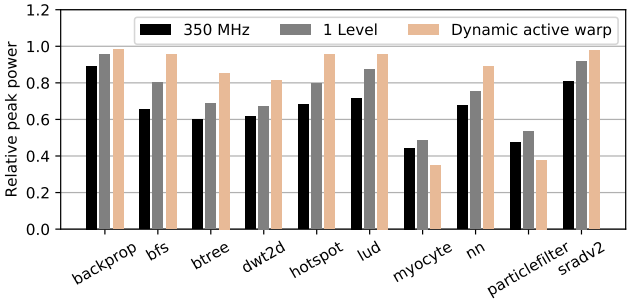



Figure 8: Normalized maximum power to high power target (43 W).

power efficiency. This shows how the best local controller achieves 7% more speedup and how GPU-CAPP leverages the additional available provisioned power to gain speedup despite the power constraint. The improved provisioned power efficiency in the local controller shows how it can add local control that improves efficiency. This behavior will not fit for all benchmarks. For example, particlefilter, myocyte and lud all are less efficient for 2-Level Warp GPU-CAPP compared to 1-Level GPU-CAPP. These cases identify current shortcomings of the local controller design. The behavior of the warps in these cases causes the local controller to reduce frequency when the system benefits from an increase in frequency without exceeding the power limit. Future work on the local controllers could target these cases to improve the average PPE.

Figure 8 shows the maximum provisioned power to determine whether the GPU-CAPP approaches violate the power constraint. 350 MHz has an average maximum power of 77.7% with a maximum of 98.7%. This verifies that 350 MHz is a valid frequency setting for the power constraint. These values also indicate that the frequency setting is close to the optimal infinite-granularity frequency setting for this power limit. 1-Level GPU-CAPP has an average maximum power of 74.8% showing that it does not violate the power constraint. The dynamic target active warp local controller-based implementation has an average maximum power of 81.1% with a maximum of 95.8%. Based on these results, the local controllers provide additional speedup and provisioned power efficiency without violating the power constraint at high power.

### *B. Experiments with Low Target Power*

The low power configuration of the system uses a power limit of 23 Watts. Under this constraint, the maximum possible frequency for the fixed frequency system is 200 MHz. We tested the 1 level GPU-CAPP implementation to understand how this design scales to other power levels. This lower power constaint shows a system with lower provisioned power and provisioning cost.

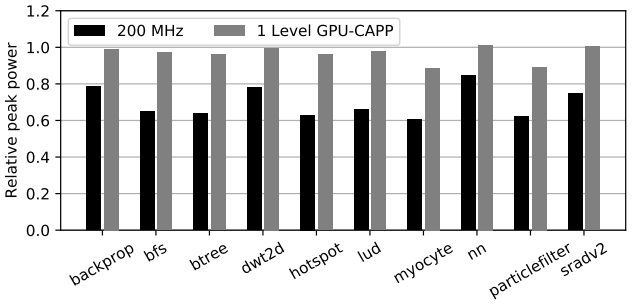



Figure 10: Normalized maximum power relative to low power target (23 W).

Figure 9a shows the speedup achieved by the GPU-CAPP implementation over the set of Rodinia benchmarks when using the 23W power target. The main factors that determine the speedup achieved by GPU-CAPP are the available speedup in the benchmark and the average power of the 200 MHz system. The more power gap there is between the power target and the 200 MHz system, the more power can be applied to raise the performance. The available speedup is the speedup that the benchmark can achieve with frequency scaling. The 1-Level GPU-CAPP implementation with no local controllers achieves an average 26.5% speedup. GPU-CAPP at 23 Watts cannot capture much of the 700 MHz performance due to the more strict power constraint. However, at this strict of a low power constraint, the comparison becomes less informative because the 700 MHz average power of some benchmarks exceeds 100 Watts, over 400% more than the provisioned power here.

The local controllers are not shown because they require additional design and tuning to meet the power constaint. At lower powers with the current local controller designs, the power fluctuations due to GPU activity compounded with the possible power impact due to changes at the local controller can cause current changes that the global controller cannot manage. At low power, we intend to design the local controllers to use smaller frequency steps to allow the global controller to maintain control despite both the activity and local controller behavior changing.

The provisioned power efficiency is shown in Figure 9b relative to the low power target. 1-Level GPU-CAPP improves the average provisioned power efficiency from 61% at 200 MHz to 88.8%. This improvement occurs because GPU-CAPP increases the frequency when the power is below the target to attempt to use all available power. This behavior is consistent with the provisioned power efficiency improvements seen at 43 Watts. Since there were no changes to the controller besides the power target, these results show that GPU-CAPP can be used to target multiple power levels without additional design or tuning.

Figure 10 shows maximum provisioned power to determine whether the GPU-CAPP approaches violate the power constraint. 1-Level GPU-CAPP has an average maximum power of 96.5% showing that it properly caps the power. This ensures that the different power target does not cause GPU-CAPP to fail the power constraint.

Overall, the low power configuration demonstrates the scalability of GPU-CAPP. Since there were no changes to the controller besides the power target, these results show that GPU-CAPP can be used to target multiple power levels without additional design or tuning. This also means that GPU-CAPP could be reconfigured to a new power target in a system if needed. The speedup and PPE of GPU-CAPP is better at lower power because there is a larger frequency gap between the fixed frequency system and the maximum operating frequency.

## V. Discussion

The results show that GPU-CAPP can speedup execution of several programs while maintaining the power cap defined by the amount of power provisioned for the system. The main difference between the benchmarks where GPU-CAPP gains significant speedup and those where it does not is the amount of provisioned power efficiency that the fixed frequency configuration attains. If there is a large gap between the average power and the maximum power, then GPU-CAPP can use the extra power during the more frequent low power times to accelerate computation.

The local controllers implemented in this paper have not been extensively optimized. More advanced and intelligent local voltage controllers could significantly improve performance, and will be investigated in future work. The local voltage controllers can be improved with either more intelligent hardware algorithms or, more intriguingly, it is straightforward to provide a mechanism for the OS to change the ratios being used at a local controller. Giving the OS and/or the user the ability to have some input regarding when a given SM should speed up or slow down has the potential to be significantly more effective, since the hardware itself has no way to know when it is entering a critical section but the programmer does. Software hints can also tell GPU-CAPP which local controller to use based on the expected behavior of the program.

For our implementation, we assume the use of a switched-capacitor voltage regulator. It is also possible to use a standard integrated buck voltage regulator, which would increase the control cycle time to $\sim$2 $\mu$s. The integrated voltage regulator technology has been demonstrated in products such as Intel's Haswell processor in 2013. In the absence of an integrated voltage regulator, the control cycle time would increase dramatically and require a new design.

In this paper we have assumed that the power constraint given to GPU-CAPP is lower than the TDP (thermal design power), so we do not take temperature and thermals into account. GPU-CAPP does not preclude local controllers

from taking local core temperature as an additional input to set its voltage during each epoch, however.

While this paper is about the chip level technique, GPU-CAPP provides benefits at the rack level and datacenter level. Since each chip is consuming a more consistent amount of power, the racks are also consuming a more consistent amount of power. Datacenter level algorithms can be added on top of GPU-CAPP and can also change the power target of each individual GPU-CAPP instance to use intelligence at the higher level to further improve execution time.

*Design Guidance*

What do these results mean to different stakeholders of a high performance computing system, namely, GPU architects, HPC operators, and programmers?

Our guidance to the GPU architect is to use GPU-CAPP and to expand the available frequency range. As shown previously, GPU-CAPP allows for the GPU to gain more performance with lower provisioning cost. By expanding the available frequency range, GPU-CAPP can use the higher frequencies to increase performance even further when there is available provisioned power.

Our guidance to the HPC operator is to configure the system using a representative kernel to determine the best local controller for their workload. The power that the system can be configured to use is likely set when they datacenter is built and does not need to be configured later. To choose the optimal local controller, we recommend that a small representative kernel is run on the system for each local controller design. If this is impossible or otherwise untenable, we recommend using the dynamic target warp-based local controller since it achieves the best average performance.

Our guidance to the programmer is to spend less time and money perfecting parallel algorithms when other tasks await. GPU-CAPP improves performance and reduces imbalance because inbalanced areas are most accelerated due to the reduced power draw. Due to this acceleration of imbalanced areas, GPU-CAPP reduces the cost of imperfectly balanced programs so other programming tasks can take priority above the difficult algorithm balancing efforts.

## VI. Related Work

Turbo Boost and RAPL by Intel attempt to address the power capping problem [4]. Turbo Boost does not truly apply because it focuses on *exceeding* the power target when the thermal conditions allow. RAPL seeks to keep the average power under a certain limit which corresponds almost directly to power capping. RAPL uses a centralized controller to accumulate all of the relevant metrics from each node in the system that it can control. In this case, a node is a power-managed subcomponent of the system such as a processor core or GPU SM. Combined with the current power measurements, the RAPL controller determines voltage and frequency settings for each node in the system and sends them back out. This approach is primarily implemented in firmware and software. The centralized implementation causes a few key issues. First, the cycle time of the RAPL control must be slow enough for all of the metrics to propagate to the controller, compute the settings and propagate back to all of the nodes. On a GPU, where the nodes may be far from the controller, this can force the control cycle time to be very slow and potentially be unable to properly control the system under the power constraints. Second, the centralized approach requires a way for all of the node information to get to the RAPL controller. This can be achieved using a bus or separate wires. As the number of nodes in the system scales, the bus must become larger or it will become congested. The use of dedicated wires leads to increases in area and power cost as the number of nodes in the system scales. Realistically, these approaches are untenable as the number of nodes scales to hundreds of SMs and other approaches are required.

Other approaches to this issue [17], [12], [11] use firmware or software-based approaches to measure the system power and change pre-existing hardware settings to reduce the power draw. These approaches change performance states or available hardware (through power gating) in a system but suffer from slower reaction times compared to hardware. As described in [3], fast power issues can occur and the need for low delay power management schemes in the data center exists to reduce the power provisioning costs.

Data center-level approaches to this problem also exist. Ellsworth et al represent this kind of approach by reallocating work among the data center machines to maintain a power cap while improving performance [8]. However, even these approaches ultimately rely on local power capping approaches to help manage the individual system power.

Despite focusing on GPUs, power management in processors directly applies to this work and serves as a comparison point with a large existing body of work. Själander et. al [25] provides a detailed survey of many power management techniques with a focus on DVFS and related variants. In this paper, we focus on the power capping problem, meaning that we seek to maximize the performance of the GPU under a power limit. Classic DVFS is related but does not directly apply because the focus of those approaches is to minimize the power consumption while satisfying a performance constraint. However, the local controllers in GPU-CAPP may implement techniques similar to classic DVFS as they seek to use the available power efficiently.

Other works focus on scaling the voltage to achieve desired performance results in large-scale computing scenarios. Adrenaline [10] attempts to reduce the tail latency of Memcached queries by voltage boosting. Rubik [14] uses fine grain voltage scaling and boosting to reduce variability in latency in datacentric workloads. Harmonia [21] dynamically tunes the hardware operating configurations to

maintain a balance between the power dissipated in compute versus memory access across GPGPU application phases, and DynaCo [22] that coordinates the distribution of power between a CPU and GPU. These three approaches seek to control the voltage to maximize performance in different scenarios.

Other works focus on scaling the voltage to achieve desired performance results in large-scale computing scenarios. Adrenaline [10] attempts to reduce the tail latency of Memcached queries by voltage boosting. Rubik [14] uses fine grain voltage scaling and boosting to reduce variability in latency in datacentric workloads. Harmonia [21] dynamically tunes the hardware operating configurations to maintain a balance between the power dissipated in compute versus memory access across GPGPU application phases, and DynaCo [22] that coordinates the distribution of power between a CPU and GPU. These three approaches seek to control the voltage to maximize performance in different scenarios.

Choi et al. [6] created a mathematical model to determine the limits of several systems under a power cap based on the computational intensity (flops per byte) of the program. GPU-CAPP differs from this work by using a dynamic GPU-focused control mechanism and focusing on GPU-specific behaviors. This model can be used to provide insight into the maximum possible performance under a power cap and can be used to approximate the translation of GPU-CAPP to other platforms.

Tsuzuku and Endo [27] provide a hybrid (dynamic-static) software-based approach to power capping by using a static model to determine ideal frequencies based on profiling the benchmark in combination with a dynamic implementation to control the GPU frequency during operation to reduce the energy consumption when possible. GPU-CAPP focuses on maximizing the performance of the system without violating the power constraint. Tsuzuku and Endos implementation seeks instead to prevent power constraint violations without affecting the energy-to-solution. GPU-CAPP does not directly compare to this work because it does not attempt to control the energy to solution but instead solely focused on maximizing the performance.

Co-Cap [19] caps the frequency of the CPU or GPU in a heterogeneous system to divert the power to the dominant component in the benchmark. However, the primary goal of Co-Cap is to reduce the energy per frame without significant performance impact. In this case, they control the frequency capping using a software-based model that was trained on a large set of benchmarks. GPU-CAPP instead seeks to maximize the performance under a power deliver constraint.

## VII. Conclusions and Future Work

The technique proposed here is a purely hardware approach to maximize the provisioned power efficiency of a GPU. 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 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.

GPU-CAPP can be extended beyond the GPU realm by combining with CAPP and other new implementations to control a variety of components of the system. The local controllers can be different throughout the system. We plan to explore how this implementation needs to be changed to include common configurations such as CPU, GPU and memory. Beyond the common configurations, we also intend to evaluate the inclusion of other interesting nodes such as hardware accelerators or FPGAs. Each of these new nodes requires new models and the development of a simulation framework to evaluate the power and performance of the system.

## References


[1] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt, "Analyzing cuda workloads using a detailed gpu simulator," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 163–174.

[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] A. A. Bhattacharya, D. Culler, A. Kansal, S. Govindan, and S. Sankar, "The need for speed and stability in data center power capping," *Sustainable Computing: Informatics and Systems*, vol. 3, no. 3, pp. 183–193, 2013.

[4] J. Charles, P. Jassi, N. S. Ananth, A. Sadat, and A. Fedorova, "Evaluation of the intel® core i7 turbo boost feature," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*. IEEE, 2009, pp. 188–197.

[5] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*. Ieee, 2009, pp. 44–54.

[6] J. Choi, M. Dukhan, X. Liu, and R. Vuduc, "Algorithmic time, energy, and power on candidate hpc compute building blocks," in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*. IEEE, 2014, pp. 447–457.

[7] 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.

[8] 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.

[9] M. S. Gupta, J. L. Oatley, R. Joseph, G.-Y. Wei, and D. M. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*. IEEE, 2007, pp. 1–6.

[10] 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.

[11] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 39. Washington, DC, USA: IEEE Computer Society, 2006, pp. 347–358. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2006.8

[12] J. A. Joao, M. A. Suleman, O. Mutlu, and Y. N. Patt, "Utility-based acceleration of multithreaded applications on asymmetric cmps," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 154–165.

[13] S. Kanev, J. P. Darago, K. Hazelwood, P. Ranganathan, T. Moseley, G.-Y. Wei, and D. Brooks, "Profiling a warehouse-scale computer," in *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ser. ISCA '15. New York, NY, USA: ACM, 2015, pp. 158–169. [Online]. Available: http://doi.acm.org/10.1145/2749469.2750392

[14] 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.

[15] 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.

[16] Y. Lee, B. Zimmer, A. Waterman, A. Puggelli, J. Kwak, R. Jevtic, B. Keller, S. Bailey, M. Blagojevic, P.-F. Chiu, H. Cook, R. Avizienis, B. Richards, E. Alon, B. Nikolic, and K. Asanovic, "Raven: A 28nm risc-v vector processor with integrated switched-capacitor dc-dc converters and adaptive clocking," Presented at HotChips 2015.

[17] C. Lefurgy, X. Wang, and M. Ware, "Power capping: a prelude to power shifting," *Cluster Computing*, vol. 11, no. 2, pp. 183–195, 2008.

[18] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "Gpuwattch: enabling energy optimizations in gpgpus," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 487–498.

[19] J.-G. Park, C.-Y. Hsieh, N. Dutt, and S.-S. Lim, "Co-cap: energy-efficient cooperative cpu-gpu frequency capping for mobile games," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM, 2016, pp. 1717–1723.

[20] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. De Supinski, "Exploring hardware overprovisioning in power-constrained, high performance computing," in *Proceedings of the 27th international ACM conference on International conference on supercomputing*. ACM, 2013, pp. 173–182.

[21] I. Paul, W. Huang, M. Arora, and S. Yalamanchili, "Harmonia: Balancing compute and memory power in high-performance gpus," in *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ser. ISCA '15. New York, NY, USA: ACM, 2015, pp. 54–65. [Online]. Available: http://doi.acm.org/10.1145/2749469.2750404

[22] I. Paul, V. Ravi, S. Manne, M. Arora, and S. Yalamanchili, "Coordinated energy management in heterogeneous processors," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '13. New York, NY, USA: ACM, 2013, pp. 59:1–59:12. [Online]. Available: http://doi.acm.org/10.1145/2503210.2503227

[23] K. Rajamani, C. Lefurgy, S. Ghiasi, J. C. Rubio, H. Hanson, and T. Keller, "Power management for computer systems and datacenters," in *Proceedings of the 13th International Symposium on Low-Power Electronics and Design*, 2008, pp. 11–13.

[24] Richtek Technology Corporation, "Dual channel pwm controller with integrated driver for imvp8 cpu core power supply," Oct 2015, accessed: 2016-09-30. [Online]. Available: http://www.richtek.com/assets/product_file/RT3606BC/DS3606BC-00.pdf

[25] 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.

[26] K. Straube, C. J. Nitta, R. Amirtharajah, M. Farrens, and V. Akella, "Improving Execution Time of Parallel Programs on Large Scale Chip Multiprocessors with Constant Average Power Processing," in *2017 IEEE 35th International Conference on Computer Design (ICCD)*, November 2017.

[27] K. Tsuzuku and T. Endo, "Power capping of cpu-gpu heterogeneous systems using power and performance models," in *Smart Cities and Green ICT Systems (SMARTGREENS), 2015 International Conference on*. IEEE, 2015, pp. 1–8.