# Adaptive Guardband Scheduling to Improve System-Level Efficiency of the POWER7+

Yazhou Zu[1], Charles R. Lefurgy[2], Jingwen Leng[1], Matthew Halpern[1],
Michael S. Floyd[2], Vijay Janapa Reddi[1]

[1] The University of Texas at Austin
{yazhou.zu, jingwen, matthalp}@utexas.edu, vj@ece.utexas.edu

[2] IBM
{lefurgy, mfloyd}@us.ibm.com

## ABSTRACT

The traditional guardbanding approach to ensure processor reliability is becoming obsolete because it always over-provisions voltage and wastes a lot of energy. As a next-generation alternative, adaptive guardbanding dynamically adjusts chip clock frequency and voltage based on timing margin measured at runtime. With adaptive guardbanding, voltage guardband is only provided when needed, thereby promising significant energy efficiency improvement.

In this paper, we provide the first full-system analysis of adaptive guardbanding's implications using a POWER7+ multicore. On the basis of a broad collection of hardware measurements, we show the benefits of adaptive guardbanding in a practical setting are strongly dependent upon workload characteristics and chip-wide multicore activity. A key finding is that adaptive guardbanding's benefits diminish as the number of active cores increases, and they are highly dependent upon the workload running. Through a series of analysis, we show these high-level system effects are the result of interactions between the application characteristics, architecture and the underlying voltage regulator module's loadline effect and IR drop effects.

To that end, we introduce adaptive guardband scheduling to reclaim adaptive guardbanding's efficiency under different enterprise scenarios. Our solution reduces processor power consumption by 6.2% over a highly optimized system, effectively *doubling* adaptive guardbanding's original improvement. Our solution also avoids malicious workload mappings to guarantee application QoS in the face of adaptive guardbanding hardware's variable performance.


*MICRO-48, December 05-09, 2015,* Waikiki, HI, USA
 ISBN 978-1-4503-4034-2/15/12 ...$15.00
DOI: http://dx.doi.org/10.1145/2830772.2830824

## Categories and Subject Descriptors

B.8 [**Hardware**]: Performance and Reliability; C.1.0 [**Processor Architectures**]: General

## Keywords

operating margin; di/dt effect; voltage drop; energy efficiency; scheduling

## 1. INTRODUCTION

Processor manufacturers commonly apply operating guardband to ensure that microprocessors operate reliably over various loads and environmental conditions. Traditionally, this guardband is a static margin added to the lowest voltage at which the microprocessor operates correctly under stress conditions. The static margin guarantees that the loadline, aging effects, fast noise processes and calibration error are all safely considered for reliable execution.

In recent years, many adaptive frequency and voltage control techniques have been developed to address the high amount of static margin [1, 2, 3, 4, 5, 6]. Such adaptive guardbanding aims at reducing the total margin to improve system efficiency while still ensuring processor reliability. However, the prior measurement studies do not present a comprehensive *system-level analysis* of how workload heterogeneity and core count impact the efficiency of a system using a processor with adaptive guardbanding capabilities.

This paper presents the first detailed, full-system characterization of adaptive guardbanding. Using measurements and running real-world workloads, we study the factors that affect adaptive guardbanding's behavior and the benefits it offers by characterizing its operation using POWER7+, an adaptive guardbanding multicore processor. Using a fully built production system, we systematically characterize the benefits and limitations of adaptive guardbanding in terms of multicore scaling and workload heterogeneity. In our analysis, we study adaptive guardbanding's undervolting and overclocking modes to fully characterize the system effects under different usage scenarios.

We find when only one core is active, adaptive guardbanding can efficiently turn the underutilized guardband into significant power and performance benefits while tolerating voltage swings. However, as more cores are progressively utilized by a multithreaded application, the benefits of adaptive guardbanding begin to diminish in both power and performance improvements. Using the processor's sensor-rich features, we systematically characterize and decompose the on-chip voltage drop that affects the adaptive guardbanding's efficiency into its different components, and analyze the root cause of the problem. Under heavy load, the IR drop across the chip and the voltage regulator module's (VRM) loadline effect limit adaptive guardbanding's ability to the point of almost no benefit.

The magnitude of the efficiency drop aforementioned, however, varies significantly from one workload to another. Thus, given the workload sensitivity of adaptive guardbanding, and the long-term nature of the observed effects, we introduce the notion of *adaptive guardband scheduling (AGS)*. The intent behind AGS is to compensate for adaptive guardbanding's inefficiencies at the system level. AGS can improve system efficiency by utilizing idle resources efficiently using a novel concept called "loadline borrowing". It can also guarantee the quality of service for critical workloads in datacenters by predicting the expected adaptive guardbanding effects of colocating any workloads together. We developed a lightweight MIPS-based prediction model for performing runtime scheduling at the middleware layer.

Our study is conducted on a POWER7+ system, one of the few commercial systems offering adaptive guardbanding, and therefore *our findings can serve as a fundamental step toward enabling more efficient and ubiquitous adaptive guardbanding in next-generation processors*. To this end, we make the following contributions:

- We characterize the benefits and limitations of adaptive guardbanding using a production server with respect to core scaling and workload variance.
- We measure and decompose the on-chip voltage drop to attribute the contribution of loadline, IR drop and di/dt noise to the system's (in)efficiency.
- We propose scheduling to opportunistically improve the power and performance benefits and predictability for adaptive guardbanding-based systems.

The remainder of the paper is structured as follows: Sec. 2 provides background for the POWER7+ architecture and its implementation of adaptive guardbanding. Sec. 3 characterizes adaptive guardbanding's limitations when scaling up the number of active cores under different workload scenarios. Sec. 4 analyzes the root cause of adaptive guardbanding's behavior as seen in the previous section. Sec. 5 proposes adaptive guardbanding scheduling to improve POWER7+'s efficiency when the load is light versus heavy. Sec. 6 compares our work with prior work, and Sec. 7 concludes the paper.

# 2. ADAPTIVE GUARDBANDING IN THE POWER7+ MULTICORE PROCESSOR

We introduce the POWER7+ processor and give an overview of its key features as they pertain to the work presented throughout the paper (Sec. 2.1). Next, we explain the processor's specific implementation of adaptive guardbanding (Sec. 2.2). Although adaptive guardbanding implementations can vary from one platform to another [7, 8, 1, 2, 3, 4, 5, 6], the general building blocks and principles largely remain the same.

## 2.1 The POWER7+ Multicore Processor

The POWER7+ is an eight-core out-of-order processor manufactured on a 32-nm process. It supports 4-way simultaneous multithreading, allowing a total of 32 threads to execute simultaneously on the system [9].

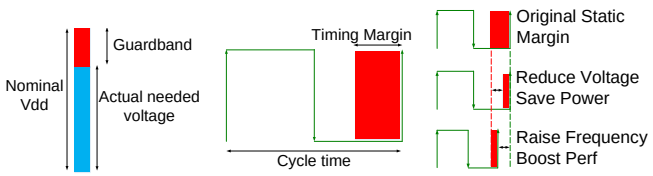
A POWER7+ processor has two main power domains, each with its own on-chip power delivery network (PDN). The $V_{dd}$ domain is dedicated to the logic circuits in the core and caches, and the $V_{cs}$ domain is dedicated for the on-chip storage structures [10, 11]. The PDNs are shared among all eight cores to reduce voltage noise [12].

The processor supports both coarse-grained and fine-grained power management. Coarse-grained power management includes per-core power gating to reduce idle power consumption. Fine-grained power management supports adaptive guardband management to enable dynamic trade-offs between higher clock frequencies and energy efficiency.

POWER7+ uses *adaptive guardbanding* to prevent circuit timing emergencies. Traditionally, chip vendors overprovision the nominal supply voltage with a fixed guardband to guarantee processor reliability under worst-case conditions, as shown in Fig. 1a. Under typical loads, the guardband results in faster circuit operation than required at the target frequency, resulting in additional processor cycle time, shown in Fig. 1b. In the event of a timing emergency caused by voltage droops, the extra margin prevents timing violations and failures by tolerating circuit slowdown. Although static guardbanding guarantees robust execution, it tends to be severely overprovisioned because timing emergencies occur infrequently, and thus it is less energy efficient.

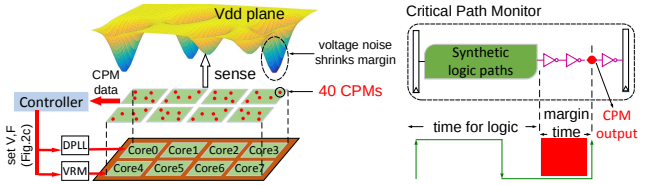
Instead of relying on the traditional static timing margin provided by the voltage guardband for reliability, the POWER7+ processor uses variable and adaptive cycle time to track circuit speed for a given voltage. In the event of a voltage droop, the processor slows down the cycle time to allow circuit operation to complete. Because voltage droops occur rarely, during normal operation the adaptive guardbanding mechanism eliminates a significant portion of the timing slack.

As shown in Fig. 1c, the reduced cycle time can be turned into either performance benefit by overclocking or energy benefit by undervolting the processor. Adaptive guardbanding can significantly reduce the magnitude of the voltage guardband required for reliability. In the POWER7+, as much as 25%




(a) Guardband. (b) Static margin. (c) Adaptive margin.

Figure 1: Voltage guardband ensures reliability by creating extra timing margin. Adaptive guardbanding relaxes the requirement on the guardband and improves system efficiency by overclocking or undervolting.




(a) Control loop overview. (b) CPM behavior.

Figure 2: Interactions among CPMs, DPLLs, and VRMs to guarantee reliability and improve efficiency in POWER7+. CPM measures the timing margin and the controller adjusts voltage and frequency accordingly.

of the static guardband can be eliminated using adaptive guardbanding. The remaining guardband is present as a precautionary measure to tolerate nondeterministic sources of error in the adaptive guardbanding mechanism itself [13].

## 2.2 Adaptive Guardbanding Implementation

We briefly review how adaptive guardbanding works in the POWER7+ [2, 14, 13]. Fig. 2a shows an overview of the feedback loop for adaptive guardbanding control. The system relies on three key components: (1) critical path monitor (CPM) sensors to sense timing margin [15, 16]; (2) digital phase locked loops (DPLLs) to quickly and independently adjust clock frequency per core based on CPM readings [17]; and (3) hardware and firmware controllers that decide when and how to leverage the benefits from a reduced guardband.

POWER7+ has 40 CPMs distributed across the chip to provide chip-wide, cycle-by-cycle timing margin measurement. Each core has 5 CPMs placed in different units to account for core-level spatial variations in voltage noise and critical path sensitivity. Detailed characterization of CPM placement, calibration, and sensitivity is provided in [13].

A CPM uses synthetic paths to mimic different logical circuits' behavior and a 12-bit edge detector to quantify the amount of timing margin left. Fig. 2b illustrates the CPM's internal structure. On each cycle, a signal is launched through the synthetic paths and into the edge detector. When the next cycle arrives, the number of delay elements the edge has propagated through in the edge detector corresponds to the CPM output. A CPM outputs an integer index from 0–11, which corresponds to the position of the edge in the edge detector.

In the POWER7+ processor, during guardband calibration the different CPMs are calibrated to output a target value. When the output is less (toward zero), the timing margin has been reduced from the calibrated point. Likewise, when the output is more (toward 11), the available timing margin has increased.

Per-core DPLL frequency control lets the processor tolerate transient voltage droops by reducing clock frequency for each core with no impact on other cores. The DPLLs can rapidly adjust frequency, as fast as 7% in less than 10 ns, while the clock is still active; thus, the processor can tolerate transient voltage droops. Every cycle, the lowest-value CPM in each core is compared against the calibration position. In response, the DPLL will slew the clock frequency up or down to control the timing margin to the calibrated amount.

POWER7+ supports two modes to convert the excess timing margin into either a performance increase by overclocking or power reduction by undervolting. In the overclocking mode, the CPM and DPLL hardware form a closed-loop controller. At the fixed nominal voltage, the DPLL continuously adjusts frequency on the basis of the CPM's timing sense to operate at the calibrated timing margin. Under light loads, clock frequency can be boosted by as much as 10% compared to when adaptive guardbanding is off. In the undervolting mode, the firmware observes CPM-DPLL's frequency and over a longer term (32ms) adjusts voltage to make clock frequency hits the target. In this case, the performance benefit from the CPM-DPLL can be turned into an energy-saving benefit.

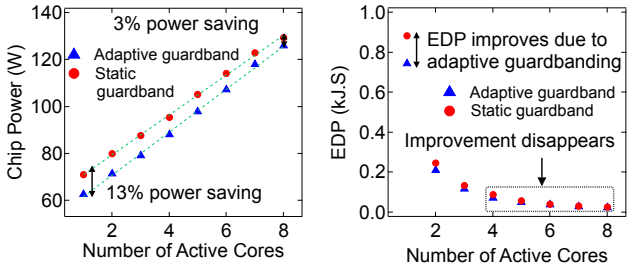
# 3. EFFICIENCY ANALYSIS OF ADAPTIVE GUARDBANDING ON MULTICORE

The benefits of reducing guardband have been explored in the past at the circuit- [1, 3, 4, 5, 6] and architecture levels [2, 18, 19, 20], and much less at the system level [21, 22]. Most of the prior work focuses on homogeneous workloads under high utilization. Our work is the first attempt at understanding the efficiency of adaptive guardbanding on a multicore system, specifically as the system activity (i.e., core usage) begins to increase using real workloads.

Using an enterprise class server (Sec. 3.1), we characterize the efficiency of adaptive guardbanding at the system level. In particular, we measure, analyze and characterize the mechanism's effectiveness under different architectural configurations and workload characteristics. We make two fundamentally new observations about the effectiveness of adaptive guardbanding on a multicore system. First, the efficiency of adaptive guardbanding can diminish as the number of active cores increases (Sec. 3.2). Second, the inefficiency is highly subject to workload characteristics (Sec. 3.3).

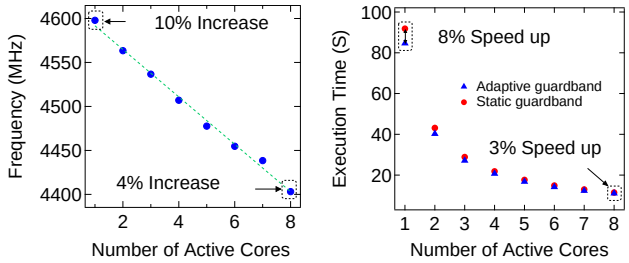
## 3.1 Experimental Infrastructure

We perform our analysis on a commercial IBM Power




(a) Power saving. (b) Energy reduction.

Figure 3: Adaptive guardbanding can save power effectively. However, the benefits decrease as more cores are used to actively run the application.




(a) Frequency-boosting mode. (b) Execution time.

Figure 4: Adaptive guardbanding can improve performance by increasing frequency. However, the overclocking benefits decrease as more cores are used.

720 Express server (7R2) that has two POWER7+ processors on the motherboard. The processors share the main memory and other peripheral resources, such as storage and network. We focus on one of the two processors, although we validated our conclusions by conducting experiments on the other processor as well. Unless stated otherwise, the first processor is configured to idle and runs background tasks. The system runs RedHat Enterprise Linux, configured with 32 GB RAM.

We use PARSEC [23] and SPLASH-2 [24, 25] in this section because they are scalable workloads and we need to the control the applications' parallelism to carefully study the impact of core scaling. The workloads are compiled using GCC with `-O2` optimization.

We characterize the efficiency of adaptive guardbanding across two modes of operation: 1) undervolting to reduce power consumption and 2) overclocking to boost performance. Hooks in the firmware let us place the system in either operating mode. The hardware and firmware autonomously select frequency and voltage depending on the configured operation mode.

## 3.2 Core Scaling

Using raytrace from PARSEC (as an example), we show adaptive guardbanding's impact on chip power. We study both average chip power consumption and total CPU energy savings using Fig. 3. We find that adaptive guardbanding is always effective at improving performance or lowering power consumption. However, it cannot always scale up efficiently with more cores.

Fig. 3a shows the program's power consumption as we use more cores, i.e., more threads to process the workload. We measure the microprocessor $V_{dd}$ rail power by reading physical sensors available on the server, which represents most of the total processor power. In undervolting mode, adaptive guardbanding turns the unused guardband into energy savings by scaling back the voltage, which reduces unnecessary power consumption. When one core is active and the others are idle, adaptive guardbanding reduces the average power consumption by 13% compared to no adaptive guardbanding.

Although adaptive guardbanding always saves power, a more important and crucial observation from Fig. 3a is the decreasing power-saving trend as the number of active cores increases in the system. The power improvement from adaptive guardbanding decreases as the parallelism in the workload is (manually) increased, forcing the usage of the additional cores. Although adaptive guardbanding can save as much as 13% power when only one core is active, the savings drop sharply to about 3% when the activity scales up to eight cores.

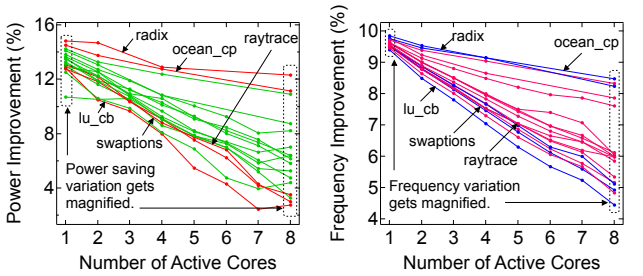
When examining the workload's overall energy-delay product (EDP), Fig. 3b shows notable energy efficiency improvement when only a small set of cores is actively processing the workload. However, beyond four cores, the improvement drops significantly. When only one core is active, processor energy efficiency improves by as much as 20% compared to using a static guardband. But the additional improvement beyond activating more than four cores becomes negligible.

Our observations hold true for frequency-boosting as well. Adaptive guardbanding's ability to boost frequency decreases as core counts increase. Fig. 4 shows experimental results for lu_cb from the SPLASH-2 benchmark suite. Compared to using a fixed target frequency of 4.2GHz under a static guardband, adaptive guardbanding can achieve substantial frequency improvement, as shown in Fig. 4a. When only one core is actively processing the workload, frequency increases by up to 10% compared to the static guardband baseline. However, when all eight cores are running the workload the frequency gain drops to only 4%.

Frequency improvement turns into program execution time speedup, especially for computing-bound workloads. For lu_cb the execution speedup varies gradually, decreasing from 8% when only one core is used to 3% when all cores are running the workload. This trend of diminishing benefit as core count scales up is similar to what we observe when the extra guardband is turned into energy savings for this workload.

## 3.3 Workload Heterogeneity

Variations in workload activity (i.e., heterogeneity) are known to strongly impact system performance from cache performance to bandwidth utilization. In this section, we demonstrate workload heterogeneity also




(a) Power-saving mode. (b) Frequency-boosting mode.

Figure 5: Improvements reduce at different rates for each of the PARSEC and SPLASH-2 workloads when cores are progressively activated, leading to magnified workload variation when all cores are active.

impacts adaptive guardbanding's runtime efficiency. We focus our analysis on the architecture-level observations and later in Sec. 4 we explore the causes for the observed behaviors.

Fig. 5 shows the results for power and frequency improvement for all PARSEC and SPLASH-2 workloads compared to the same number of cores active when adaptive guardband is disabled. The improvements are with respect to the system using a static guardband. The results are from two experiments, one in which the control loop is operating in energy-saving mode (Fig. 5a) and the other in which it is operating in frequency-boosting mode (Fig. 5b). Each line in both figures corresponds to one benchmark.

From Fig. 5a and Fig. 5b, we draw four conclusions. First, adaptive guardbanding consistently yields improvement, regardless of its operating mode and workload diversity. Across all of the workloads, adaptive guardbanding reduces power consumption somewhere between 10.7% and 14.8% and improves processor clock frequency by as much 9.6% on average, when one core is active. Even when all eight cores are active, improvements are at least above 4%. Power-saving improvements are slightly larger than frequency improvements because of the quadratic relationship between voltage scaling and power, as opposed to the linear relationship between frequency and power.

Second, the improvements monotonically decrease as the number of active cores increases. Across all the workloads, we observe a consistent drop in adaptive guardbanding's efficiency. The average power efficiency improvement across the workloads drops from 13.3% when one core is active to 10% when two cores are active to 6.4% when all cores are actively processing the workload. We observe a similar trend with frequency.

Third, the rate of monotonic decrease for each workload varies significantly. For instance, radix's power improvement drops from 15% when one core is active to around 12% when all eight cores are active. However, in swaptions, the improvement drops drastically from 13% to 3%. In the frequency-boosting mode, the decreasing magnitude is slightly smaller, although the variation in improvements is still strongly present. Frequency for radix and ocean_cp almost remains unchanged at 9%, but the frequency of lu_cb, swaptions and raytrace drops notably from 10% to 4%.

Fourth, regardless of the adaptive guardbanding operating mode (i.e., power saving or frequency boosting), workload heterogeneity significantly impacts the mechanism's efficiency when all cores are active. This finding is especially important in the context of enterprise systems, because server workloads are ideally configured to fully use all computing resources to reduce the operator's total cost of ownership (TCO) [26].

In multicore systems that rely on adaptive guardbanding, the system's behavior will vary significantly depending on how many cores are being used and what workloads are simultaneously coscheduled for execution on the processor. To prove this point, we later discuss the implications of workload coscheduling using our system. In the future, we suspect workload heterogeneity could be a major source of inefficiency, especially as we integrate more cores into the processor, unless we identify the problem's source for mitigation.

# 4. ROOT-CAUSE ANALYSIS OF ADAPTIVE GUARDBANDING INEFFICIENCIES

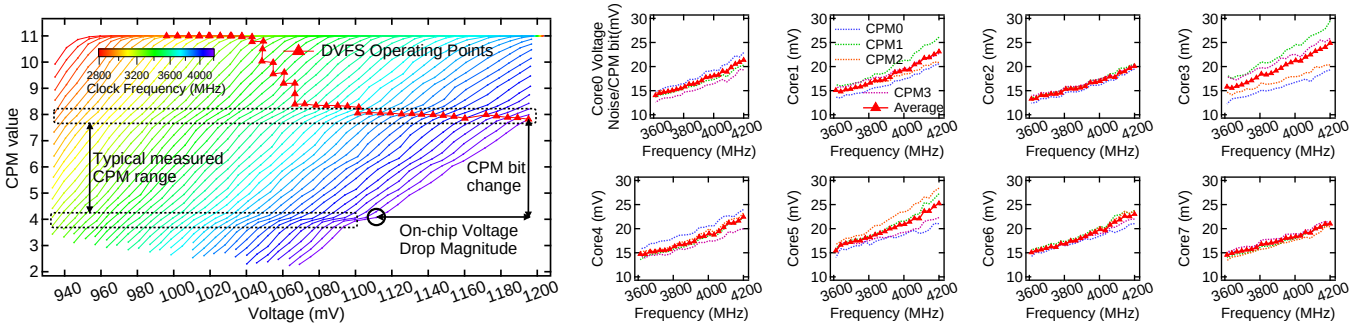
In this section, we analyze the root cause of adaptive guardbanding's inefficiency under increasing core counts and workload heterogeneity to understand how to reclaim the loss in efficiency. We present an approach for characterizing adaptive guardbanding's inefficiency using CPM sensors (Sec. 4.1). On this basis, we characterize the voltage drop in the chip across both core counts and workloads because the on-chip voltage drop affects adaptive guardbanding's efficiency. Our analysis reveals that core count scaling results in a large on-chip voltage drop (Sec. 4.2), whereas workload heterogeneity plays a dominant role in affecting the processor's IR drop and loadline (Sec. 4.3).

## 4.1 Measuring the On-chip Voltage Drop

We developed a novel approach to capture and characterize adaptive guardbanding's behavior using CPMs. We use CPM output to capture the on-chip voltage drop that affects the timing margin, which in turn affects the adaptive guardband's efficiency. In effect, we use CPMs as "performance counters" to estimate on-chip voltage, similar to how performance counters were first shown to be useful for predicting power consumption [27, 28].

Because timing margin is determined by on-chip voltage, capturing the CPM's output would reflect the transient voltage drops between the VRM output and on-chip voltage. Low on-chip voltage leads to less time for the CPM's synthetic-path edge to propagate through the inverter chain, and thus the CPM will yield a low output value. Under high on-chip voltage, the circuit runs faster, and the CPM yields a higher output.

To read the CPMs, we disable adaptive guardbanding because it dynamically adjusts the timing margin to keep the margin small and CPMs constant. The CPMs typically hover around an output value of 2 when adaptive guardbanding is active due to CPM




(a) Mapping between on-chip voltage and CPM values.

(b) The CPMs' sensitivity toward supply voltage in each core.

Figure 6: CPMs can sense the chip supply voltage with a precision of about 21mV per CPM bit at peak frequency.

calibration. By disabling adaptive guardbanding, we allow the CPMs' output values to "float" in response to on-chip voltage fluctuations, and thus we can study how supply voltage affects the behavior of CPMs.

We use the IBM Automated Measurement of Systems for Temperature and Energy Reporting software (AMESTER) [29] to read the CPMs' output. We record CPM readings under different on-chip voltage levels to determine how CPM responds to different on-chip voltage. AMESTER reads the CPMs at the minimal sampling interval of 32ms, which is restricted by the service processor. AMESTER can read the CPMs in either *sticky mode* or *sample mode*. In sticky mode, AMESTER reads the worst-case, i.e. smallest, output of each CPM during the past 32 ms, which is useful for quantifying worst-case droops. In sample mode, AMESTER provides a real-time sample of each CPM, which is useful for characterizing normal operation.

We use CPMs in sample mode to convert their output into on-chip voltage. To minimize experimental variability, we let the operating system run and throttle each core to fetch one instruction every 128 cycles. Fig. 6 shows the mapping between CPM output and on-chip voltage. In Fig. 6a, we sweep the voltage range for all possible clock frequencies and look at the average output of all 40 CPMs over 12,500 samples, which corresponds to about 1 minute of measurement. Each line corresponds to one frequency setting, and the system default voltage levels at DVFS operating points are highlighted with the marked line. Starting from 2.8 GHz, each diagonal line, as we move to the right, corresponds to a 28 MHz increase in frequency. The rightmost line corresponds to the peak frequency of 4.2 GHz. For any one frequency, the CPM value gets smaller as we lower the voltage, confirming the expected behavior that smaller voltages correspond to less timing margin. Also, for a fixed voltage ($x$-axis), higher frequency yields smaller CPM values ($y$-axis) because of less cycle time and a tighter timing margin.

Fig. 6a lets us establish a direct relationship between CPM and on-chip voltage. We observe a near-linear relationship between the two variables under each frequency. Therefore, with a linear fit, we can determine each CPM bit's significance. On average, one CPM output value corresponds to 21 mV of on-chip voltage. On this basis, we can estimate the magnitude of on-chip voltage drop during any 32 ms interval. For instance, if the measured CPM output drops from eight to four, the estimated on-chip voltage has dropped by 84 mV.

Fig. 6b shows the sensitivity of the CPMs within each processor core. Although we see a near-linear relationship between frequency and all the CPMs, there is variation among the CPMs in each core and between cores. For instance, CPMs in Core 2, 6, 7 have steadier sensitivity compared to Core 1, 3, 5. The latter have higher distribution across CPMs. We attribute this behavior to process variation and CPM calibration error, as explained by prior work [13].

To ensure the robustness of our measurement results, we considered both repeatability and temperature effects. We repeated our experiment on another socket in the same server, and the result conforms to the same trend shown in Fig. 6a. We observe that chip temperature varies between 27°C at the lowest frequency to 38°C at the highest. Internal benchmark runs show such temperature variation does not have significant influence over CPM readings, and thus we can draw general conclusions from Fig. 6a.

## 4.2 On-chip Voltage Drop Analysis

Using our on-chip voltage drop measurement setup, we quantify the magnitude of the on-chip voltage drop to explain the general core scaling trends seen in Sec. 3. It is important to understand what factors, and more importantly how those factors, impact the efficiency of adaptive guardbanding as more cores are activated.

Fig. 7 shows the measured results for the voltage drop across different cores within the processor, ranging from Core 0 through Core 7. The cores are spatially located in the same order as they appear on the physical processor [10]. The $y$-axis is the percentage of on-chip voltage drop from the nominal. Given the magnitude of voltage drop and knowledge about the system's nominal operating voltage, we can determine the percentage change. The $x$-axis indicates the total number of simultaneously active cores, specifically as they are activated in succession from core 0 to 7. Keeping consistent with Fig. 5, each line in the

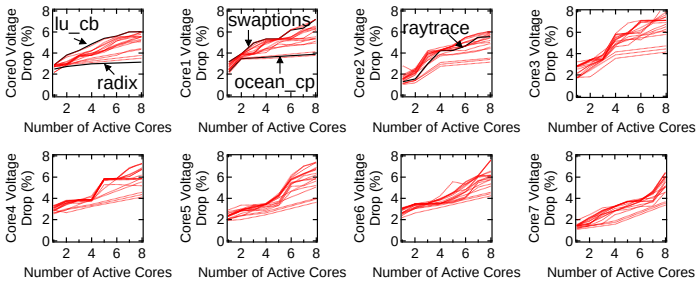



Figure 7: On-chip voltage drop analysis across cores under different workloads.

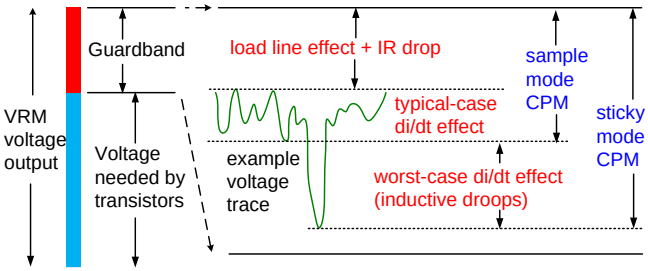



Figure 8: Voltage drop component analysis, including $di/dt$ droop, IR drop and the loadline effect.

subplots corresponds to one workload from PARSEC and SPLASH-2. Each subplot shows a particular core's characteristics with respect to every other (active or inactive) core in the processor.

Fig. 7 lets us understand several important factors that affect adaptive guardbanding's efficiency. First, voltage drop increases as more cores are activated. For all workloads, voltage drop increases from about 2% to 8% as the number of active cores increases. The trend is similar to the diminishing benefits seen previously in the power and frequency improvement in Fig. 5. As the magnitude of voltage drop increases, timing margin decreases and thus adaptive guardbanding's efficiency decreases at higher loads.

Second, the increasing on-chip voltage drop trend manifests as chip-wide global behavior because voltage drop affects all cores at the same time, regardless of whether they are idling or actively running a workload. For instance, when cores on the upper row (Core 0 through Core 3) are actively running a workload, they experience voltage drop. Meanwhile, cores in the bottom row also experience voltage drop even though Core 4 through Core 7 are not running any workloads.

The implications of the second finding are that global effects, such as chip-wide $di/dt$ noise [30, 31, 22] and off-chip IR drop, can affect adaptive guardbanding's system-wide power-saving efficiency because adaptive guardbanding makes decisions on the basis of the worst-case behavior of all cores. In particular, this behavior impacts the power-saving mode because the processor has a single off-chip VRM that will need to supply the highest voltage to match the most demanding core's voltage requirement. So, even if some cores are lightly active, the system may have to forgo their adaptive guardbanding benefits to support the activity of the busy core(s). In applications where workload imbalance exists, this can become a major efficiency impediment.

Third, the on-chip voltage drop's scaling trend, as the number of active cores increases, tends to differ across cores, indicating that voltage drop has localized behavior in addition to the global behavior described previously. For instance, across all the cores the magnitude of voltage drop shifts upward significantly whenever that particular core is activated. For instance, Core 7's voltage drop increases by 2% when it is activated, as evident in Core 7's voltage drop plot.

More generally, cores that are activated earlier have a higher voltage drop at first, and thereafter their voltage drop begins to saturate and plateau. For instance, Core 0 and Core 1 have a higher voltage drop when Core 0 through Core 3 are activated. These cores' voltage drop increase quickly when the number of active cores is less than four. On the contrary, the voltage drop for Core 4 through Core 7 does not change much while Core 0 through Core 3 are activated, but thereafter their voltage drop increases much more quickly.

Localized effects impact the operation of the per-core frequency-boosting mode. Each POWER7+ core has its own DPLL that can dynamically perform frequency scaling to improve performance when required. However, each core's performance can be boosted only when it is not affected by activity on its neighboring cores. In general, our observations imply that it is easier to boost clock frequency and, hopefully, performance – at least for computing-bound workloads – over reducing voltage, because frequency-boosting is largely affected by localized voltage drop. By comparison, the global voltage drop typically tends to have a more pronounced effect on the chip-wide power-saving mode.

## 4.3 Decomposing the On-chip Voltage Drop

To understand how workload heterogeneity affects the power-saving and frequency-boosting modes when all cores are active, we must understand why the on-chip voltage drop varies significantly from one workload to another with an increasing number of cores. For example, in Fig. 7 lu_cb's voltage drop increases more quickly compared to radix, whose voltage drop does not change much as the number of active cores increases. We decompose the on-chip voltage drop into its three primary components (see Fig. 8): worst-case $di/dt$ noise, also called voltage droops due to sudden current surges caused by microarchitecture activities; typical-case $di/dt$ noise due to regular current ripples; and passive voltage drop due to IR drop across the PDN and the loadline effect [2] at the VRM.

We use a mixture of current sensing techniques and CPM measurements to decompose the voltage drop. To measure passive voltage drop (i.e., loadline effect + IR drop), we use VRM's current sensors. The IR drop and

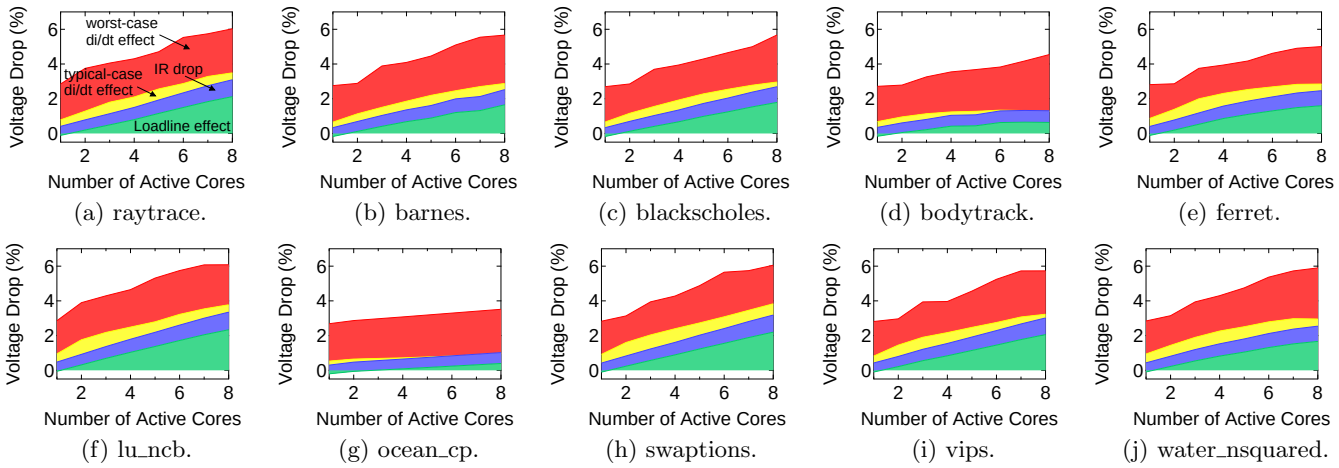



Figure 9: Different components of on-chip voltage drop for some PARSEC and SPLASH-2 benchmarks. In general, as more of the processor's cores are activated, voltage drop increases by varying magnitudes across workloads.

loadline effects are quantified using a heuristic equation verified against hardware measurements. The input to the equation is the current going from the VRM into the POWER7+ processor, sampled periodically.

We use CPMs to calculate the magnitude of typical and worst-case voltage noise. To get the typical $di/dt$ value, we run the CPMs in sample mode to acquire an immediate CPM reading, and after converting the CPM output into voltage, we subtract the passive component from it. To get the worst-case $di/dt$ value, we run the CPMs in sticky mode to acquire the largest voltage droop seen in the past 32 ms and subtract it from the long-term average measured in sample mode.

We select several representative benchmarks from previously discussed data and decompose their on-chip voltage drop into $di/dt$ noise and passive drop in Fig. 9. The subplots are in the form of a stacked area chart, showing the trend as more cores are progressively activated. Only Core 0 data simplifies the presentation of our analysis, although we have verified that the conclusions described in the following paragraphs hold true for the other cores as well.

By analyzing the data, we conclude that passive voltage drop, including IR drop across PDN and VRM's loadline is the dominant factor contributing to increasing voltage drop. Intuitively, these two passive effects have the most direct influence over adaptive guardbanding's behavior because they always exist steadily during execution as compared to $di/dt$ noise.

As we scale the number of active cores, the worst-case $di/dt$ noise increases slightly across all of the benchmarks, and typical-case $di/dt$ noise decreases. For instance, the worst-case $di/dt$ noise growth is noticeable in bodytrack, vips and water_nsquared. When multiple cores are active simultaneously, they can have synchronous behavior, or random alignment, that can cause large and sudden current swings leading to voltage droops [21, 31, 32]. However, our droop frequency analysis (not shown here) indicates that such large worst-case droops occur infrequently. On the contrary, typical-case $di/dt$ noise gets smaller when core count scales. With more active cores, microarchitectural activities stagger among different cores, which can lead to noise smoothing [31, 21].

Compared to $di/dt$ noise, we find a clear scale-up trend of passive voltage drop from Fig. 9, and it contributes most to the scale-up of total voltage drop. IR drop and loadline effects increase almost linearly with the number of active cores because the passive voltage drop is caused by processor current draw, which is further determined by chip power. When more cores are used, the whole chip consumes more dynamic power and will lead to higher IR drop and loadline effects.

Because adaptive guardbanding can deal with occasional $di/dt$ voltage droops by slowing down frequency quickly, the rare voltage drop caused by this effect does not strongly influence the power-saving and frequency-boosting capability of adaptive guardbanding, even though they consume a significant portion of the total voltage guardband. Thus, we believe passive voltage drop is the main source of impact to adaptive guardbanding's efficiency.

We confirm that loadline and IR drop cause adaptive guardbanding's inefficiency at full load by quantifying the relationship between their voltage drop under static guardbanding with respect to the system's two optimization modes: power saving (i.e., undervolting) and frequency boosting (i.e., overclocking). Fig. 10 shows the causal relationship between workload power consumption, loadline and IR drop, and the adaptive guardbanding's two modes. To ensure we have enough data points, we consider 27 SPECrate workloads on top of the existing 17 PARSEC and SPLASH-2 workloads used before. Each point represents the data we experimentally measured for one benchmark.

In Fig. 10, across all the subfigures, we see a strong correlation between passive voltage drop and the power-saving and frequency-boosting modes. Fig. 10a shows a

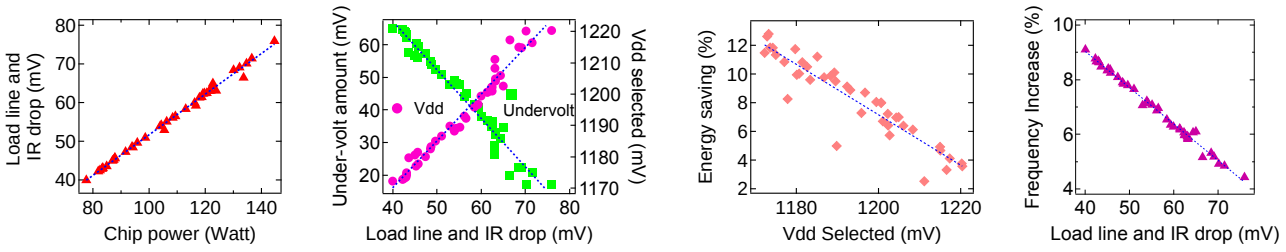



Figure 10: Power-intensive workloads induce large loadline and IR drop, which severely limits the adaptive guardbanding system's undervolting capability, and thus impacts the system's overall power-saving potential.

strong linear relationship between power and passive voltage drop. Fig. 10b shows when a workload has a high loadline and IR drop, the voltage guardband is highly utilized, and so adaptive guardbanding has less room for undervolting. Thus, the voltage selected by adaptive guardbanding is higher. The result is fewer energy savings for high-power workloads, as the data in Fig. 10c demonstrates. The same holds true for adaptive guardbanding's frequency-boosting mode. Here as well, a high loadline and IR drop reduce the timing margin; thus, the DPLL has limited room left to overclock the frequency as shown in Fig. 10d.

## 5. ADAPTIVE GUARDBAND SCHEDULING

We propose system-level scheduling techniques to improve the benefits of adaptive guardbanding. Our scheduler's overarching goal is to minimize the impact that loadline and IR drop have on an adaptive guardbanding processor's power and performance efficiency. We demonstrate *adaptive guardband scheduling* (AGS) in the context of two enterprise scenarios, as it pertains to real-world datacenter operations in which POWER7+ systems are deployed: one in which the system is not fully utilized and has idle computing resources (Sec. 5.1), and one in which the system is highly utilized and has some critical workload (e.g., latency-sensitive applications like WebSearch), and whose performance must be at some quality-of-service level to avoid service-level agreement violations (Sec. 5.2). We use these two scenarios to demonstrate that adaptive guardbanding has fundamentally new implications for how workloads are managed by the operating system or job schedulers.

### 5.1 Loadline Borrowing

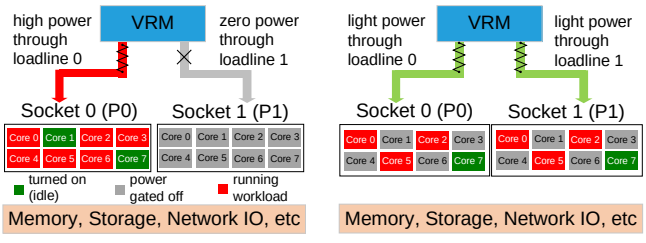
In a multi-socket server, conventional wisdom says to consolidate workloads onto fewer processors so that the idle processor can be shut down to eliminate wasted power [33, 34, 35]. However, this principle does apply to servers with adaptive guardbanding and per-core power-gating capability. Our measured results show consolidation actually leads to higher power o these systems. To this end, we propose loadline borrowing to maximize adaptive guardbanding's power-saving benefits for the underlying processors. Compared to workload consolidation, loadline borrowing achieves up to 12% power savings.

#### 5.1.1 Solution for Recovering Multicore Scaling Loss

We use Fig. 11 to introduce how loadline borrowing optimizes workload distribution among a server's VRM-multiprocessor subsystem. In Fig. 11, multiple processor sockets share a common VRM chip, each with its own power delivery path from the VRM to the die. The VRM can generate multiple $V_{dd}$ levels for different processors, which is normal for contemporary systems. In the following discussion, we use Fig. 11a and Fig. 11b to analyze the scenarios of workload consolidation and loadline borrowing and highlight the necessity of considering VRM's role in systems with adaptive guardbanding processors. Other components such as memory chips and disks are powered on steadily throughout our analysis.

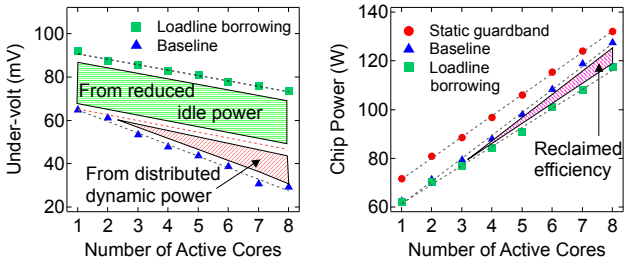
Fig. 11a shows a traditional consolidation schedule for a multisocket server. Workloads are all mapped to socket 0 so that socket 1 can be shut down. Because all power goes to socket 0, the passive voltage drop along the power-delivery path from VRM to processor 0 is very high, which limits adaptive guardbanding's potential to undervolt.

Loadline borrowing balances workloads equally among all available sockets, and power gates off unneeded cores to eliminate idle power consumption. Fig. 11b illustrates a loadline-borrowing schedule. In Fig. 11b active cores are distributed to each socket




(a) Workload consolidation. (b) Loadline borrowing.

Figure 11: Loadline borrowing balances workloads across multiple sockets to reduce per-socket voltage drop and create room for adaptive guardbanding.




(a) Undervolt scaling. (b) Power scaling.

Figure 12: Distributing raytrace across two processors reduces passive voltage drop, allowing more power saving under high core count.

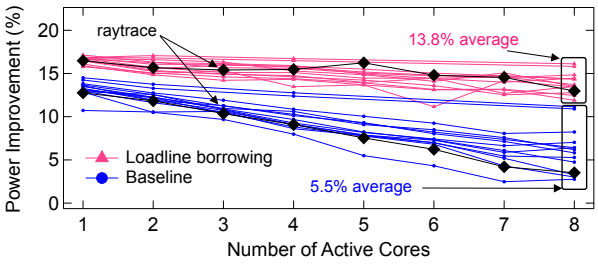



Figure 13: Loadline borrowing's power and energy improvement under different numbers of active cores. Compared to the baseline, loadline borrowing consistently shifts up every workload's power improvement.

evenly, and each socket power gates off a set of unused cores to achieve the same idle power elimination effect as in a consolidated schedule. In this schedule, each socket draws less power, reducing the passive voltage drop each processor experiences. This allows adaptive guardbanding to reduce more voltage from each processor and hence improve total processor power.

We use our two-socket platform to illustrate the benefits of loadline borrowing. We compare the case of conventional workload consolidation, which places all loaded cores on one processor as the baseline, to loadline borrowing, which balances the loaded core count across both processors. In this scenario, we keep eight of the total 16 cores turned on to respond instantly to utilization levels of up to 50%. The remaining eight cores are assumed to be not instantly needed, and therefore are put into a deep sleep (power-gated) state. We run the workload using one to eight cores. In the conventional case, all of the turned-on cores reside on a single processor. In the loadline borrowing case, each processor has four cores that are turned on and active. In either case, we measure and compare the two processors' total chip power.

As an example, Fig. 12 shows the results for raytrace with loadline borrowing. Fig. 12a shows that loadline borrowing offers a better undervolting benefit no matter how many cores are used. There are two reasons. First, loadline borrowing lets each processor power on fewer cores, which cuts down leakage power, and thus substantially reduces the idle power. For raytrace, less idle power gives 20mV more undervolting benefit when one core is active. Second, balancing application activity (threads) and system requirements (idle cores) across the processors' loadline distributes dynamic power across each processor, which further reduces the passive drop for each processor. When eight cores are active, reduced dynamic power allows an additional 20mV reduction.

Fig. 12b shows loadline borrowing can reduce a significant amount of total chip $V_{dd}$ power. The biggest effect is achieved when more cores are used. In Fig. 12b loadline borrowing reduces power consumption by 1.6%, 4.2% and 8.5% when two, four and eight cores are used, respectively. The result is intuitive because each processor's passive voltage drop is reduced when fewer cores are active. Thus, distributing the workload when more cores are active yields larger benefits.

For now, our loadline-borrowing proposal is suitable only for workload scheduling within a multisocket server. In this setting, all other resources, such as memory, disk and network I/O, remain active when workloads are consolidated onto a few processors. When workloads are consolidated across multiple servers, the idle power reduction from turning off the used memory and hard drive outweighs adaptive guardbanding's processor power savings. In this case, the scheduler will consolidate workloads onto fewer servers first, then on each server loadline borrowing can be used to further improve cluster power consumption. We leave this discussion to future studies.

### 5.1.2 *Evaluation of Loadline Borrowing*

Current operating systems are unaware and do not incorporate loadline knowledge into process scheduling. Therefore, we use the Linux kernel's taskset affinity mechanism to emulate a schedule that dynamically performs loadline borrowing. We evaluate loadline borrowing on a wider set of benchmarks including all of PARSEC and SPLASH-2 workloads to capture the general trends. Briefly, the key highlight is that loadline-aware OS-level software scheduling can effectively *double the efficiency* of adaptive guardbanding at high core counts.

Fig. 13 shows adaptive guardbanding's scaling power improvement against static guardbanding under workload consolidation and loadline borrowing. Ideally, adaptive guardbanding's power improvement will not scale down, and it will be identical across workloads. Loadline borrowing approaches this goal by increasing adaptive guardbanding's power-saving capability for all active cores, shown by the clustered lines at the top of the figure. When fewer cores are active, loadline borrowing's power improvement comes mainly from the reduced idle power on each processor. The improvement increases when more cores are active because each chip's dynamic power also reduces when the workload is distributed. Fig. 13 shows that on average consolidated adaptive guardbanding achieves 5.5% power improvement over static guardbanding when eight cores are active, whereas loadline borrowing

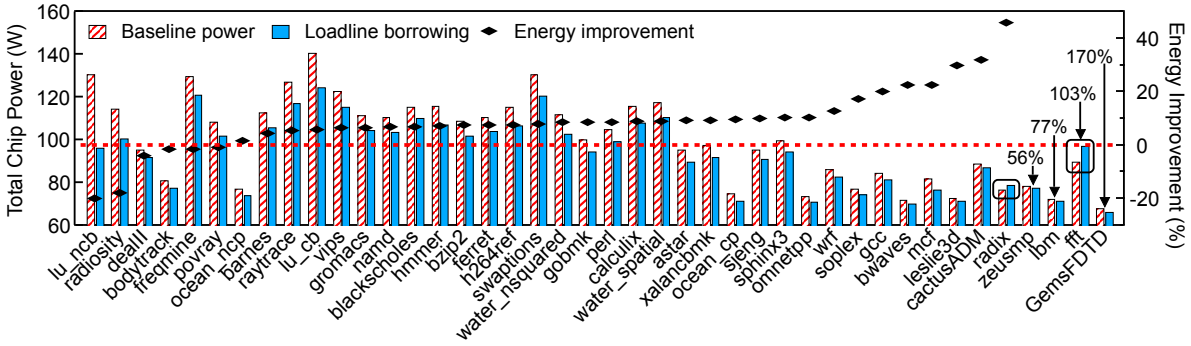



Figure 14: Loadline borrowing's power and energy improvement when eight cores are active.

improves by 13.8%, over 50% improvement atop the original system design.

We study more benchmarks along with PARSEC and SPLASH-2, including SPEC CPU 2006 workloads running in the form of SPECrate [36], to further demonstrate loadline borrowing's power and energy improvement when all eight cores are active. SPECrate is commonly used to measure system throughput, typical of evaluating performance when running different tasks simultaneously. In this case, we use 32 PARSEC and SPLASH-2 threads and eight SPECrate workload copies to match POWER7+'s eight-core architecture. The results are shown in Fig. 14. On average, loadline borrowing achieves 6.2% and 7.7% reduction in power and energy, respectively, across the workloads. For power-intensive workloads such as lu_cb, loadline borrowing can achieve 12.7% improvement.

A handful of benchmarks fall into one of two extremes. On one extreme, some benchmarks that are to the leftmost side on the $x$-axis, such as lu_ncb (not to be confused with lu_cb) and radiosity, suffer from severe performance loss. Performance decreases by more than 20% due to interchip communication overhead (not shown). This in part leads to reduced core power consumption during loadline borrowing (see left $y$-axis), but the longer execution time negatively offsets the benefit and increases total energy consumption.

On the other extreme, some other benchmarks that are to the rightmost side on the $x$-axis, such as radix, zeusmp, lbm, fft and GemsFDTD, experience large performance improvements from load balancing because there is less memory subsystem contention. This performance improvement increases chip activity that could sometimes lead to higher power consumption than the baseline system, such as in the case of radix and fft. Nonetheless, the improved performance brings about large energy reductions for these workloads, as the right $y$-axis in Fig. 14 shows. Improvements range between 50% and 171%.

## 5.2 Adaptive Mapping

Adaptive guardbanding introduces an interesting challenge for deploying latency-sensitive applications in enterprise settings where quality of service (QoS) and service-level agreement (SLA) are critical. On the one hand, adaptive guardbanding's frequency-boosting mode can improve a critical and latency-sensitive application's performance significantly (by as much as 8% according to the data shown earlier in Fig. 5b). On the other hand, chip frequency is a no longer fixed, but is susceptible to fluctuations based on other chip activity. Thus, datacenter operators deploying systems utilizing adaptive guardbanding processors must be cognizant of scheduling implications and workload mapping on these emerging processors.

Fig. 15 illustrates the problem of runtime frequency variation based on measured data. Assume critical application coremark is guaranteed application performance at 4.5 GHz as part of the SLA.[1] This SLA can be met when the adaptive guardbanding processor is filled only with coremark threads (i.e., bar in the center). However, the SLA can be violated if the scheduler coschedules lu_cb threads onto the same chip. coremark's frequency will decrease noticeably when more lu_cb threads are colocated. When only one coremark is scheduled with seven other lu_cb threads (i.e., <1,7> on the $x$-axis), peak frequency drops to 4433 MHz from 4517 MHz. On the contrary, colocating mcf leads to frequency increase. The frequency difference between coscheduling lu_cb threads and mcf threads with coremark is more than 100 MHz. Several other experiments across a wide variety of mappings reveal the same trend.

### 5.2.1 *Solution to Guarantee Performance*

To guarantee application QoS in the face of the adaptive guardbanding processor's variable performance, we propose *adaptive mapping*, which prevents malicious co-runners from taking out the critical workload-frequency resource. Fig. 18 shows our adaptive mapping's end-to-end scheduling logic. Its overall design is based on a standard feedback-driven optimization model. During every scheduling interval, the scheduler checks whether an application has high priority and whether its QoS has been violated by indexing into its job description file. If so, and if the application is sensitive to frequency, the scheduler finds the desired frequency level with the help of an application-specific frequency-QoS model. Then

[1]We use coremark because its footprint is core-contained, so it isolates interference from the memory subsystem and shows frequency changes due only to adaptive guardbanding.

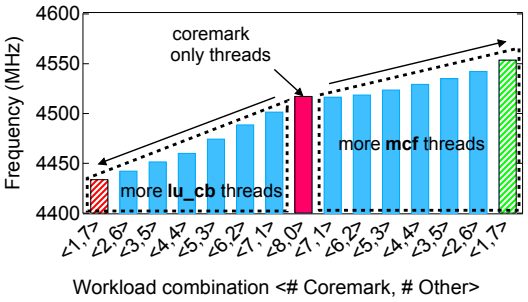



Figure 15: Colocation changes critical application (coremark) frequency by more than 100MHz.

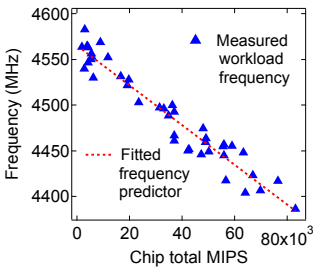



Figure 16: MIPS-based frequency prediction for doing runtime adaptive mapping.

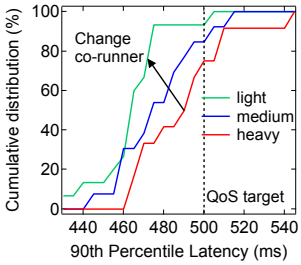



Figure 17: Adaptive mapping co-runner swapping to improve Web Search's QoS.

the scheduler locates a set of suitable co-runners that satisfy the constraint using a frequency predictor. A selected co-runner will replace the current malicious workload. This process repeats every scheduling quantum.

Because a scheduler's overall structure is fairly typical, we focus here on the components that we develop to enable adaptive mapping. These two critical components are shaded in Fig. 18. The first critical component of adaptive mapping is the frequency prediction module. It enables the scheduler to find suitable co-runners that satisfy a particular frequency target under different (hypothetical) application combinations. The second critical component is the scheduling act itself.

We present a simple MIPS-based frequency prediction model that can do this task accurately and quickly. Speed is of the essence because the scheduler is exploring the workload-combination space during runtime, every quantum. We construct a MIPS-based frequency prediction model because processor power consumption corresponds to adaptive guardbanding's behavior strongly (Fig. 10), and to a first order MIPS can be used to accurately predict power. Moreover, it can be readily deployed using existing hardware performance counters.

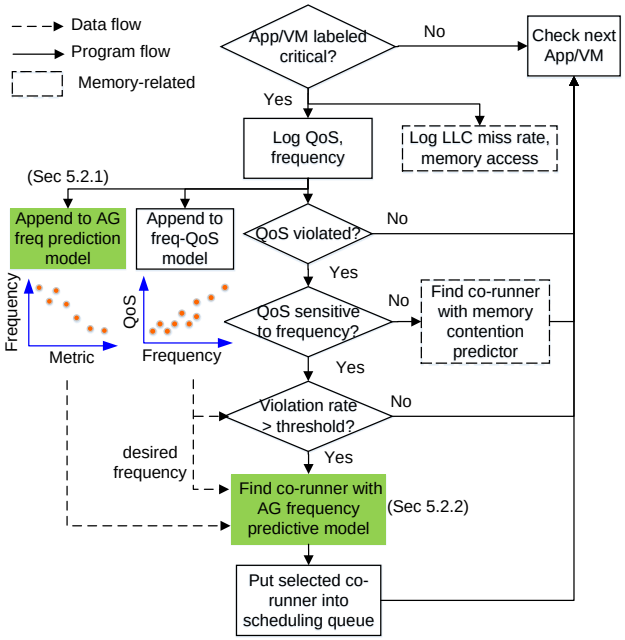



Figure 18: Adaptive mapping scheduler.

To construct a MIPS-based prediction model, we measure adaptive guardbanding's frequency choice when all the cores are stressed by SPEC CPU 2006, PARSEC and SPLASH-2 workloads. Fig. 16 shows the results. Chip total MIPS is the aggregated result of accumulating each core's individual MIPS using hardware counters. Each data point represents one benchmark; together a linear model has root mean square error of only 0.3%. The simplicty of this model makes it a good choice for a scheduler.

### 5.2.2 *Evaluation of Guaranteed Performance*

We demonstrate how adaptive mapping helps guarantee workload QoS using WebSearch [37], a canonical datacenter application. In our simulated scenario, WebSearch runs on one core and is faced with three potential co-runners, each with a different power-consumption profile: *light*, *medium* and *heavy*. We construct the co-runners from coremark threads by constraining the issue rate of the other seven cores on which coremark is running. Moreover, Fig. 15 already shows that real workloads have a detrimental impact on clock frequency. The *light*, *medium* and *heavy* co-runners have a MIPS of about 13,000, 28,000 and 70,000, respectively. These values are chosen because the SPEC, PARSEC and SPLASH-2 applications that we study fall into one of those three performance levels.

The adaptive mapping scheduler aims to control WebSearch's throughput to a level that ensures that its $90^{th}$ percentile latency meets the 0.5-second target 100% of time when it runs by itself, i.e. with no co-runner at all. Initially, WebSearch is blindly co-located with the *heavy* co-runner. As times go on, the scheduler finds that QoS violates more than 25% of the time, as shown in Fig. 17. Guided by the frequency predictor, and to guarantee QoS, the scheduler replaces the current co-runner with the one that has lowest MIPS, i.e., *light*. This reduces the QoS violation rate to less than 7%. As a comparison, co-locating with *medium* reduces the QoS violation rate to about 15%, which is also better than *heavy*.

## 6. RELATED WORK

The $di/dt$ effect and its impact on reliability has been well noted [12, 21, 32, 22]. A plethora of work aims at reducing inductive noise in microprocessors, ranging from the circuit [38, 39], architecture [40, 20, 30, 19, 41, 18, 21, 31, 42] and software [43]. These works usually require intrusive design changes to the hardware [38, 39, 19, 18] and rely on simulation, microarchitecture event detection and activity throttling [40, 20, 41, 18, 43, 31].

Unlike the prior work, we use a measurement-based approach to studying adaptive guardbanding processors [7, 8, 1, 2, 3] that handles droops in a fundamentally new way. Because adaptive guardbanding can effectively improve efficiency and guarantee reliability at the same time, it has gained more attention recently [4, 5, 6].

Prior work on adaptive guardbanding focuses on voltage droop tolerance and system-efficiency analysis at one core or one processor level [7, 8, 1, 2, 3, 4, 5, 6]. In our work, we showcase adaptive guardbanding's system-level implications for core scaling and workload heterogeneity, and we investigate its root causes. Our analysis incorporates $di/dt$ noise and extends to total on-chip voltage drop. Our multicore $di/dt$ noise characterization confirms prior observations [30, 21, 31]. We also observe mitigated typical-case noise and magnified worst-case noise [31] due to on-chip noise propagation [30, 21]. Because adaptive guardbanding deals with $di/dt$ noise well, further investigation should focus on improving its performance with respect to passive voltage drop.

## 7. CONCLUSION

Adaptive guardbanding provides energy and performance benefits, but it is highly workload dependent and its benefits diminish as the number of active cores increases. VRM loadline and the PDN's IR drop are the root causes of adaptive guardbanding's efficiency drop. We propose adaptive guardband scheduling (AGS) to retain the benefits of adaptive guardbanding at high core counts. Under light load, loadline borrowing creates more opportunity for adaptive guardbanding by distributing load across processors. Under heavy load, adaptive mapping guarantees predictable performance for latency-sensitive workloads. Measured results from a production enterprise-class POWER7+ server show loadline borrowing achieves 6.2% power improvement, effectively doubling adaptive guardbanding's original benefit. Our measurements with the latency-sensitive Web application shows adaptive mapping can avoid malicious workload colocations to guarantee, and even improve query-tail latency by 5.2%. These node-level improvements, when put into proper context (i.e., hundreds to thousands of nodes), yield large savings because of the economies of scale at the datacenter level.

## Acknowledgment

We appreciate all anonymous reviewers for their constructive feedbacks. This work is supported by the National Science Foundation grant CCF-1255892 and SRC. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the NSF or the U.S. Government.

## 8. REFERENCES

[1] N. Kurd, J. Douglas, P. Mosalikanti, and R. Kumar, "Next generation intel® micro-architecture (nehalem) clocking architecture," in *IEEE Symposium on VLSI Circuits (VLSIC)*, 2008.

[2] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, and J. B. Carter, "Active management of timing guardband to save energy in power7," in *Proceedings of the International Symposium on Microarchitecture (MICRO)*, 2011.

[3] K. A. Bowman, C. Tokunaga, T. Karnik, V. K. De, and J. W. Tschanz, "A 22nm dynamically adaptive clock distribution for voltage droop tolerance," in *IEEE Symposium on VLSI Circuits (VLSIC)*, 2012.

[4] A. Grenat, S. Pant, R. Rachala, and S. Naffziger, "Adaptive clocking system for improved power efficiency in a 28nm x86-64 microprocessor," in *International Solid-State Circuits Conference (ISSCC)*, 2014.

[5] C. Tokunaga, J. F. Ryan, C. Augustine, J. P. Kulkarni, Y.-C. Shih, S. T. Kim, R. Jain, K. Bowman, A. Raychowdhury, M. M. Khellah *et al.*, "A graphics execution core in 22nm cmos featuring adaptive clocking, selective boosting and state-retentive sleep," in *International Solid-State Circuits Conference (ISSCC)*, 2014.

[6] K. Bowman, S. Raina, T. Bridges, D. Yingling, H. Nguyen, B. Appel, Y. Kolla, J. Jeong, F. Atallah, and D. Hansquine, "A 16nm auto-calibrating dynamically adaptive clock distribution for maximizing supply-voltage-droop tolerance across a wide operating range," in *International Solid-State Circuits Conference (ISSCC)*, 2015.

[7] T. Fischer, F. Anderson, B. Patella, and S. Naffziger, "A 90nm variable-frequency clock system for a power-managed itanium®-family processor," in *International Solid-State Circuits Conference (ISSCC)*, 2005.

[8] J. Tschanz, N. S. Kim, S. Dighe, J. Howard, G. Ruhl, S. Vangal, S. Narendra, Y. Hoskote, H. Wilson, C. Lam *et al.*, "Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging," in *International Solid-State Circuits Conference (ISSCC)*, 2007.

[9] S. Manousopoulos, M. Moreto, R. Gioiosa, N. Koziris, and F. J. Cazorla, "Characterizing thread placement in the ibm power7 processor," in *Proceedings of the International Symposium on Workload Characterization (IISWC)*, 2012.

[10] V. Zyuban, S. Taylor, B. Christensen, A. Hall, C. Gonzalez, J. Friedrich, F. Clougherty, J. Tetzloff, and R. Rao, "Ibm power7+ design for higher frequency at fixed power," *IBM Journal of Research and Development*, 2013.

[11] J. Barth, D. Plass, E. Nelson, C. Hwang, G. Fredeman, M. Sperling, A. Mathews, W. R. Reohr, K. Nair, and N. Cao, "A 45nm soi embedded dram macro for power7 32mb on-chip l3 cache." in *International Solid-State Circuits Conference (ISSCC)*, 2010.

[12] N. James, P. Restle, J. Friedrich, B. Huott, and B. McCredie, "Comparison of split-versus connected-core supplies in the power6 microprocessor," in *International Solid-State Circuits Conference (ISSCC)*, 2007.

[13] M. Floyd, A. Drake, N. Schwartz, R. Berry, C. Lefurgy, M. Ware, K. Rajamani, V. Zyuban, R. Willaman, and R. Zgabay, "Runtime power reduction capability of the ibm power7+ chip," *IBM Journal of Research and Development*, vol. 57, no. 6, pp. 2–1, 2013.

[14] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, J. B. Carter, and R. W. Berry, "Active guardband management in power7+ to save energy and maintain reliability," *IEEE Micro*, 2013.

[15] A. Drake, R. Senger, H. Deogun, G. Carpenter, S. Ghiasi, T. Nguyen, N. James, M. Floyd, and V. Pokala, "A distributed critical-path timing monitor for a 65nm high-performance microprocessor," in *International Solid-State Circuits Conference (ISSCC)*, 2008.

[16] A. J. Drake, M. S. Floyd, R. L. Willaman, D. J. Hathaway, J. Hernandez, C. Soja, M. D. Tiner, G. D. Carpenter, and R. M. Senger, "Single-cycle, pulse-shaped critical path monitor in the power7+ microprocessor," in *Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on*. IEEE, 2013.

[17] J. Tierno, A. Rylyakov, D. Friedman, A. Chen, A. Ciesla, T. Diemoz, G. English, D. Hui, K. Jenkins, P. Muench *et al.*, "A dpll-based per core variable frequency clock generator for an eight-core power7 microprocessor," in *Symposium on VLSI Circuit Digest of Tech Papers*, 2010.

[18] V. J. Reddi, M. S. Gupta, G. Holloway, G.-Y. Wei, M. D. Smith, and D. Brooks, "Voltage emergency prediction: Using signatures to reduce operating margins," in *Proceedings of the International Symposium on High Performance Computer Architecture(HPCA)*, 2009.

[19] M. S. Gupta, K. K. Rangan, M. D. Smith, G.-Y. Wei, and D. Brooks, "Decor: A delayed commit and rollback mechanism for handling inductive noise in processors," in *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, 2008.

[20] M. D. Powell and T. Vijaykumar, "Pipeline damping: A microarchitectural technique to reduce inductive noise in supply voltage," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2003.

[21] V. J. Reddi, S. Kanev, W. Kim, S. Campanoni, M. D. Smith, G.-y. Wei, and D. Brooks, "Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling," in *Proceedings of the International Symposium on Microarchitecture (MICRO)*, 2010.

[22] R. Bertran, A. Buyuktosunoglu, P. Bose, T. J. Slegel, G. Salem, S. Carey, R. F. Rizzolo, and T. Strach, "Voltage noise in multi-core processors: Empirical characterization and optimization opportunities," in *Proceedings of the International Symposium on Microarchitecture(MICRO)*, 2014.

[23] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques (PACT)*, 2008.

[24] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodological considerations," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 1995.

[25] C. Bienia, S. Kumar, and K. Li, "Parsec vs. splash-2: A quantitative comparison of two multithreaded benchmark suites on chip-multiprocessors," in *Proceedings of the International Symposium on Workload Characterization (IISWC)*, 2008.

[26] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE computer*, 2007.

[27] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proceedings of the International Symposium on Microarchitecture (MICRO)*, 2003.

[28] W. Huang, C. Lefurgy, W. Kuk, A. Buyuktosunoglu, M. Floyd, K. Rajamani, M. Allen-Ware, and B. Brock, "Accurate fine-grained processor power proxies," in *Proceedings of the International Symposium on Microarchitecture (MICRO)*, 2012.

[29] M. Floyd, M. Allen-Ware, K. Rajamani, B. Brock, C. Lefurgy, A. J. Drake, L. Pesantez, T. Gloekler, J. A. Tierno, P. Bose *et al.*, "Introducing the adaptive energy management features of the power7 chip," *IEEE Micro*, 2011.

[30] 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 *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2007.

[31] T. Miller, R. Thomas, X. Pan, and R. Teodorescu, "Vrsync: Characterizing and eliminating synchronization-induced voltage emergencies in many-core processors," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2012.

[32] Y. Kim, L. K. John, S. Pant, S. Manne, M. Schulte, W. L. Bircher, and M. S. S. Govindan, "Audit: Stress testing the automatic way," in *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*. IEEE, 2012, pp. 212–223.

[33] P. U. Murthy, "Overview of the current approaches to enhance the linux scheduler," in *Linux Foundation Collaboration Summit*, 2013.

[34] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, and C. Kozyrakis, "Towards energy proportionality for large-scale latency-critical workloads," in *Proceeding of the International Symposium on Computer Architecuture (ISCA)*, 2014.

[35] J. Leverich and C. Kozyrakis, "Reconciling high server utilization and sub-millisecond quality-of-service," in *Proceedings of the European Conference on Computer Systems (EuroSys)*, 2014.

[36] "Spec cpu 2006." [Online]. Available: https://www.spec.org/cpu2006/

[37] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds," in *Proceedings of the International Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2012.

[38] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proceedings of the International Symposium on Microarchitecture (MICRO)*, 2003.

[39] D. Blaauw, S. Kalaiselvan, K. Lai, W.-H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull, "Razor ii: in situ error detection and correction for pvt and ser tolerance," 2008.

[40] E. Grochowski, D. Ayers, and V. Tiwari, "Microarchitectural simulation and control of di/dt-induced power supply voltage variation," in *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*, 2002.

[41] M. S. Gupta, V. J. Reddi, G. Holloway, G.-Y. Wei, and D. M. Brooks, "An event-guided approach to reducing voltage noise in processors," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2009.

[42] R. Zhang, K. Wang, B. H. Meyer, M. R. Stan, and K. Skadron, "Architecture implications of pads as a scarce resource," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2014.

[43] V. J. Reddi, S. Campanoni, M. S. Gupta, M. D. Smith, G.-Y. Wei, D. Brooks, and K. Hazelwood, "Eliminating voltage emergencies via software-guided code transformations," *ACM Transactions on Architecture and Code Optimization (TACO)*, 2010.