Cooperative Boosting: Needy Versus Greedy Power Management

Indrani Paul*† Srilatha Manne* Manish Arora*‡ W. Lloyd Bircher* Sudhakar Yalamanchili†

*Advanced Micro Devices, Inc.
srilatha.manne@amd.com
lloyd.bircher@amd.com

†Georgia Institute of Technology
indrani@ece.gatech.edu
sudha@ece.gatech.edu

‡University of California, San Diego
marora@eng.ucsd.edu

ABSTRACT

This paper examines the interaction between thermal management techniques and power boosting in a state-of-the-art heterogeneous processor consisting of a set of CPU and GPU cores. We show that for classes of applications that utilize both the CPU and the GPU, modern boost algorithms that greedily seek to convert thermal headroom into performance can interact with thermal coupling effects between the CPU and the GPU to degrade performance. We first examine the causes of this behavior and explain the interaction between thermal coupling, performance coupling, and workload behavior. Then we propose a dynamic power-management approach called cooperative boosting (CB) to allocate power dynamically between CPU and GPU in a manner that balances thermal coupling against the needs of performance coupling to optimize performance under a given thermal constraint. Through real hardware-based measurements, we evaluate CB against a state-of-the-practice boost algorithm and show that overall application performance and power savings increase by 10% and 8% (up to 52% and 34%), respectively, resulting in average energy efficiency improvement of 25% (up to 76%) over a wide range of benchmarks.

Categories and Subject Descriptors

C.1.4 **[Processor Architectures]**: Parallel Architectures; C.4 [**Performance of Systems**]: Design studies

General Terms

Measurement, Performance, Design, Management

Keywords

Thermal management, power management, GPU computing

1 INTRODUCTION

Modern, high-performance client processors are composed of heterogeneous cores that are managed to create a compelling user experience. Power management is a critical piece of the user experience, with the goal to allocate power adaptively across cores to produce the best performance outcome within a fixed processor power and thermal envelope.

The maximum power for a processor (i.e., the thermal design point (TDP)) is set based on running a heavy workload under worst-case conditions [36]. It is an upper bound for the sustainable power draw of the core and is used to determine the cooling system required. Under normal operating conditions, however, not all components are active at the same time or to the same extent, leaving thermal headroom in the system. Power-management technology such as Intel's Turbo Boost [36] and AMD's Turbo CORE [33] take advantage of the thermal headroom to increase the active cores' frequencies until either the maximum performance state or the thermal limit is reached.

A common state-of-the-practice is to boost the frequencies of CPU or GPU cores greedily to utilize all of the available thermal headroom for improving performance. These boost algorithms seek fairness through allocation of power across cores in proportion to expected performance benefits. This works well for many applications in which the type of computation dictates the component that requires boosting. For graphics applications, the GPU is the obvious choice, as is the CPU for many control-divergent, general-purpose applications. However, for applications those require cooperative execution of both CPU and GPU cores, these boost algorithms can break down and degrade – rather than improve performance. This occurs for two reasons: performance coupling and thermal coupling.

Performance coupling refers to control and data dependencies between computations executing on CPU and GPU cores. For example, for peak GPU utilization, the CPU must provide data to the GPU at a certain rate to sustain GPU performance. Performance coupling between CPU and GPU cores is accentuated by the tighter physical coupling due to on-die integration and emergence of applications that attempt a more balanced use of the CPU and the GPU [43]. Thermal coupling refers to the heat exchange that occurs when the CPU and GPU cores share the same die. For example, heat from the CPU cores can accelerate the temperature rise of the GPU. This can cause premature throttling of the GPU cores and loss of performance, whereas the absence of thermal coupling may have permitted the GPU to execute at a higher frequency and, hence, performance. Power-management techniques, such as boost algorithms, must balance the needs of performance coupling between CPU and GPU cores against the negative impact of thermal coupling to deploy power effectively across the CPUs and the GPU.

In this paper, we examine the nuances of thermal coupling and performance coupling and how to balance the two to maximize performance. The apparent solution of throttling the CPU cores to mitigate thermal coupling effects can become counterproductive if the CPU units become too slow to utilize the GPU fully. Further, emerging applications are more likely to utilize both the CPU and the GPU as first-class computational engines, increasing the importance of power-management solutions that balance performance and thermal coupling effects.

This paper makes the following contributions:

- Demonstrates the impact of thermal and performance coupling on system performance using hardware measurements and analysis from a state-of-the-art heterogeneous client system;

ISCA'13, Tel Aviv, Israel.

- Proposes a cooperative boosting (CB) algorithm for the coordinated management of power states on the CPU and GPU to optimize performance; and,
- Provides a detailed, measurement-based analysis of the performance of CB in comparison to a state-of-the-practice boost algorithm for exploiting thermal headroom across a range of benchmark applications.

In the rest of the paper, Section 2 discusses existing power-management architectures. Section 3 explains the thermal and performance coupling phenomena and their interactions motivating the dynamic CB algorithm described in Section 4. In Sections 5 and 6, we present the experimental set-up and our results, while Sections 7 and 8 present related work and our conclusions.

2 BACKGROUND

Heterogeneous processors such as Sandy Bridge from Intel® [28][36] and the Trinity accelerated processing unit (APU) from AMD® [33] (Figure 1) consist of one or more CPU cores in combination with many GPU cores. Both systems feature several heterogeneous cores in close proximity.

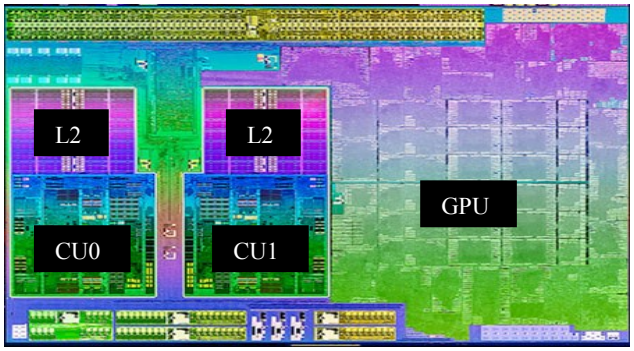

Figure 1: Die shot of AMD Trinity APU [33].

2.1 Power Management

Although the CPUs and the GPU are on independent power planes, they share the same die and system power supply, and hence share the same power and thermal headroom. These heterogeneous processors use a sophisticated power-monitoring and management technology, referred to as Turbo Boost on Sandy Bridge and AMD Turbo CORE on Trinity, to determine the dynamic voltage and frequency scaling (DVFS) states for the CPU and GPU to optimize performance given power and thermal constraints. These technologies use some combination of measured and approximated power and/or temperature values to monitor and guide the power-management algorithm.

The maximum software-visible voltage and frequency for the processor is defined using a combination of heavy activity and worst-case operating conditions. This corresponds to the thermal design point (TDP) power. However, across time-varying workloads it is common for the processor to operate well below the TDP power and, therefore, well below the peak temperature allowed. The difference between the current and peak temperatures is the thermal headroom. Thermal headroom can be utilized by permitting the CPU and/or the GPU components to exceed the maximum frequency and TDP power for short periods. Power-management algorithms differ in how the timing, extent, and duration of the boosted operation are determined. However, regardless of the specific implementation, it is safe to say that both Sandy Bridge and Trinity processors dynamically manage power allocation across the CPUs and the GPU under a pre-set thermal limit. For the rest of the paper, we use the Trinity processor as our test bed for analyzing and optimizing power-management techniques in heterogeneous systems. However, similar concepts and methodologies are applicable to other heterogeneous processors using thermal-based power management.

2.2 Trinity Accelerated Processing Unit

The Trinity APU in Figure 1 contains two PileDriver modules or compute units (CUs), AMD Radeon™ GPU cores, and other logic components such as a NorthBridge and a Unified Video Decoder (UVD). Each CU is composed of two out-of-order cores that share the front-end and floating-point units. In addition, each CU is paired with a 2MB L2 cache that is shared between the cores. The GPU consists of 384 AMD Radeon cores, each capable of one single-precision fused multiply-add computation (FMAC) operation per cycle. The GPU is organized as six SIMD units, each containing 16 processing units that are four-way VLIW. The memory controller is shared between the CPU and the GPU.

Table 1 shows all possible DVFS states for the CPU cores in the A8-4555M Trinity APU. On Trinity, DVFS states can be assigned per CU; however, because the CUs share a voltage plane, the voltage across all CUs is set by the maximum-frequency CU. P0 through P4 are software-visible DVFS states that are referred to as performance states, or P-states, and are managed either by the OS through the Advanced Configuration and Power Interface (ACPI) specification [1] or the hardware. Pb0 and Pb1 are boost states only visible to and managed by the hardware – in other words, entrance to and exit from Pb0 and Pb1 are managed only by hardware.

The GPU has an independent power plane whose voltage and frequency are controlled independently. However, unlike the CPU, the GPU does not have DVFS states visible to software. Entrance to and exit from these states are managed entirely in hardware. Throughout the rest of the paper we refer to these hardware-managed GPU DVFS states as GPU-high (highest frequency), GPU-med (medium frequency) and GPU-low (lowest frequency).

Table 1: HW- and SW-managed DVFS states for the CPU compute units on the Trinity APU.

	P-state	Voltage (V)	Freq (MHz)
HW Only	Pb0	1	2400
	Pb1	0.875	1800
SW-visible	P0	0.825	1600
	P1	0.812	1400
	P2	0.787	1300
	P3	0.762	1100
	P4	0.75	900

AMD's Turbo CORE technology uses the bidirectional application power management, or BAPM, algorithm [8][33] to manage to thermal limits. BAPM controls the power allocated to each thermal entity (TE) in the processor. TEs are defined to be any sub-component of the processor that interfaces with BAPM to report its power consumption and receive its power limits. Once BAPM has assigned power limits, each TE manages its own frequency and voltage to fit within that limit. For the Trinity system evaluated in this paper, BAPM interfaces with the two CPU compute units (CU0 and CU1) and the GPU. At regular time intervals, the BAPM algorithm does the following:

1) Calculates a digital estimate of power consumption for each TE;
2) Converts the power estimates into temperature estimates for each TE; and,

3) Assigns new power limits to each TE based on the temperature estimates.

To estimate the temperature across the die, the chip is divided into regions in which local power and thermal properties are calculated and transfer coefficients (represented as an RC network) are utilized to compute heat transfer among the thermal regions, substrate, and package. Temperatures within each region are computed using numerical methods.

The BAPM algorithm is optimized for a fair and balanced sharing of power between the TEs. When thermal headroom is available, BAPM proportionally allocates power to each TE using a pre-set static distribution weight derived using empirical analysis reflecting the individual thermal properties of each TE (i.e., its thermal behavior for a given power). Such static allocation is an effective choice in the absence of dynamic feedback from application execution. When the core reaches its thermal limit, BAPM reduces the allocation of power to all TEs in the system. BAPM is designed to provide reasonable performance improvements without any significant outliers for today's applications.

3 POWER, THERMALS, AND PERFORMANCE

Managing power in a heterogeneous system is a complex task given the variability in power and performance within each TE. This section presents an analysis of the relationship between power and heat in the CPU and GPU and its impact on performance. The section concludes by showing the need for cooperative boosting between the CPU and GPU.

3.1 Motivation

To motivate the rest of the discussion, we present an example of the impact of thermal coupling. Figure 2 illustrates thermal coupling using an AMD A8-4555M Trinity APU comprised of two dual-core CPU compute units (CU0 and CU1) and one six-SIMD unit GPU (Figure 1). The x-axis shows time. The left-side y-axis shows measured power relative to time zero using the techniques described in Section 5. The right-side y-axis shows the peak die temperature relative to the maximum junction temperature.

Initially, the GPU operates at its highest frequency and the CUs are fixed at a low-frequency, low-power state. After the GPU heats up and stabilizes, at around 230 seconds, we enable the BAPM algorithm to control power allocation. Because there is significant thermal headroom available, BAPM allocates additional power to CU0 and CU1 and they enter a higher-power DVFS state. Not only do the CUs increase their power dissipation, but due to thermal coupling and the impact of heat on leakage power, the GPU power also rises. The increase in system power causes an increase in peak temperature and eventually triggers temperature-based throttling of both the CUs and the GPU at around 267 seconds to maintain a steady-state peak junction temperature. This results in a net performance loss for all components.

This behavior can be attributed to thermal coupling effects between the TEs. As the GPU warms up (see Figure 2), CU1 has a stronger thermal coupling to the GPU due to its proximity to the GPU (Figure 1), and so its power is initially higher than that of CU0 although they are both relatively low. When they switch to a higher-performing DVFS state, at around 230 seconds, the power in both CUs increases, but thermal effects cause CU0 power to exceed CU1 power. CU0 is on the edge of the die, and its heat is trapped between the edge of the silicon and CU1. The GPU acts as a thermal sink for CU1 due to its larger die area and more distributed heat and, as a result, lower temperature. However, once steady state is reached (more than 267 seconds), CU0 and CU1 temperatures stabilize to roughly equal values.

We conducted two additional experiments to support the preceding discussion on thermal coupling between the CUs and the GPU. In the first experiment, we boosted the CUs to run at a higher power while the GPU executed the same workload at a constant voltage and frequency. We observed the GPU temperature was 6°C higher once the CUs were boosted, indicating thermal coupling between CU1 and the GPU.

In the second experiment, we performed temperature measurements with a high-power, two-thread CPU application. We first pinned the threads to CU0, then pinned them to CU1. The GPU is idle and power is managed by BAPM. When the application ran on CU0, we observed the peak die temperature was higher than when the application ran on CU1, indicating worse heat flow from the CU next to the edge of the die. Further, the GPU and idle CU temperatures rose by 13°C when one of the CUs was active with all others idle, indicating heat transfer effects.

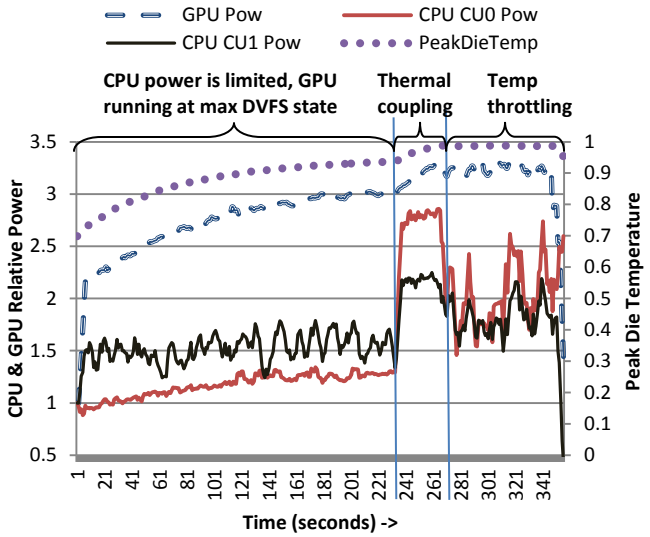

Figure 2: Example of the impact of thermal coupling.

3.2 Thermal Signatures and Thermal Coupling

Section 3.1 discussed the heat transfer properties of the TEs on the die. This section details the differences in the thermal characteristics of CPUs and GPUs and how they affect performance-coupled applications.

The thermal signature of a TE reflects its ability to translate power to temperature. It is measured by the distribution of power density across the occupied area. In this sense, the thermal signature of a GPU is quite different from that of a CPU – the latter is more "thermally dense". In Figure 3, we show on the left side a simulated heat map of the Trinity system when running a CPU-centric, L1 cache-resident, high-power workload with an idle GPU. The simulated heat map was constructed by feeding measured power levels and power density while running the workload into a thermal grid model. The right side shows a heat map for HotSpot [10], a GPU-centric workload with the serial portion being executed on the CPU. The thermal maps show the steady-state thermal fields produced with the BAPM algorithm across the two CPU CUs, the GPU, and the NorthBridge as labeled in the figure. Tjmax refers to the maximum junction temperature allowed by the die. The temperature distributions in Figure 3 are steady-state distributions and therefore correspond to the region of Figure 2

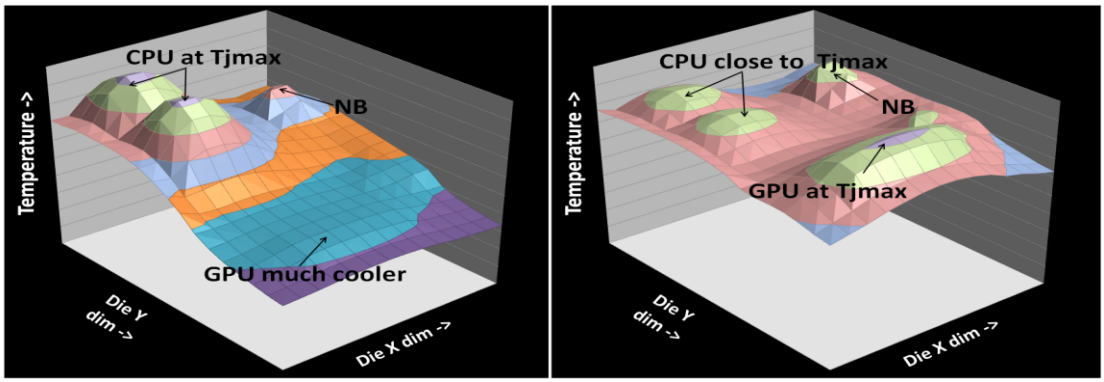

Figure 3: Thermal densities under CPU-centric (left) and GPU-centric (right) workloads.

after 267 seconds (i.e., after the BAPM algorithm throttled the CPUs and GPU once they reached peak junction temperature).

The thermal characteristics of the workloads vary significantly. The CPU-centric workload shows high heat density in the CPU CUs while the GPU-centric workload shows a wider and flatter temperature distribution across the GPU. The computational area of the CPU, which is where most of the power is consumed, is much smaller than the computational area of the GPU. The complex, out-of-order CPU structures combined with their relatively small areas lead to higher thermal density for the same power and, thus, higher temperatures [22]. The GPU, on the other hand, performs computation across many simple in-order SIMD units that encompass a large area, leading to a lower thermal density for the same amount of power.

There are two consequences to the higher thermal density in the CPU. The first is that the CPU consumes its available thermal headroom more rapidly than the GPU. In our analysis, we observed that the CPU heats up approximately 4X faster that the GPU. As a result, the GPU can sustain a higher power boost than the CPU for a longer period before locally reaching the thermal limit. In some cases, this results in sustained power dissipation that is higher than the TDP power. For example, in the simulations shown in Figure 3, the TDP of the APU complex is 19W; the total power for the CPU-centric workload is 18.8W, while the total power for the GPU-centric workload is 19.7W for the same thermal limit.

The second consequence of the higher thermal density in the CPU is the destructive effect of thermal pollution on other components on the die. The rate and extent of thermal pollution depends on the thermal signatures of the entities. The distinct thermal signatures lead to a larger thermal gradient between the CPU and GPU when the CPU is active than when the GPU is active. Heat from the CPU spreads, heating neighboring components, increasing leakage, and accelerating temperature rise. The thermal coupling effects can be seen in Figure 2 and Figure 3.

In a thermally coupled system, the TEs do not influence each other as long as they are all running well below the thermal limit. Power management employs boost algorithms to improve performance by pushing the processor to operate near the thermal limit, reallocating power across the CPU and GPU. As shown in Figure 2, boosting based on available thermal headroom can sometimes be detrimental to the application performance. The complexity of the power-management task is exacerbated in heterogeneous systems because application performance relies on components with widely varying thermal signatures and coupling.

3.3 Performance Coupling

For many applications, the type of computation dictates the component to be used. For graphics applications, the GPU is the obvious choice, as is the CPU for many control-divergent, general-purpose applications. However, for compute offload applications, both the CPU and the GPU are possible candidates for different portions of the computation or data processing. For such applications, the CPU and the GPU are performance-coupled. For example, a slow CPU can starve the GPU of data, leading to underutilization of the GPU. Therefore, we must balance the requirements of performance coupling with the realities of thermal coupling to produce the best overall outcome.

This section explores the performance-coupled nature of offload applications by demonstrating the sensitivity of performance to the CPU performance states. In the following experiments, we statically fix the highest-performing (i.e. frequency) CPU P-state permitted by the power-management algorithm, which we denote

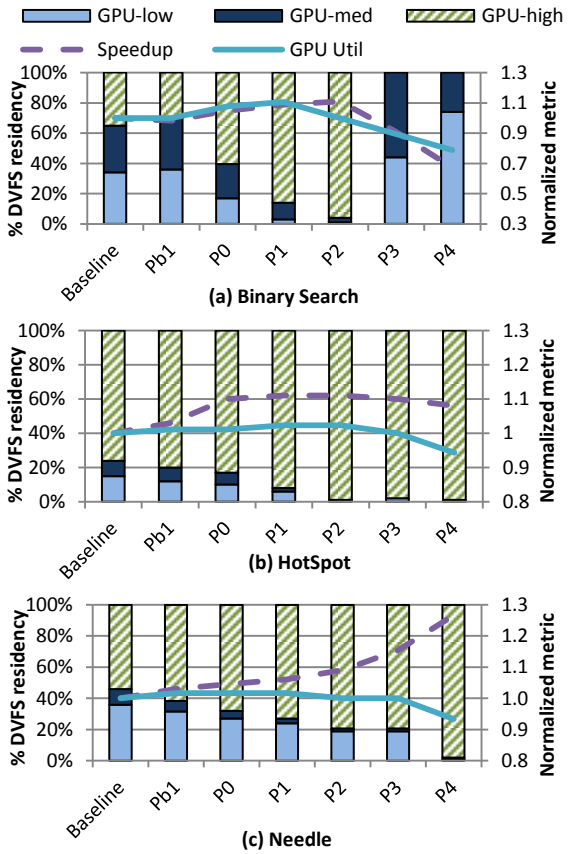

Figure 4: Impact of CPU P-state limit on performance, GPU residency, and GPU utilization.

as the P-state limit. The local CPU power controller may change the P-state to a lower-performing P-state based on thermals, but it cannot exceed the P-state limit. Figure 4 presents the impact of CPU P-state limits for Binary Search, HotSpot, and Needle.

In Figure 4, the x-axis is labeled with the CPU P-state limit for that experiment. In addition, we show results for the baseline, which refers to the default Trinity power-management system (Section 2.2). Limiting to Pb0 means all P-states are available to the power-management controller, which is the same as the baseline case and hence not shown separately. The left-side y-axis refers to the stacked bar charts, and it shows the percent of time the GPU spends active in low-, medium-, and high-DVFS states. The right-side y-axis shows speed-up and GPU utilization normalized to the baseline results. We define GPU utilization as the ratio of time when at least one of the SIMD units in the GPU is active versus the total execution time. These data were collected on the Trinity system hardware described in Section 5.

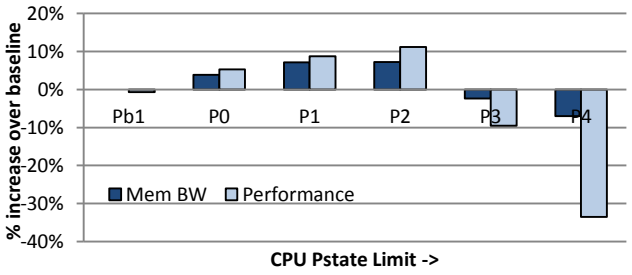

Figure 5: P-state limit effects on GPU memory bandwidth.

For Binary Search and HotSpot, as the CPU maximum frequency decreases (moving to the right), the application transitions from being thermally coupled to being performance-coupled. As the frequency decreases from Pb1 to P2, the GPU spends a larger portion of time in its higher-frequency performance states, indicated by GPU-high. In addition, speed-up increases, indicating that the GPU is utilizing the extra thermal headroom to improve performance. However, as the CPU frequency decreases beyond P2, we see a marked reduction in overall performance because performance coupling begins to dominate. GPU utilization decreases beyond P2, indicating that the GPU is being starved by the slower CPU. In the case of HotSpot, the thermal headroom permits the GPU to continue operating at its highest-performance state when active. However, for Binary Search, the local GPU power controller reduces the GPU frequency because of a significant drop in GPU utilization. For Needle, the GPU is thermally limited by the CPU across all P-state limits. Performance improves by 27% when the CPU operates at its lowest-performance P-state. However, performance coupling becomes more dominant at a CPU P-state limit of P4 because GPU utilization starts to decrease.

As is evident from the preceding analysis, during any time interval there is an optimal CPU operating frequency (and, equivalently, P-state) for each application depending on its thermal and performance coupling characteristics. We refer to this P-state as the critical P-state and the corresponding frequency as the critical frequency. We observe that the critical P-state is a time-varying function of the workload and our goal is to have the CPU always operating in the critical P-state. Our approach is to first define a measurable performance metric that is sensitive to the CPU and GPU frequencies. By tracking the behavior of this metric, we can determine and set the CPU to its critical P-state periodically.

Microsoft® Windows® OS Power Management [44] using ACPI provides a capability for managing the CPU P-state based on application requirements. However, in experiments with ACPI, the lower-performing P-states were never utilized. There are a number of shortcomings here. First, the OS uses the highest utilization among all cores as the metric to determine the P-state of all cores, while most of the applications analyzed have varying degrees of core utilization. Second, it does not consider the performance requirements of an application, while our analysis shows that applications experience phases that require higher CPU performance. Finally, ACPI does not include any concept of performance coupling or thermal coupling; therefore, it cannot be used readily to regulate to either requirement.

In this paper, we propose to use the GPU memory access rate gradient as a proxy for CPU-GPU performance coupling [26]. When the CPU transitions to a lower-performing P-state, the GPU frequency and the memory access rate increase due to decreasing thermal coupling effects. However, if the CPU operation drops below the critical frequency, the GPU is starved by the slower CPU and the GPU memory access rate drops. This observation can be used as a starvation hint to transition the CPU to the critical P-state. In Figure 5, we illustrate the impact of P-state-limiting on GPU memory bandwidth for Binary Search, HotSpot, and Needle. Memory bandwidth tracks performance for these three benchmarks, indicating the applicability of this metric. In addition, to deal with phase changes in the applications that require high CPU frequency, we also use retired instructions per clock (IPC) of the CPU as a measure of the application's sensitivity to CPU frequency [20].

4 COOPERATIVE BOOSTING

Based on the preceding analysis, we see that the power-management problem is one of determining the critical P-state – the state that mitigates the negative effects of thermal coupling while providing sufficient power for performance-coupled operation. This section describes our CB algorithm for the dynamic determination of the critical P-state.

4.1 Structure

The CB algorithm operates as a decision layer on top of the baseline Trinity power-management system (from now on referred to as the baseline). The baseline was designed to optimize the average case behavior across a wide range of applications with a fair allocation of power to both the CPU and the GPU by utilizing all of the available thermal headroom. The CB algorithm enhances such thermal headroom-based management techniques by tailoring its behavior to i) the asymmetry of the thermal behavior of the CPU and GPU and ii) the phase behavior of applications – specifically, thermal and performance coupling over short intervals. Such optimization becomes increasingly important as emergent workloads are making more balanced use of the CPU and the GPU. The goal of CB is to determine when thermal coupling effects are detrimental and set the critical P-state limit under such cases. This is the highest-performing P-state the baseline system is permitted to use for the CPU. The voltage and frequency for the GPU is managed by the baseline and is not directly managed by CB; thus, CB essentially controls the CPU limits under which the baseline power-management system operates. Figure 6 illustrates the CB algorithm flow.

Cooperative Boosting Algorithm

At beginning

```
If (Peak_Temp > Temp_Threshold) {
   EnableCB();
   Prev_PStateLimit=P0;
}
```

Every 10 ms

```
for i=0, i<Core_Count; i++ {
   IPC[i] = ReadIPC(i);
   Active_Clks[i] = ReadActiveCoreClock(i);
}
Weighted_IPC = ComputeWeightedIPC(IPC,Active_Clks)
IPC_Gradient = Weighted_IPC – Prev_Weighted_IPC
Prev_Weighted_IPC = Weighted_IPC;
Peak_Temp = ReadPeakTemp();
GPU_Mem_BW = ReadGPUMemBW();Short_Term_BW =
ComputeShortTermBW();
Long_Term_BW = ComputeLongTermBW();

If (CB_Enabled && (IPC_Gradient > =IPC_Threshold)) {
   Prev_PstateLimit = CPU_PStateLimit;
   UnsetPStateLimit();
}
```

Every 500 ms

```
If (CB_Enabled && (IPC_Gradient < IPC_Threshold)) {
   CPU_PStateLimit = Prev_PStateLimit;
   BW_Gradient = Short_Term_BW – Long_Term_BW;
   If (BW_Gradient >= BW_Threshold) {
      Last_Good_PState = CPU_PStateLimit;
      CPU_PStateLimit++; /* Until P4 is reached */
   }
   Else {
      CPU_PStateLimit = Last_Good_PState;
   }
}
```

Figure 6: Cooperative boosting algorithm.

CB monitors temperature and performance metrics at intervals of 10 ms and modifies the CPU P-state limit at intervals of 10 or 500 ms to account for frequent workload phase changes and relatively slow thermal response times, respectively. Periodic enforcement of P-state limits at short intervals can cause dampening of natural workload behavior, whereas long intervals can cause inaccuracy in measurements. Therefore, we decouple the monitoring and the control intervals in CB. These intervals are chosen carefully to account for the long thermal rise times, shorter performance intervals and workload activities, and overheads of P-state change. In practice, the intervals can be adjusted based on RC time constants of the die, floor plan, and process technology. The critical P-state limit can be computed at any granularity (per core, per CU, or for the entire CPU). In this paper we apply the same critical P-state limit to all CPU cores due to a shared voltage plane.

4.2 Algorithm

The CB algorithm operates in three major steps: i) being invoked, ii) determining and setting the critical P-state limit, and iii) damping control to prevent oscillations. In the beginning, the processor starts with the highest-performance-boost P-states for the CPU CUs and the GPU, and power and temperature are managed by the baseline. At intervals of 10 ms, we determine if the processor is thermally limited and if CB should be applied. If so, power management moves into CB mode.

The second step is determining which CPU P-state limit to apply. This involves both instrumentation and decision-making. CB samples peak die temperature, per-core retired IPC, and memory bandwidth usage at every 10-ms monitoring interval. Note that although the algorithm in Figure 6 and the discussion in Section 3.3 refer to the GPU memory bandwidth, the implementation of CB uses a combined CPU and GPU bandwidth due to hardware restrictions that prevent GPU-only bandwidth measurements while CB is enabled. We found that this did not hinder the performance of the algorithm due to the overwhelming dominance of the GPU in memory bandwidth usage.

Each core's IPC is weighted by the number of active clock cycles seen by the core during the sampling period, and the aggregate IPC for the CPU is the sum of the weighted IPCs for all four cores. For memory bandwidth, in addition to monitoring memory bandwidth at each 10-ms interval, CB also keeps a short-term and a long-term moving average of memory bandwidth to track how the bandwidth changes over time. Because P-state limiting to reduce thermal coupling effects is made at intervals of 500 ms, the short-term average is computed over the last 500-ms interval while the long-term moving average is computed over the last five such intervals. Bandwidth gradients are computed by comparing the short-term moving average with the long-term moving average.

The CPU P-state limit may be established by observing changes in the CPU IPC or GPU memory bandwidth (these metrics were advocated as proxies to detect performance-coupled operation in Section 3.3). CB utilizes the gradient of memory bandwidth to determine the critical P-state for the CPU. If the gradients are positive, then the workload benefits from shifting power to the GPU. In this case, the algorithm moves the CPU P-state limit to a lower performance state. The converse occurs when the gradient is negative. Over time, the controller is trying to move the CPU to the critical P-state.

If the workload enters a CPU compute-intensive phase, as indicated by a high-CPU IPC phase, the current P-state limit is saved and the control part of CB is suspended by disabling P-state limiting. The check for CPU IPC changes occurs at 10-ms intervals to capture frequent phase changes and data dependencies. When the CPU workload exits the compute-intensive phase, CB operation is resumed at the saved P-state limit. This dampens multiple transitions through performance states arising from a short

burst of high-power CPU phases that would otherwise re-initialize the CPU performance state to the highest performance state. Finally, to prevent oscillation between a pair of P-state limits, we employ a damping mechanism such that a new P-state limit is weighted towards the previous P-state limit after more than a certain number of transitions.

To encompass non-performance-coupled applications that may have a constant CPU IPC (such as SPEC CPU2006 applications), we use an absolute average IPC in conjunction with IPC phase changes for CPU-centric workloads with no activity on the GPU. Although CPU-centric workloads are not the focus of this paper, we show that our CB algorithm can sometimes improve the performance of these applications by limiting the performance state when the application is memory-bound.

5 EXPERIMENTAL SET-UP

We perform all measurements and analysis on an AMD A8-4555M Trinity APU with 19W TDP. Base CPU frequency is 1.6 GHz, with AMD Turbo CORE frequency up to 2.4 GHz. The GPU frequency is 320 MHz with AMD Turbo CORE frequency of 423 MHz [47]. We use four, 2-GB DDR3-1600 DIMMs. Hardware performance counters for IPC, memory bandwidth, etc., are monitored using performance libraries running in Windows OS. A maximum cap on the CPU P-state limit is implemented using model-specific registers as described in [8].

We evaluate three different boost algorithms. The baseline is the BAPM algorithm, which is the state-of-the-practice algorithm in the Trinity power-management system described in Section 2.2. The second is the CB algorithm described in Section 4.2. Third, we evaluate the behavior of a static P-state-limit algorithm in which a fixed P-state limit is applied throughout the entire run of the application. This means that the CPU can enter a lower-performing (but not higher) P-state than the P-state limit. We refer to this as the static PX limit scheme, where PX is one of the performance states (e.g., P1, P3, etc.). For CB, P-state limits are applied according to the algorithm described in Section 4.2. Although CB can be implemented in any layer such as hardware, power-management firmware, or system software, we implement CB at the system software level by layering it on top of the baseline.

For CPU and GPU power and temperature, we use the digital estimates provided by the power-management firmware running in the Trinity system, accuracies for which are described in [33]. For all schemes, we run the benchmarks for at least a few minutes to reach a thermally stable steady state. A fixed-time cool-down period is applied before each run to eliminate any variations in start temperature. We also run many iterations of the application and take an average across those to eliminate run-to-run variance in our hardware measurements.

We use 18 applications, summarized in Table 2. These are a mix of both state-of-the-art and emergent applications. Eight of them are from Rodinia (NDL, LUD, HS, SRAD, CFD, BFS, KM, and BP [10][11]), three are from the AMD APP SDK (BF, MM, and BS [2]), two are stand-alone (FAH [17] and Viewdle [42]), and five are from SPEC CPU2006 (Mcf, Lbm, Perl, Pvr, and Gcc [40]). We selected the applications to represent i) GPU-centric (where GPU is used as a compute accelerator with CPU feeding the data to the GPU), ii) CPU-GPU mixed workloads (where computation is more balanced between CPU and GPU although the fraction of work division may not be the same), and iii) CPU-centric workloads (where computation is done only on the CPU and the GPU is unused). All GPU applications execute one or more parallel kernels for multiple iterations to reach steady-state thermals. The SPEC CPU applications are run with four threads, one on each core.

We report performance, power, and energy efficiency as defined by the energy-delay^2 product (ED^2) [21]. We show all values normalized to the baseline scheme, which is the default Trinity power-management system. Average total power (CPU and GPU) and average energy efficiency are also measured over the entire run-time of an application.

Table 2: Summary of benchmarks.

BM (Description)	Problem Size	Type
NDL (Needleman-Wusch [10])	4096x4096 data points, 1K iterations	GPU
LUD (LU decomposition [11])	512x512, data points, 500 iterations	GPU
HS (HotSpot [10])	1024x1024 data points, 100K iterations	GPU
SRAD (Image Proc [10])	502x458,500K iteration	GPU
BF (BoxFilter SAT [2])	1Kx1K input image, 6x6 filter,10K iterations	GPU
MM (Matrix Mult [2])	2Kx2K, 10K iterations	GPU
FAH (Folding at Home [17])	Synthesis of large protein: spectrin$	GPU
CFD (Computational fluid dynamics [10]	200K elements, 20K iterations	GPU
BFS (Breadth first search [10])	1M nodes, 1K iterations	GPU
BS (Binary Search [2])	4096 inputs, 256 segments, 1M iterations	GPU
KM (Kmeans [10])	819200 points, 34 features, 1K iterations	Mixed
BP (BackProp [10])	252,144 input nodes, 10K iterations	Mixed
Viewdle (Haar facial recognition [42])	Image 1920x1080, 2K iterations	Mixed
Mcf (CPU2006 [40])	4 threads, Ref input	CPU
Lbm (CPU2006 [40])	4 threads, Ref input	CPU
Perl (CPU2006 [40])	4 threads, Ref input	CPU
Pvr (CPU2006 [40])	4 threads, Ref input	CPU
Gcc (CPU2006 [40])	4 threads, Ref input	CPU

6 RESULTS

In this section, we present performance, power, and ED^2 results for CB and the static P-state limit algorithm. All results are shown relative to the baseline BAPM algorithm described in Section 2.2, and all performance and power numbers are measured results from running the applications in Table 2 on real hardware.

6.1 Performance

Figure 7 illustrates the speed-up of CB and static schemes. Across the 18 applications, we see a 10% speed-up with CB, a 3% speed-up with P0 (the highest-performance software-visible P-state), a 1% speed-up with P2, and a 10% performance loss with P4. For the performance-coupled workloads (i.e., GPU-centric and CPU-GPU mixed workloads), the average speed-up with CB is 15%. The static schemes clearly demonstrate good performance gains compared to the baseline for certain types of workloads but impose a high performance penalty for others, motivating the need for dynamic schemes.

GPU-centric applications such as NDL, LUD, MM, and SRAD improve in performance compared to the baseline with both CB and static. In general, these applications have low CPU IPC and are not very sensitive to CPU performance in the frequency ranges

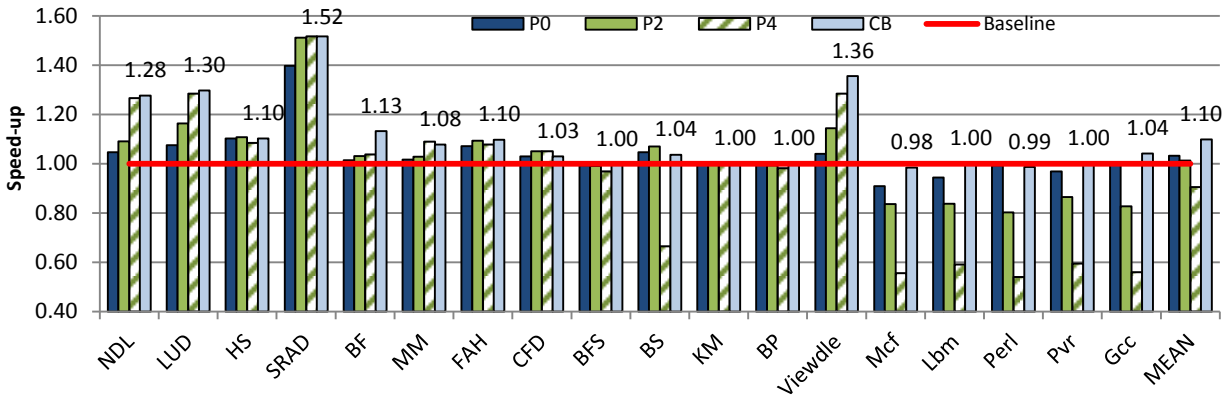

Figure 7: Performance results with static limits and CB.

explored. Both CB and static P4 limiting show comparable gains, with performance improvement as high as 52% in SRAD. Thermal coupling dominates these applications at all CPU frequencies because they have high activity in the GPU and, hence, high power requirements. The critical P-state for the CPU is at a lower frequency than the lowest P-state P4 available in our part. These workloads reach the peak temperature quickly, and high-performance CPU P-states result in excessive thermal throttling without a commensurate application performance improvement.

Similar thermal coupling effects occur in applications such as HS, BS, and FAH. However, here we reach the critical CPU P-state before the lowest P-state limit of P4. At P2, thermal and performance coupling effects are balanced and we see the maximum performance gains. Decreasing CPU frequency beyond P2 causes performance coupling to dominate over thermal coupling and degrades performance by 3%, 34%, and 1%, respectively, for HS, BS, and FAH at P4. CB achieves comparable results to the critical P-state of P2.

Applications such as KM, BFS, BP, and CFD see minimal to no benefits compared to the baseline with static or CB schemes. KM, BFS, and BP never reach the peak junction temperature, and so CB never invokes P-state limiting. Although KM has high-IPC phases, it is primarily memory-bound and its performance stays relatively flat with CPU frequencies. BP has serial phases between parallel kernels requiring significant CPU-GPU communication. BFS has a high control flow divergence with low GPU activity. In both BFS and BP, CB results in the same performance as the baseline, whereas static P4 limit shows performance degradation up to 3% due to performance coupling. Although CFD is heavily memory-bound, it reaches the peak temperature due to high activity and a relatively high compute-to-memory ratio in the GPU; as a result, it shows a slight improvement of 3-5% compared to the baseline scheme using static limiting and CB. Performance gains from reducing thermal coupling effects flatten out beyond P2 as the memory-related stall time of the kernel starts to dominate.

In balanced workloads such as Viewdle, a face-recognition application, both the CPU and the GPU are utilized heavily for computation. Thermal coupling is dominant at the higher CPU frequencies, and so static P-state limiting to both P2 and P4 improves performance compared to the baseline. CB, however, outperforms all static P-state-limiting schemes by dynamically adjusting to the critical P-state based on application needs. Viewdle's IPC varies periodically from low to high, and it is sensitive to CPU frequency during high-IPC phases. CB dynamically shifts power to the CPU during high-IPC phases and to the GPU during low-IPC phases, thereby limiting the impact of thermal coupling while providing the required power for performance coupling. Section 6.3 provides further insights in Viewdle's performance. We see similar behavior with BF, which is an image-filtering application with frequent CPU communication phases between the horizontal and vertical passes in the image blur filter. CB performs 13% better than the baseline and 9%-12% better than any of the static schemes in the case of BF.

Finally, we analyze the performance of CPU-centric, non-performance-coupled applications such as Perl and Pvr. As we see in Figure 7 the baseline does very well for these workloads and static limiting significantly degrades performance. CB largely performs as well as the baseline, indicating that CB is a well-rounded approach to multiple usage scenarios. Although analyzing multiple non-performance-coupled applications (e.g., a CPU-centric app and a GPU-centric app) running together was not the focus of our paper, we believe CB will perform as well as or better than the baseline because CB tries to limit CPU power only when it is not needed.

6.2 Thermal and Performance Coupling Analysis

In Figure 8 we illustrate how CB mitigates the effects of thermal coupling in the case of BS. The y-axis indicates the measured peak temperature normalized to Tjmax. With a static limit of P4, the application heats the chip to a value less than the peak. CB, on the other hand, does not initially restrict the baseline algorithm; instead, it tries to find the critical P-state for CPU once we approach the peak temperature threshold. As power is shifted from the CPU to the GPU, peak die temperature decreases because the GPU is able to sustain a higher power boost for a longer period due

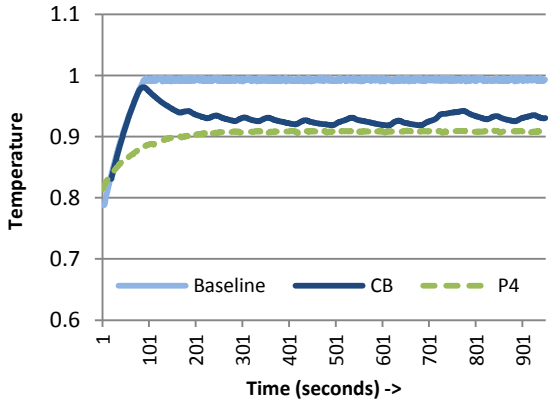

Figure 8: Thermal behavior of Binary Search with CB.

to its lower thermal density, as described in Section 3.2. Further, the effects of thermal coupling become less dominant because the CPU is running at a lower P-state. As a result, the GPU residency in the high-performance state increases significantly compared to the baseline, thereby improving application performance. Moreover, the short variations in temperature result from the fact that CB constantly adjusts the critical P-state based on workload phases. This helps balance performance and thermal coupling effects.

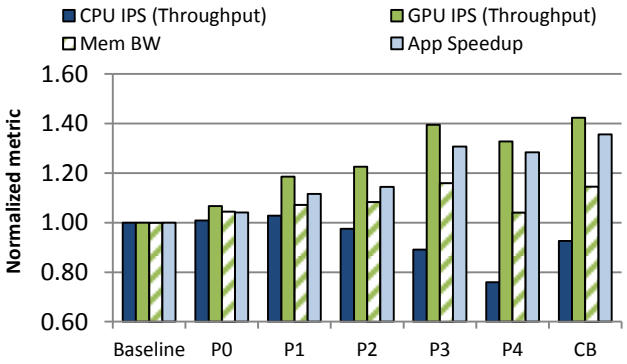

Figure 9: Viewdle performance analysis with CB.

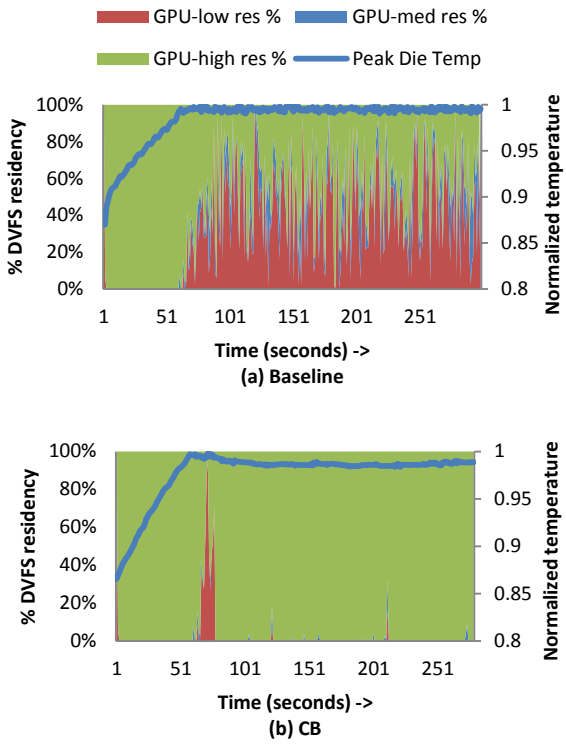

Figure 10: Thermal throttling in Needle with CB.

Figure 9 provides further insights into the performance of Viewdle in terms of instructions per second (IPS), memory bandwidth, and speed-up. As we apply CPU P-state limiting with lower-performing P-states, CPU IPS understandably drops. However, the GPU IPS continues to increase, and so does memory bandwidth due to the GPU's ability to sustain higher frequencies because of the reduction in thermal coupling. For P-state limiting beyond P3, both GPU throughput and memory bandwidth drop due to performance coupling effects. However, with CB, the CPU P-state limit is managed dynamically to balance performance and thermal coupling effects: GPU throughput and speed-up increase by 42% and 36%, respectively, compared to the baseline.

In Figure 10, we illustrate how CB mitigates the effects of thermal coupling when running Needle. The left-side y-axis shows GPU residencies in the different performance states. The right-side y-axis shows the measured peak temperature normalized to Tjmax. In the baseline case (Figure 10(a)), we see a considerable amount of residencies in the medium and low GPU frequencies once temperature reaches the steady state to maintain performance within the maximum thermal limits. GPU frequency throttling occurs because of thermal coupling and heat transfer effects from the CPU to the GPU as both CPU and GPU are run at their maximum frequencies during the initial ramp-up stage due to availability of thermal headroom. However, as shown in Figure 10(b), CB tries to find the critical P-state for the CPU once we approach the peak temperature threshold. Once invoked, CB starts shifting power to the GPU. Because Needle is a high-power workload, we see a slight temperature-based throttling initially, after which the temperature decreases and power shifts from CPU to GPU. This allows boosting of the GPU to higher frequencies for a much longer period, thereby improving application performance.

Because CB is designed to mitigate detrimental effects of thermal coupling in thermally limited situations, it effectively lowers the peak operating temperature of the processor opportunistically compared to the baseline (2% lower on average across all applications). Although temperature is not a direct optimization goal for CB, lower peak temperatures have many additional benefits: i) increased TDP power budget to achieve more performance within a fixed thermal envelope; ii) lower cooling costs within a fixed power budget; iii) lower leakage power and, hence, lower overall energy; and/or, iv) improved reliability through increased mean-time-to-failure rates.

6.3 Power and Energy

The power saving achieved with CB over the baseline is illustrated in Figure 11, which shows an average power savings of 8% across all applications and an average of 10% across performance-coupled GPU-centric and mixed workloads. Highest power reduction is seen in BS, where we see a 5% reduction in average peak temperature and, hence, leakage power during run-time. BFS, BP, and KM never reach their peak temperatures, so power savings are minimal because CB does not limit P-states under such cases and allows both CPU and GPU to take full advantage of boosting. We also achieve a small amount of power savings in the SPEC CPU2006 workloads, up to 11% with Mcf because CB continuously tracks high-IPC compute-bound phases. When the workload encounters memory-bound phases, a P-state limit is applied to lower the frequency; this limit has little to no performance impact but it saves power [20].

Figure 12 shows the ED^2 product (lower numbers signify improvement over the baseline). With CB, we see an average energy-efficiency improvement of 25% (up to 76%) across all applications, and 33% across performance-coupled GPU-centric and mixed CPU-GPU workloads. Interestingly, a static limit of P4 (the lowest-performing P-state) performs 30% worse than the baseline, but we see an improvement of about 10% with static limits of P0 and P2 due to reduction in thermal coupling and a large reduction in power at those states. However, as shown in Figure 7, a fixed static P-state of P0 and P2 results in significant performance outliers for CPU-centric workloads; hence, it is not a

viable solution. CB, however, can achieve similar or better results for performance and energy efficiency than any static scheme without requiring any offline profiling or user intervention.

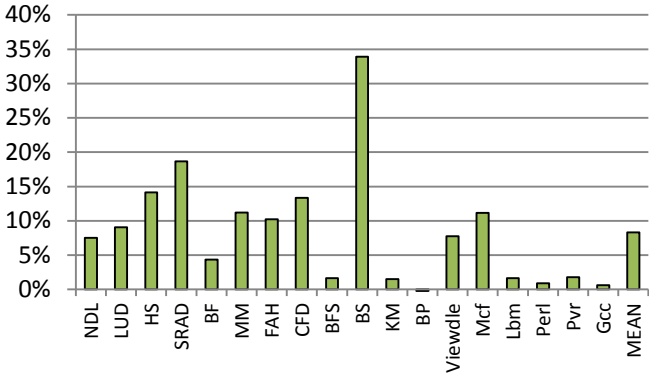

Figure 11: Reduction in power for CB relative to baseline.

6.4 Summary

In this section, we summarize our results and insights. First we show that workloads with high GPU activity are more sensitive to thermal coupling with the CPU. The baseline can degrade performance while both CB as well as static P-state limiting shift a greater portion of the power to the GPU, reduce thermal coupling, and improve performance. For applications with tight performance coupling with the CPU, CB finds the critical P-state and thus performs better.

For applications with very low GPU utilization such as high control flow divergence, thermal coupling may not be a factor since these workloads tend to run much cooler. While the baseline does not hurt performance, static schemes can degrade performance significantly by amplifying the low GPU utilizations when the CPU P-state is fixed below the critical P-state. However, CB dynamically detects when an application is not thermally limited and stops limiting CPU's P-state under such cases. This allows CB's performance to track baseline for such workloads.

Balanced workloads that actively utilize both the CPU and GPU are particularly susceptible to thermal coupling effects. CB outperforms the baseline and static schemes by continuously tracking the time-varying critical P-state during execution. CB uses only the power it needs, and thus reducing thermal coupling without impacting performance coupled operation. This is one of the fastest growing classes of future workloads [30][43].

For non-performance coupled CPU-centric workloads, greedy boosting approaches work well while static schemes understandably perform poorly since performance scales with frequency. CB performs largely as well as the baseline since the critical P-state tends to be the highest performance state. However, CB delivers slightly better performance for memory bound workloads by detecting memory bound phases and adjusting the critical P- state, which builds up thermal credits for compute phases that need higher performance state.

In summary, CB is a well-rounded technique that can be used to dynamically manage power, performance and thermals across a wide range of applications. Although for a given application one can profile the critical P-state limit statically offline, such an approach is impractical and often detrimental if the goal is to support a variety of applications including emergent and as yet unanticipated ones. CB improves over current headroom based greedy boost algorithms by balancing the needs and dependencies of CPU and GPU performance with the effects of thermal coupling.

7 RELATED WORK

CB differs from the large body of work on dynamic thermal management. The latter dealt primarily with homogenous multi-core processors, did not consider coupled interactions among cores, and evolved originally to prevent harmful thermal capacity violations to peak temperature. Consequently, architectural efforts focused first on preventing unwanted thermal excursions and quickly evolved to balancing the system-level performance impact of such management techniques [9][31]. The range of techniques included i) activity migration, ii) power reduction by various forms of throttling [18][45], iii) feedback control [37][38][39][46], or iv) a combination of techniques to balance performance loss against thermal management. These techniques were concerned primarily with managing peak temperatures.

That philosophy continued with the advent of multi-core architectures [16] through run-time techniques such as heat and run [34] or a combination of design- and run-time techniques [31], while more recent work considers the impact of reliability [13] and relationships to process variation [25]. The management issues naturally evolved to 3D architectures, which exacerbate the thermal management problem [14][41]. Architectural techniques are complemented by efforts in the system software community primarily through managing power dissipation using various scheduling techniques [4][12][23]. The preceding are just a few examples of the extensive knowledge base developed in the past decade or so, and [15][24] provide a thorough overview of the techniques.

Our work addresses the impending challenge not addressed in these prior works: the consequences of thermally coupled behavior of heterogeneous cores that share the same die, and whose

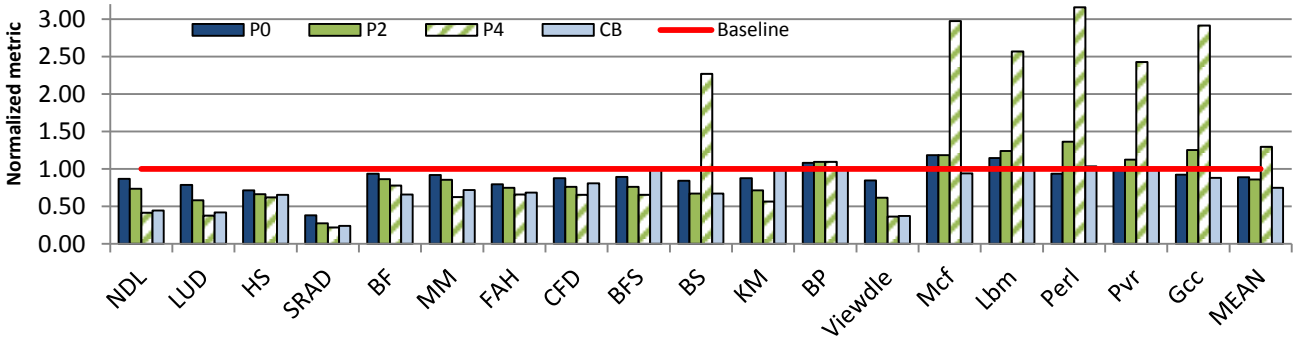

Figure 12: Energy-delay^2 product normalized to baseline.

performance is also coupled through applications that use both the CPU and the GPU. Some recent work includes efforts to couple thermal management, cooling management, and power management [6][7][35]. However, these efforts do not directly address architecturally coupled operation. Recent studies [3][5] have identified throughput-computing performance-coupled applications as an emergent class of future applications. Wang et al. [43] proposed power-efficient ways of workload partitioning this class of applications between CPU and GPU in heterogeneous systems. There has also been a number of works on dynamic power management of such applications [19][26][27][29][30]. However, these do not address effects of thermal coupling balanced with performance coupling.

The most relevant work is that of the Intel Sandy Bridge processor that introduces whole-chip thermal-based power management, recognizing the shared power and thermal headroom between the CPU and GPU [36]. A measurement-based budgeting process allocates this headroom between the CPU and the GPU. When cores are executing below the headroom, they acquire "energy credits" that are used to determine the new (boosted) power state for short durations. They recognize the dependency of performance between CPU and GPU and expose a software interface that can be used by the operating system or driver to specify how to partition the energy headroom between CPU and GPU. Our work differs from [36] by characterizing thermal coupling, noting and quantifying its negative interactions with unregulated boosting algorithms and proposing a solution that can be implemented to balance thermal and performance coupling effects dynamically.

Finally, unlike much past work in this area, we implement our algorithms on hardware and show measureable performance and power benefits when compared to a state-of-the-practice power-management algorithm.

8 CONCLUSIONS

This paper addressed the complex relationship among power, thermals, and performance in a heterogeneous system running diverse applications. We described and explored thermal entities with varying thermal signatures and demonstrated the relationship between thermal coupling and performance coupling through detailed empirical analysis. Based on our analysis, we proposed a cooperative boosting solution that balances the effects of thermal coupling with the requirements of performance coupling to determine the critical frequency of operation. We compared the CB algorithm with a state-of-the-practice boost algorithm and static power-limiting methods for a varied set of homogeneous and heterogeneous benchmarks. We showed on hardware that CB achieves an average 10% speed-up and an average 8% power reduction compared to the other algorithms, resulting in a 25% improvement in the ED^2 product.

We presented an initial assessment of performance-coupled applications and how to manage them dynamically. In the future, we plan to expand this work to manage the GPU directly in addition to the CPU, explore additional measures to detect performance coupling, and examine more complex usage scenarios.

9 ACKNOWLEDGMENTS

The authors gratefully acknowledge the constructive comments of the reviewers, which have improved the quality of the final manuscript.

10 REFERENCES

[1] Advanced Configuration and Power Interface (ACPI), Specification, http://www.acpi.info/spec.htm.

[2] AMD APP SDK, http://developer.amd.com/tools/heterogeneous-computing/amd-accelerated-parallel-processing-app-sdk/.

[3] M. Arora, S. Nath, S. Mazumdar, S. Baden, and D. Tullsen, "Redefining the Role of the CPU in the Era of CPU-GPU Integration," *IEEE Micro2012*.

[4] A. Arani et al., "Online thermal-aware scheduling for multiple clock domain CMPs," *ISOCC 2007*.

[5] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick, "The landscape of parallel computing research: A view from Berkeley," *Technical Report UCB/EECS-183*, 2006.

[6] R. Ayoub, R. Nath, and T. Rosing, "JETC: Joint Energy Thermal and Cooling Management for Memory and CPU Subsystems in Servers," *HPCA 2012*.

[7] Peter Bailis, V. J. Reddi, S. Gandhi**,** D. Brooks, and M. Seltzer, **"**Dimetrodon: Processor-level Preventive Thermal Management via Idle Cycle Injection," *DAC 2011*.

[8] BKDG:http://support.amd.com/us/Processor_TechDocs/42300_15h_Mod_10h-1Fh_BKDG.pdf.

[9] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," *HPCA 2001*.

[10] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, Lee S-H, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," *IISWC 2009*.

[11] S. Che, J. W. Sheaffer, M. Boyer, L. Szafaryn, and K. Skadron, "A characterization of the Rodinia benchmark suite with comparison to contemporary CMP workloads," *IISWC 2010*.

[12] J. Choi, C. Cher, H. Franke, H. Haman, A. Weger, and P. Bose, "Thermal-aware task scheduling at the system software level," *ISLPED 2007*.

[13] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Temperature management in multiprocessor SoCs using online learning," *DAC 2008*.

[14] A. K. Coskun, T. S. Rosing, D. A. Alonso, J. Leblebici, and J. Ayala, "Dynamic thermal management in 3D multicore architectures," *DATE 2009*.

[15] J. Cong, S. W. Chung, and K. Skadron, "Recent Thermal Management Techniques for Microprocessors," *ACM Computing Surveys 2012*.

[16] J. Donald and M. Martonosi, "Techniques for multicore thermal management: classification and new exploration," *ISCA 2006*.

[17] Folding At Home, http://folding.stanford.edu/English/Download

[18] V. Hanumaiah and S. Vrudhula, "Temperature- Aware DVFS for Hard Real-Time Applications on Multicore Processors," *IEEE Transactions on Computers 2012*.

[19] S. Hong and H. Kim, "An integrated GPU power and performance model," *ISCA 2010*.

[20] C. Hsu and W. Feng, "Effective dynamic voltage scaling through CPU-boundedness detection," *Lecture Notes in Computer Science 2004*.

[21] Z. Hu, D. Brooks, V. Zyuban, and P. Bose, "Microarchitecture-level power-performance simulators: modeling, validation and impact on design," *MICRO 2003*.

[22] W. Huang, M. Stan, K. Sankaranarayanan, R. Ribando, and K. Skadron, "Many-core design from a thermal perspective," *DAC 2008*.

[23] W-L. Hung, Y. Xie, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Thermal-Aware Allocation and Scheduling for Systems-on-a-Chip Design," *DATE 2005.*
[24] S. Kaxiras and M. Martonosi, "Computer Architecture Techniques for Power Efficiency," *Synthesis Lectures on Computer Architecture.*
[25] E. Kursun and C. Y. Cher, "Temperature Variation Characterization and Thermal Management in Multicore Architectures," *IEEE Micro 2009.*
[26] J. Lee and H. Kim, "TAP: A TLP-aware cache management policy for a CPU-GPU heterogeneous architecture," *HPCA 2012.*
[27] J. Lee, V. Sathish, M. Schulte, K. Compton, and N. Kim, "Improving throughput of power-constrained GPUs using dynamic voltage/frequency and core scaling," *PACT 2011.*
[28] O. Lempel, "2nd Generation Intel Core Processor Family: Intel Core i7, i5, and i3," *HotChips 2011.*
[29] J. Li and J. Martinez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," *HPCA 2006.*
[30] C. Luk, S. Hong, and H. Kim, "Qilin: Exploiting parallelism on heterogeneous multiprocessors with adaptive mapping," *MICRO 2009.*
[31] R. Mukherjee and S. O. Memik, "Physical aware frequency selection for dynamic thermal management in multi-core systems," *ICCAD 2006.*
[32] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. P. Boyd, L. Benini, and D. Micheli, "Temperature control of high-performance multi-core platforms using convex optimization," *DATE 2008.*
[33] S. Nussabaum, "AMD Trinity APU," *HotChips 2012.*
[34] M. D. Powell. M. Gomaa, and T. N. Vijaykumar, "Heat-and-run: leveraging SMT and CMP to manage power density through the operating system," *ASPLOS 2004.*
[35] A. Raghavan, Y. Luo, A. Chandawalla, M. Papaefthymiou, K. P. Pipe, T. F. Wenisch, and M. M. K. Martin, "Computational Sprinting," *HPCA 2012.*
[36] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weisman, "Power Management Architectures of the Intel Microarchitecture Code-Named Sandy Bridge," *IEEE Micro 2012.*
[37] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management," *HPCA 2002.*
[38] K. Skadron, M. R. Stan, W. Huang, S. Veluswamy, K. Sankaranrayan, and D. Tarjan, "Temperature-aware microarchitecture," *ISCA 2003.*
[39] K. Skadron, "Hybrid architectural dynamic thermal management," *DATE 2004.*
[40] The Standard Performance Evaluation Corporation (SPEC). Web resource, http://www.spec.org.
[41] C. Sun, L. Shang, and R. P. Dick, "Three-dimensional multiprocessor system-on-chip thermal optimization," *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2007.*
[42] Viewdle, http://viewdle.com/products/desktop/index.html
[43] H. Wang, V. Sathish, R. Singh, M. Schulte, and N. Kim, "Workload and power budget partitioning for single chip heterogeneous processors," *PACT 2012.*
[44] Windows Power Management Overview, http://www.microsoft.com/en-us/download/details.aspx?id=23878.
[45] J. A. Winter and D. Albonesi, "Addressing thermal non-uniformity in SMT workloads," *ACM TACO 2008.*
[46] F. Zanini, D. Atienza, and G. D. Micheli, "A control theory approach for thermal balancing of MPSoC," *ASP-DAC 2009.*
[47] http://www.amd.com/us/products/notebook/apu/ultrathin/pages/ultrathin.aspx#3, AMD A8 4555M.