2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)

Adaptive Voltage/Frequency Scaling and Core Allocation for Balanced Energy and Performance on Multicore CPUs

George Papadimitriou Athanasios Chatzidimitriou Dimitris Gizopoulos
Dept. of Informatics and Telecommunications, University of Athens, Greece
{georgepap | achatz | dgizop}@di.uoa.gr

***Abstract*—Energy efficiency is a known major concern for computing system designers. Significant effort is devoted to power optimization of modern systems, especially in large-scale installations such as data centers, in which both high performance and energy efficiency are important. Power optimization can be achieved through different approaches, several of which focus on adaptive voltage regulation. In this paper, we present a comprehensive exploration of how two server-grade systems behave in different frequency and core allocation configurations beyond nominal voltage operation. Our analysis, which is built on top of two state-of-the-art ARMv8 microprocessor chips (Applied Micro's X-Gene 2 and X-Gene 3) aims (1) to identify the best performance per watt operation points when the servers are operating in various voltage/frequency combinations, (2) to reveal how and why the different core allocation options on the available cores of the microprocessor affect the energy consumption, and (3) to enhance the default Linux scheduler to take task allocation decisions for balanced performance and energy efficiency. Our findings, on actual servers' hardware, have been integrated into a lightweight online monitoring daemon which decides the optimal combination of voltage, core allocation, and clock frequency to achieve higher energy efficiency. Our approach reduces on average the energy by 25.2% on X-Gene 2, and 22.3% on X-Gene 3, with a minimal performance penalty of 3.2% on X-Gene 2 and 2.5% on X-Gene 3, compared to the default system configuration.**

***Keywords*-Energy efficiency; voltage and frequency scaling; power consumption; multicore characterization; micro-servers;**

I. Introduction

In the last decade, intense research activities in studying the limits of energy consumption in microprocessors for both single- and multi-thread applications took place. One of the most popular methods for aggressive energy reduction is by exploiting the pessimistic voltage guardbands of the microprocessor chips [1]–[6]. The minimum safe operating voltage (V_{min}) of a microprocessor depends on the technology node, static variation (caused by manufacturing and environmental factors) and dynamic variation (caused by workloads differences and device degradation). In order to guarantee correct execution in all cases, the nominal operating voltage is set to a pessimistic level that is expected to be sufficient to account for all these variable factors. The difference between the actual V_{min} of a chip and the nominal operating voltage is a safe guardband with which every chip and every workload is guaranteed to operate correctly. Unfortunately, the voltage (and corresponding frequency) guardbands are excessively conservative and impede power efficiency and performance levels, which could potentially be derived by bringing the supply voltage and the clock frequency closer to the system's best capabilities.

To improve the microprocessor's efficiency (in terms of either power or performance), several hardware and software techniques have been proposed, such as Dynamic Voltage and Frequency Scaling (DVFS) [7] as well as several power capping approaches [8]. The ability to cap peak power consumption has recently gained strong interest in several important computing domains (e.g., mobile devices to data centers [9]) since the state-of-the-art high-end microprocessors currently support such features within their power management subsystem. Power capping is realized through power-performance knobs such as DVFS, pipeline throttling or memory throttling [8].

The premise of such approaches is that the microprocessor's workloads as well as the cores' activity vary. Voltage and/or frequency scaling during epochs, where peak performance is not required, enable a DVFS-capable system to achieve average energy efficiency gains without affecting peak-performance adversely or harming correctness. At a specific clock frequency, energy efficiency gains are limited by voltage guardbands that guarantee correct operation in the presence of dynamic variations. Voltage and frequency scaling methods can efficiently reduce the energy consumption without compromising performance. They, however, do not take into consideration either the important characteristics of the executed applications or the core-to-core variation, in order to provide even more aggressive energy reduction.

In this paper, we are based on two recent state-of-the-art ARMv8-compliant multicore CPUs, Applied Micro's (now Ampere Computing) X-Gene 2 and X-Gene 3 to propose a new software-based scheme for server-grade machines, which considers all the diverse aspects of the problem, and provides large energy savings while maintaining high performance levels. Naturally, relaxed performance constraints can lead to further energy improvements, but in this paper, we restrict ourselves to minimal performance impact. The X-Gene 3 microprocessor is developed for HPC applications and has comparable performance to high-end Intel Xeon microprocessors [10]. However, neither X-Gene 2 nor X-Gene 3 microprocessors have predefined power management states as in x86 and POWER architectures. Although X-

Gene microprocessors support dynamic frequency scaling, the voltage is always the same for every different frequency step (which is equal to $V_{nominal}$) and can be only explicitly modified. Particularly, the main contributions of this work are:

1. We expose the pessimistic voltage guardbands of two state-of-the-art ARMv8 microprocessors (manufactured in 28nm and 16nm – the X-Gene 2 and X-Gene 3, respectively) to identify the safe V_{min} points of the CPU chips in multicore executions. We show that as the number of active threads increases, core-to-core and workload-to-workload variability have a minimal impact on V_{min}.
2. We present measurements on the correlation of the safe V_{min} to the voltage droop magnitude, and show that in multicore executions the emergency voltage droop events occur regardless of the workload. However, for executions in a single or very few cores, core-to-core and workload-to-workload variability exist.
3. We perform an extensive study to identify and analyze the tradeoffs between energy and performance at different voltage and frequency combinations, as well as at different thread scaling and core allocation configurations. Our analysis reveals that depending on the course-grain characteristics of a program and the number of active threads, there is an optimal combination of voltage, frequency and core allocation for better energy efficiency.
4. We also developed a simple online monitoring daemon which monitors the running processes on the system and guides the Linux scheduler to take the appropriate decisions regarding: (a) the core(s) to which a new process should be assigned, and (b) when one or more running processes should be migrated to other cores. At the same time, the daemon dynamically adjusts the V/F settings according to the optimal policies.
5. Finally, we evaluate the optimal energy efficient scheme by running a realistic scenario of a server's operation, which (a) randomly selects the issued programs, (b) dynamically migrates the running processes on the system, and (c) dynamically adjusts the voltage and frequency settings. We report several comparisons among different configurations to present a detailed evaluation of the optimal scheme, and show that it can achieve on average 25.2% energy savings on X-Gene 2, and 22.3% energy savings on X-Gene 3, with a minimal performance penalty of 3.2% on X-Gene 2 and 2.5% on X-Gene 3 compared to the default voltage and frequency microprocessor's conditions.

II. Experimental Setup

A. Platforms

This study was performed on two different state-of-the-art ARMv8 micro-servers: Applied Micro's (now Ampere Computing) X-Gene 2 and X-Gene 3, which consist of 8 and 32 64-bit ARMv8-compliant cores, respectively. Both microprocessors offer high-end processing performance and come along with a subsystem that features a Scalable Lightweight Intelligent Management processor (SLIMpro) to enable flexibility in power management, resiliency and end-to-end security for a wide range of applications. The dedicated SLIMpro processor monitors system sensors, configures system attributes (e.g. regulates supply voltage, etc.) and can be accessed by the system's running Linux kernel.

X-Gene 3 microprocessor has a main power domain that includes the CPU cores, the L1, L2 and L3 cache memories, and the memory controllers, which is called PCP (Processor ComPlex) power domain, as shown in Figure 1, and is the one that consumes the largest part of the overall power consumption. Figure 1 presents the architecture of X-Gene 3, however, the X-Gene 2 has similar structure; the difference is that it has 8 cores instead of 32, and the L3 cache, which is 8MB instead of 32MB, is located in a different domain. The operating voltage of the main power domain can change from 980mV downwards in X-Gene 2 and from 870mV downwards in X-Gene 3. While all the CPU cores operate at the same voltage, each pair of cores (PMD – Processor MoDule) can operate at different frequency in both chips. The frequency can range from 300MHz up to 2.4GHz in X-Gene 2, and from 375MHz to 3GHz in X-Gene 3 (at 1/8 steps of the maximum clock frequency in both microprocessors). Table I presents the main characteristics of the two microprocessors. Both platforms run CentOS 7.3 with Linux kernel version 4.11.

B. Experimental Configuration

In our analysis, we use 25 benchmarks from 3 different benchmark suites: the NAS Parallel Benchmark Suite v3.3.1 (NPB) [11], the SPEC CPU2006 suite [12], and the PARSEC v3.0 suite [13]. NPB are programs designed to evaluate the performance of parallel supercomputers and have been used in several studies regarding performance

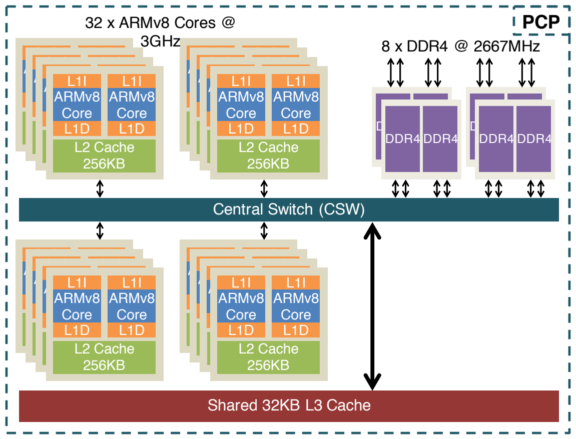

Figure 1. X-Gene 3 block diagram.

TABLE I
BASIC PARAMETERS OF X-GENE 2 AND X-GENE 3.

Parameter	X-Gene 2	X-Gene 3
ISA	64-bit OoO (4-issue)	
Pipeline	ARMv8 (AArch64, AArch32, Thumb)	
CPU	8 cores	32 cores
Core clock	2.4 GHz	3.0 GHz
L1 Instr. Cache	32KB per core (Parity Protected)	
L1 Data cache	32KB per core (Parity Protected)	
L2 cache	256KB per PMD (ECC Protected)	
L3 cache	8MB (ECC Protected)	32MB (ECC Protected)
Technology	28 nm (bulk CMOS)	16 nm (FinFET)
TDP	35 W	125 W
Nominal Voltage	980 mV	870 mV

and energy efficiency [14], [15]. Given that this study is primarily based on the multicore execution, we use 6 NPB parallel benchmarks (*CG*, *EP*, *FT*, *IS*, *LU*, *MG*) and 6 PARSEC parallel benchmarks (*swaptions*, *blackscholes*, *fluidanimate*, *canneal*, *bodytrack*, *dedup*) for multi-thread executions, and moreover, 13 SPEC CPU2006 single-thread benchmarks (both FP and INT class, which provide diverse behavior in voltage-scaled conditions, part of those were also used in [6]) for evaluating multiple copies of single-threaded executions. Note that, in a parallel execution with N threads (assume an NPB or PARSEC program with N parallel threads), all active threads compute parts of the same work, which is executed once. On the other hand, when we execute N copies of the same single-threaded benchmark, the microprocessor executes N times the same work. Therefore, we cannot directly compare the two cases as they refer to different amount of work and thus, the energy values of the single-threaded benchmarks are normalized to the number of running instances in order to deliver a fair comparison between the two groups of programs. For example, if the microprocessor executes N instances of the *namd*, the energy will be equal to energy_of_N_instances / N.

For a comprehensive coverage of as many different execution behaviors as possible, we executed all 25 benchmarks in 3 different threading configurations in both X-Gene 2 and X-Gene 3 systems:

Max threads: In all available cores of each microprocessor (8 cores for X-Gene 2 and 32 cores for X-Gene 3)

Half threads: In half of the cores (4 cores for X-Gene 2 and 16 cores for X-Gene 3), and

Quarter threads: In one quarter of the cores (2 cores for X-Gene 2 and 8 cores for X-Gene 3).

For these 3 different thread-scaling options, we executed the programs at the maximum frequency of each microprocessor (2.4GHz for X-Gene 2 and 3GHz for X-Gene 3), and at the half frequency (1.2GHz and 1.5GHz, respectively). It is essential to highlight, however, that clock frequencies larger than the half clock of both microprocessors have similar safe V_{min} as in the highest clock frequency, and frequencies smaller than the half clock have similar safe V_{min} as in the half clock. The reason is that both microprocessors support clock skipping and clock division, which, in combination, set the effective frequency of the PMD relative to its clock source. The reason is that clock ratios greater or less than 1/2 on the input clock are implemented via clock skipping on the input clock. Clock ratio equal to 1/2 is naturally implemented via clock division on the input clock. For this reason, we do not present any results for the intermediate frequencies because they provide exactly the same V_{min} points.

Exceptionally, for X-Gene 2 only, we also present results at 0.9GHz, in which we noticed a significant reduction of the V_{min}, and thus, much larger energy savings compared to 1.2GHz, with minimal impact on performance. The reason is that these micro-servers implement the most recent CPPC (Collaborative Processor Performance Control) power and performance management specification of ACPI 5.1 [16]. CPPC is a new way to control the performance of cores using an abstract continuous scale in frequency, instead of a discretized P-state scale (as in legacy ACPI). Therefore, during runtime, when there is a request for 1.2GHz, in practice the actual frequency of the microprocessor is scaled below and above of the 1.2GHz, so that it effectively provides an average frequency of 1.2GHz. As a result, the actual frequency properties are limited by the highest frequency setting being used, which in this case is above half (without clock division). This is a frequency interleaving strategy which is provided by the CPPC and cannot be changed by software. Although X-Gene 3 operates also with CPPC specification, we did not observe the same behavior below the 1.5GHz as in X-Gene 2. This is an interesting finding of the characterization part of this work, and thus, we report experiments in X-Gene 2 for three different frequencies that represent all different behaviors: 2.4GHz, 1.2GHz and 0.9GHz and in X-Gene 3 we report our experiments at 3GHz and 1.5GHz.

Furthermore, for the needs of our core-allocation analysis, when we execute multicore experiments with the number of threads being less than the available cores of each chip, we characterize different combinations of core allocation, and we have two main categories: *spreaded* threads and *clustered* threads. As shown in Figure 2, spreaded thread configuration

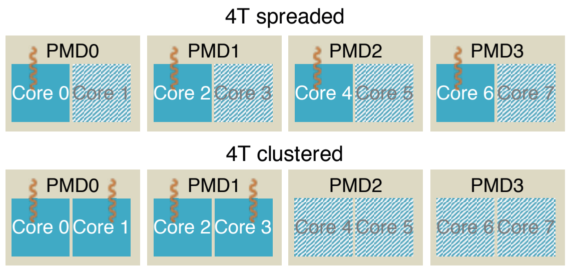

Figure 2. 4 running threads in 2 different core allocation configurations: spreaded & clustered.

refers to the threads running in separate PMDs each, while the clustered thread configuration to the threads running in consecutive cores (both cores of the PMD are occupied).

III. Voltage Margins Identification

This part of the paper focuses on a quantitative analysis of the safe V_{min} for two micro-servers of the same architecture in order to expose the potential guardbands of each chip, as well as to quantify the factors that determine the V_{min} of multicore executions.

A. Exposing Safe V_{min} Values

We experimentally obtain the safe V_{min} values on the two different technology micro-servers: X-Gene 2 and X-Gene 3. For all of our experiments, we consider a voltage level as a safe V_{min} if the program passes it 1000 times. Safe V_{min} is the minimal working voltage. Note that we also study the error behavior for each program (subsection III-B) operating below its safe V_{min} point, but we run it 60 times for each configuration (frequency, core allocation, and thread scaling) through the entire voltage range from the safe V_{min} until the system crash point.

Figure 3 shows the V_{min} characterization results for the 25 benchmarks on X-Gene 2 and X-Gene 3. The reported V_{min} for each program is the lowest (safe) voltage setting where all 1000 executions of each program completed successfully, without hardware errors notification, program output mismatches (silent data corruptions – SDCs) or other abnormal behavior, such as a process timeout, system crash or thread hang. The top graphs in Figure 3 present the 8-thread and 4-thread executions of the benchmarks in X-Gene 2 for the three different frequencies: 2.4GHz, 1.2GHz and 0.9GHz. The bottom graphs present the V_{min} results for the same benchmarks on X-Gene 3 with 32, 16, and 8-thread executions for 3GHz and 1.5GHz. Figure 3 clearly shows, that for the same number of threads and at the same frequency, the safe V_{min} for all 25 benchmarks is virtually the same. There are some cases, where a benchmark has a little lower V_{min} than the rest, however, the maximum difference is only 10mV or ∼1% of the nominal voltage.

However, as shown in Figure 4, in single-core (top graphs) and two-core (bottom graphs) executions we observed on average 5% core-to-core and workload-to-workload variation (similar to prior studies, e.g.: [1], [2], [6], [17]–[19]). The yellow-colored region of each graph is the safe region, in which all executions are completed correctly, and the dark-colored region is the unsafe region, in which we observed several abnormal behaviors like SDCs, crashes, etc. The intersection of the two colors is the safe V_{min}. For all programs, we observed up to 40mV workload variation and up to 30mV core-to-core variation in X-Gene 2, and up to 20mV core-to-core and workload variation in X-Gene 3. Specifically, Figure 4 shows that PMD2 (cores 4, 5 at the top graph that belong to that PMD) is the most robust one (its safe region is larger than the other PMDs), in contrast to PMD0 (cores 0, 1 at the top graph) and PMD1 (cores 2, 3 at the top graph), which are the most sensitive ones (their safe region is the smallest one). Since the scope of this study is not single-core executions, a major finding is that in multicore executions the safe V_{min} marginally depends on the workload (different program), but heavily depends on the number of active cores and the frequency, as shown in Figure 3. We discuss the reasons for these differences in Section IV.

B. Unsafe Region Investigation

As we described in the previous subsection, in multicore executions the safe V_{min} for every program is virtually the same for the same frequency and number of threads. Interestingly, we can notice such a behavior, across different benchmarks, frequencies and number of active threads, also in the region below the safe V_{min} (the unsafe region, where at least one run, faces an abnormal behavior). Figure 5 shows the cumulative probability of failure (pfail) at each voltage level below the V_{min} (the V_{min} is the last voltage

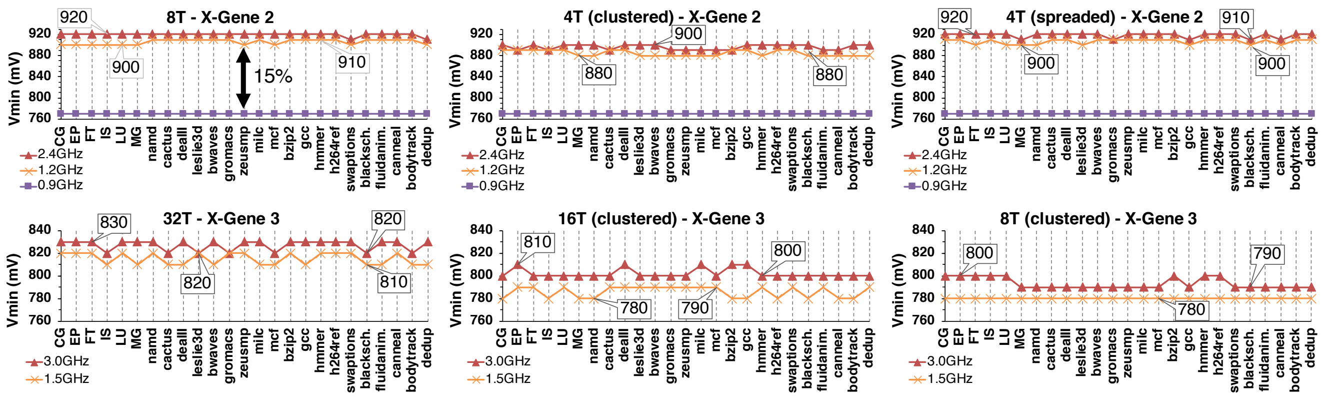

Figure 3. The complete V_{min} characterization results. The top diagrams show the X-Gene 2 safe V_{min} points for all benchmarks with 8 and 4 threads in 2.4, 1.2 and 0.9 GHz. The bottom diagrams show the X-Gene 3 safe V_{min} points for all benchmarks with 32, 16 and 8 threads in 3.0 and 1.5 GHz.

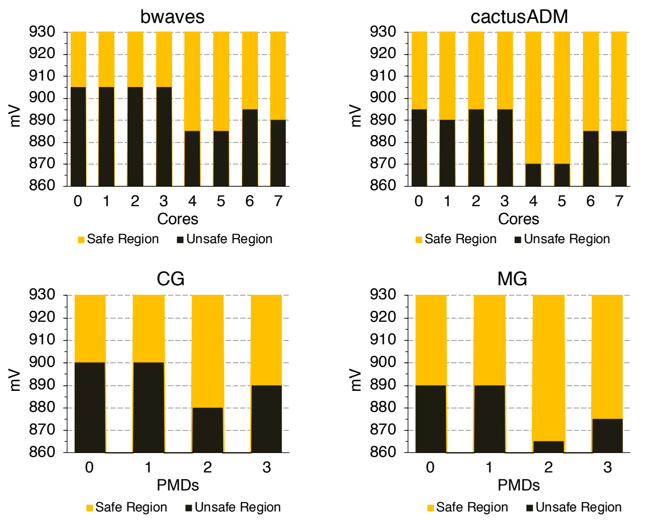

Figure 4. Single-core execution (top) and two-core executions (bottom) in X-Gene 2 at 2.4GHz.

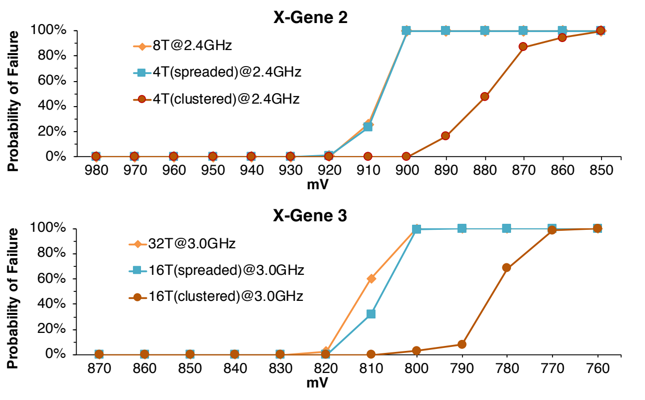

Figure 5. Probability of Failure (pfail) in all voltage levels from nominal level down to the levels of complete failure for different frequency, core allocation, and thread scaling options.

step with pfail=0). Each line of the graph corresponds to the average pfail of the 25 benchmarks, in two different core allocation and thread scaling options. Similar to Figure 3, for the same configuration (number of threads and frequency) the workloads have very small differences on the safe V_{min}, however, for different core allocation options we observe different behavior in V_{min}. The same pattern appears also in the unsafe region.

Consider for example the configurations with "max threads" and with spreaded "half threads" at maximum frequency (8T @ 2.4GHz and 4T (spreaded) @ 2.4GHz in X-Gene 2, and 32T @ 3GHz and 16T (spreaded) @ 3GHz in X-Gene 3). As shown in Figure 5, these two lines of the graph are virtually identical and have the most severe behavior (pfail=100% means that all identical executions failed to complete. On the other hand, a pfail=10% means that there are 90% chances for an application to execute correctly in that voltage). On the contrary, if we change the core allocation of "half threads", we notice that the behavior changes significantly. Consider now, the "clustered" core allocation option (8T @ 2.4GHz and 4T (clustered) @ 2.4GHz in X-Gene 2, and 32T @ 3GHz and 16T (clustered) @ 3GHz in X-Gene 3). Although the frequencies between "max threads" and "half threads" are the same, the pfail (and also the safe V_{min}) are very different ("half threads" configuration has lower safe V_{min} and pfail than "max threads") because the core allocation was changed. We demonstrate in Section IV how this observation is correlated to the voltage droop magnitude.

Based on this experimentation, we conclude that the dominant factors which can affect the safe V_{min} in multicore executions and also the failure probability, are (a) the frequency, and (b) the core allocation. The workload itself has only a marginal impact on the V_{min} in multicore execution for the same number of threads in different frequencies (see Figure 3), however, we can see that lower frequencies have significantly lower safe V_{min} (and pfail respectively) for all the benchmarks than at the maximum frequency. For the same number of threads, we also notice that reducing the frequency to the "half speed" we can further decrease the operating voltage by approximately 3%, while the 0.9GHz runs show a significant reduction (approximately 15%) at the V_{min} due to the clock division that is activated at that frequency (see explanation in subsection II-B). Moreover, using a different core allocation strategy (clustered or spreaded) for the same number of threads, we can achieve further voltage reduction by 4%.

IV. Analysis of V_{min} Impact Factors

In this section we study the correlation of the contributors in V_{min} presented before, and identify the best combination of workload characteristics, frequency and core allocation towards the highest energy efficiency. We also discuss how these findings can be exploited in runtime.

A. Impact of Frequency and Core Allocation on Safe V_{min}

Apart from the reduced frequency, in which the operating voltage can be decreased, the second major factor that can reduce the safe V_{min} is the core allocation. Several studies have exposed the safe V_{min} for each individual core of the microprocessor, by exploiting the static variation and the workload-to-workload variation [1]–[6]. These studies demonstrate that different workloads have different safe V_{min}, and thus, there is an intense research to provide several methods which will be able to predict the safe V_{min} for each program, dynamically, when the microprocessor operates normally. However, we show that the V_{min} variation among different workloads (multi-threaded or multiple copies of a single-threaded application) fades away as the number of running threads increases, and the remaining key factors that determine the V_{min} are the frequency and core allocation. It is known that the larger the number of threads running in the microprocessor, the more noise is generated from voltage droops and the interference of running processes in the system. The findings of our analysis quantify the magnitude of this dependence.

To understand this phenomenon, we study the voltage droop magnitude of the microprocessors for all the different frequency and core allocation configurations, by leveraging the embedded oscilloscope in the X-Gene 3 microprocessor. PMU (Performance Monitoring Unit) counters, which can be accessed by the Perf tool [20], are located in the microprocessor and can be used to monitor the frequency and the magnitude of voltage droop events. Figure 6 presents two different ranges of voltage droop magnitude when the microprocessor operates at 3GHz: (a) the [55mV, 65mV) in which we present the configurations of all programs that produce voltage droops more than or equal to 55mV and less than 65mV, and (b) the [45mV, 55mV) in which we present the configurations of all programs that produce voltage droops more than or equal to 45mV and less than 55mV. Both graphs show the total number of droop detections per 1M cycles for each program (benchmark names are omitted to save space). As we can see in the left graph of Figure 6, the configurations with 32 threads and 16 spreaded threads, which means that all 16 PMDs of the microprocessor running at 3GHz frequency (note that the frequency can be changed per pair of cores – PMD) produce voltage droop magnitude between 55mV and 65mV. However, the configuration of 16 clustered threads (meaning 8 PMDs of the microprocessor running at 3GHz) has almost zero droops in the range of [55mV, 65mV) for all programs. On the other hand, in the right graph of Figure 6, the configurations with 16 clustered threads and 8 spreaded threads (8 PMDs operate at 3GHz frequency in both configurations) produce voltage droop magnitude in the range of [45mV, 55mV). However, the configuration with 8 clustered threads (4 PMDs) has almost zero droops in that range for any program.

B. Impact of the Workload on Frequency and Core Allocation

Apart from the magnitude of the correlation of the safe V_{min} to frequency and core allocation, we also quantify the impact of the workload class (CPU-intensive vs. memory-intensive) to: (a) the safe V_{min}, (b) the workloads' performance, and (c) the energy consumption. Figure 7 shows an illustrative example of the difference in energy of all 25 programs when running at the maximum frequency, with the same number of threads (4 threads in this case) but in different core allocations (4T clustered vs. 4T spreaded) on the X-Gene 2 (the observation is similar in X-Gene 3). The red line with numbers at the top of each pair of bars shows the energy consumption difference between the two core allocations. We can see that the energy difference between these two configurations varies from -9.6% to 14.2%, depending on the characteristics of each workload. Negative percentages indicate that the spreaded-thread configuration needs higher energy than the clustered-thread configuration, while positive percentages indicate the opposite. In particular, the benchmarks shown at the right side of the dashed line have better energy when their threads are spreaded across the cores, unlike the benchmarks shown at the left side of the vertical dashed line which are more energy efficient when executed in consecutive cores (clustered configuration; see Figure 2). The reason is that the rightmost benchmarks are the most memory-intensive benchmarks, while the leftmost benchmarks are the most CPU-intensive.

As the previous sections present, frequency reduction and the optimal core allocation can enable significant opportunities of lowering supply voltage, however, reduced frequency also translates to degraded performance. Reduced frequency in CPU cores impacts their performance without affecting the lower memory levels (L3 cache and DRAM). This means that a program that is highly computational (CPU-intensive) will be proportionally affected by the reduced frequency. On the other hand, a program that experiences long stalls waiting for the memory to respond will be less affected from the core frequency reduction, as this will be hidden by the long memory delays (memory-intensive programs). The key difference between them is that in CPU-intensive programs the system part that acts as a performance limiter is the CPU core part (pipeline, L1 and L2 caches) while for the memory

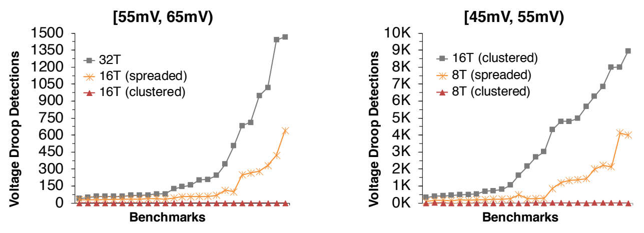

Figure 6. Voltage droop detections for each program in 2 different voltage droop magnitudes.

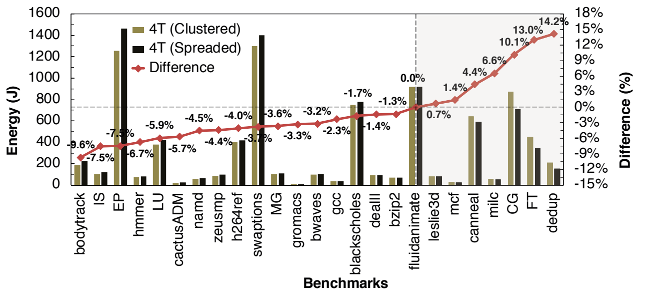

Figure 7. Energy of all benchmarks for 2 different core allocations. The benchmarks on the left of the dashed line are CPU-intensive while the ones on the right are memory-intensive.

intensive programs, it is the memory part (L3 and DRAM).

In practice, we can use this property to combine lower frequency and lower voltage with small performance difference for memory intensive workloads, to increase their energy efficiency while still complying with high performance constraints. Previous studies (e.g.: [21], [22]), have exposed the phases of a program which are memory-intensive, and have proposed this feature as a proxy in a single-core microprocessor to guide the DVFS to reduce the frequency in that specific program phases. In this work we further extend this practice to show how this property can be also used to guide core allocation decisions and how this feature impacts the microprocessor's energy and program's performance. In order to identify the class of each program, we follow the method proposed in [23], to track the access rates of the lower memory hierarchy, and more precisely L3 cache accesses. High L3 access rate means high memory activity in the lower memory hierarchy (memory-intensive program). To find the exact threshold level that separates the two classes, we initially identify what programs are memory intensive and then the L3 access rates of these benchmarks.

Figure 8 presents the performance impact of each program when we introduce contention on the shared CPU resources. We do that by executing multiple copies of the same program on all cores. Programs that are affected the most have high activity on the shared resources (and thus the contention negatively impacts the performance). As an example, we can see the *CG* and *FT*, which are the most memory-intensive benchmarks because their execution time is significantly reduced in a multi-threaded execution compared to the single-threaded one (ratio is much smaller than 1) due to the high contention at the memory system. On the other hand, the *namd* and *EP* are the most CPU-intensive benchmarks because their execution time in multi-threaded execution is virtually the same as in the single-threaded execution (ratio is very close to 1).

We then use the performance monitoring counters of the microprocessor as an indication of the workload class (CPU vs. memory intensive). In particular, we use the L3 Cache (L3C) memory access rate (by monitoring the L2 miss counters), which will allow us to identify the class of each program during runtime. Figure 9 shows the L3 Cache access rate for the 3 different threading configurations of 32, 16, and 8 threads (for this example we used the X-Gene 3 platform; the same behavior occurs in X-Gene 2 also), measured at 3GHz clock frequency. Based on the L3C access rate metric, and also on the experimental analysis of the safe V_{min}, we found that the threshold which defines the high memory activity is 3K accesses per 10^6 cycles. Executions above this threshold are the most memory-intensive, while those below the threshold are the most CPU-intensive.

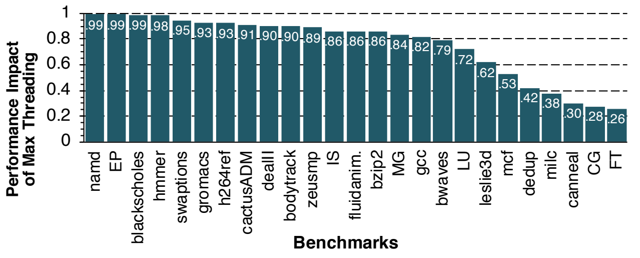

Figure 8. Relative performance of all benchmarks. The y-axis presents the ratio of the execution time of one instance of the single-threaded execution divided by the execution time of multiple instances.

C. Summarizing the Factors that Impact the Safe V_{min}

Figure 10 quantifies the impact on the safe V_{min} that each factor has on the microprocessors of our analysis. The values shown in this figure are derived from our study in the X-Gene 2 microprocessor, however, the corresponding values for X-Gene 3 are similar. As we can see in Figure 10, the workload variability can affect at most 1% the safe V_{min}, while the core allocation and clock frequency are the major contributors to the safe V_{min}. The reason it that, as we demonstrated in subsection IV-A, frequency and core allocation are the main factors that can affect the voltage droop magnitude. In particular, the largest amount of voltage reduction (12%) is a result of clock division in a specific clock frequency, while just one step further frequency reduction (due to clock skipping) delivers 3% further voltage reduction. The following sections identify the best combinations concerning the voltage, frequency and application characteristics, which can lead to the highest energy savings in CPUs with many cores.

V. Performance And Energy Trade-Offs

For a more comprehensive comparison of the different options and configurations we measure and present both the

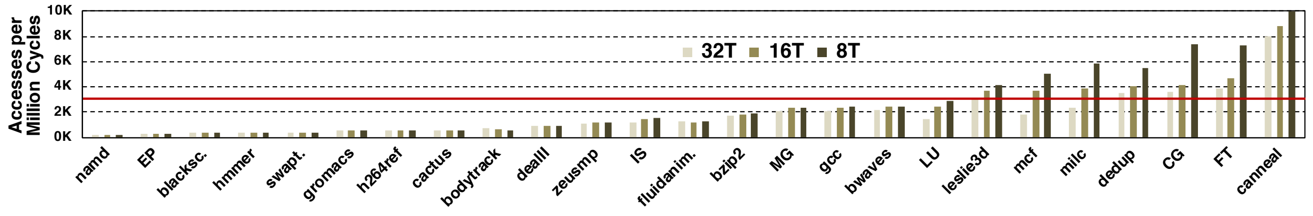

Figure 9. L3 Cache access rate per 1M cycles for the 25 benchmarks and the 3 threading configurations (32, 16 and 8 threads).

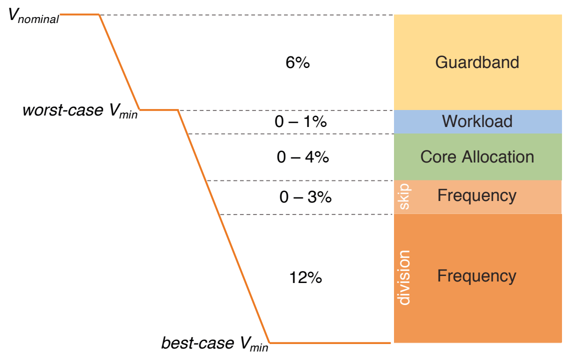

Figure 10. The magnitude of V_{min} dependence on frequency, core allocation and workloads.

overall energy consumed for each particular workload execution, as well as the energy delay squared product (ED2P) metric which combines the energy and the performance of each workload and configuration.

A. Energy Efficiency

Figure 11 shows the energy consumption for 5 benchmarks (due to space limitations; all other benchmarks follow the same pattern as the ones presented) and all configurations of X-Gene 2 and X-Gene 3 systems. The benchmarks are sorted (from left to right) from the most CPU-intensive to the most memory-intensive ones, as it is also shown in Figure 8 (*namd* and *EP* are the most CPU-intensive benchmarks, while *milc*, *CG* and *FT* are the most memory-intensive ones).

X-Gene 2 reports significant energy savings for all cases when running at 0.9GHz, which is attributed to the significantly lower safe V_{min} voltage (as shown in Figure 3, which is possible due to the clock division) and the lower frequency. For CPU-intensive benchmarks (*namd*, *EP*), the frequency reduction (from 2.4GHz to 1.2GHz) does not have an observable impact on the total energy consumption. We can only notice energy improvements for 1.2GHz on the memory-intensive applications (*milc*, *CG*, *FT*). Lower CPU speed reduces the performance gap between CPU and memory and leads to a more balanced system. This is why we can gain significant power savings without sacrificing too much performance, and thus achieve better energy efficiency.

Similar observations hold for X-Gene 3. The low-frequency behavior of X-Gene 3 (1.5GHz) matches the one of the 1.2GHz of X-Gene 2. For the CPU-intensive benchmarks the highest frequency (3GHz) gives the best energy (due to the faster execution), but again we can see that memory-intensive applications are more energy efficient in lower frequency. Note that, in X-Gene 3 we do not present results for lower frequencies than 1.5GHz, because as we have described in subsection II-B, frequency settings bellow 1.5GHz have the same safe V_{min} as in 1.5GHz due to clock skipping, and thus, there is only performance impact.

B. Combined Energy and Performance Considerations

Energy consumption itself is a valuable metric that directly translates to cost, however, it occasionally implies very slow system configurations (very low frequencies), which could violate latency and throughput requirements on a server environment. To avoid this bias in the comparisons of different configurations, other metrics such as the energy delay product ($EDP = E \times D$) and the energy-delay squared product ($ED2P = E \times D^2$) have been proposed for high-end systems [24]. Given that our work focuses on server-grade CPUs, we have chosen to present the energy delay squared product (ED2P) for all of our experiments to show a fair comparison among benchmarks, between different microprocessors, and more importantly to provide a fair representation of the relation between energy and performance. In this subsection, we focus on this metric for the comparison of the different X-Gene 2 and X-Gene 3 configurations.

Figure 12 shows the ED2P for 2.4GHz, 1.2GHz and 0.9GHz configurations for the X-Gene 2, and 3GHz and

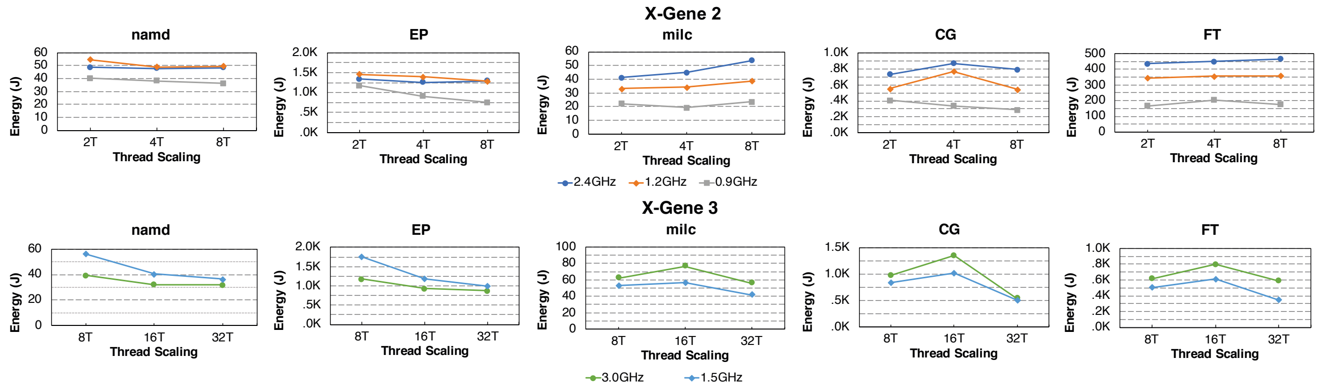

Figure 11. Energy consumption in Joules for 8, 4, and 2 threading options for 3 different frequencies (2.4GHz, 1.2GHz, 0.9GHz) of X-Gene 2 (top) and 32, 16, and 8 threading options for 2 different frequencies (3GHz, 1.5GHz) of X-Gene 3 (bottom).

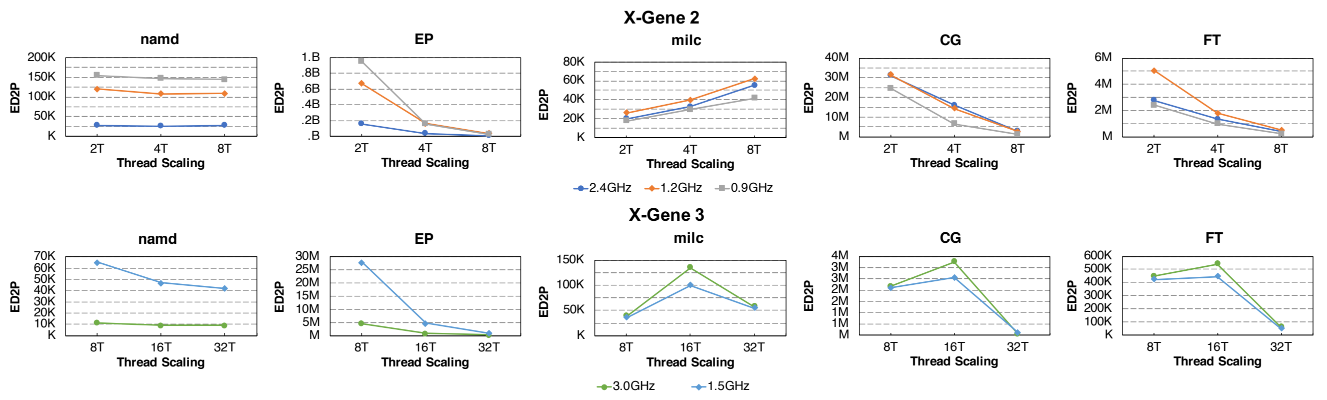

Figure 12. Energy Delay Squared Product (ED2P) for 8, 4, and 2 threading options for 3 different frequencies (2.4GHz, 1.2GHz, 0.9GHz) of X-Gene 2 (top) and 32, 16, and 8 threading options for 2 different frequencies (3GHz, 1.5GHz) of X-Gene 3 (bottom).

1.5GHz configurations for the X-Gene 3. In the first 2 benchmarks (*namd*, *EP*) in both X-Gene 2 and X-Gene 3, which are the most CPU-intensive benchmarks, we can see that the higher the frequency, the most efficient (in terms of ED2P) the configuration (i.e. the blue lines in X-Gene 2 and green lines in X-Gene 3 are always lower in all cases). However, we can see that the trend lines are totally different among the 3 memory-intensive benchmarks (*milc*, *CG* and *FT*) and the CPU-intensive ones. We can see that the frequency is inversely proportional to ED2P efficiency for all the thread scaling options (grey lines vs. blue lines in X-Gene 2, and blue lines vs. green lines in X-Gene 3). Our analysis shows that the identification of the program class (CPU vs. memory-intensive) in runtime can guide the selection of the optimal system configuration (frequency and threading) to achieve high energy savings without compromising performance.

VI. Mitigating Energy: A Real System Implementation

A. Online Monitoring Daemon

According to our study and observations, we developed a simple online monitoring daemon which encapsulates all these conditions and constraints as we described in previous sections and guides process placement, core frequency and supply voltage to achieve higher energy savings on both X-Gene 2 and X-Gene 3 platforms. The daemon has two main functionalities: monitoring and placement. The monitoring part of the daemon acts as a watchdog, which periodically monitors the utilized PMDs (which correspond to the droop magnitude shown in Table II) and the L3C accesses of each running process (except for the system processes). For each process, it counts the L3C accesses during 1M cycles (this actually varies from 300ms to 500ms in our systems; it depends on the IPC rate of each process) and if the L3C accesses are more than 3K (see Figure 9), then it classifies this process as memory-intensive, otherwise it classifies it as CPU-intensive. Moreover, it classifies the processes according to utilized PMDs in order to estimate the current V_{min}.

The second part (Figure 13) of the daemon is the Placement. Figure 13 presents how the system reacts to a process list change and how the placement function of the daemon operates. Placement is the part that places the processes to the CPU cores (or migrate them) and adjusts the voltage and frequencies accordingly. As we presented earlier, each group of core allocation options corresponds to a specific droop magnitude class (Table II) with a distinct safe V_{min} for each frequency. Moreover, as shown in Figure 6, for each core allocation option, all programs produce the same maximum droop magnitude. Given that, we do not use any sophisticated mechanism for predicting the safe V_{min} because the prediction schemes for V_{min} that have been proposed in the literature are error-prone (e.g. [5], [6], [18], [25]–[31]) and can lead to system failures in real microprocessors. To this end, the daemon has been equipped with a fail-safe mechanism: either before the process(es) are invoked or before the frequency should be increased in one or more PMDs (i.e.: a CPU-bound process), the daemon first increases the voltage to the next safe V_{min} level (see Table II) and then, if the voltage can be decreased according to utilized PMDs (this information is provided by the monitoring part), the

TABLE II
Correlation of Voltage Droops Magnitude With Frequency And Core Allocation (V_{min} Columns Concern X-Gene 3).

Droop Magnitude	Utilized PMDs	Thread Scaling	V_{min} @ 3GHz	V_{min} @ 1.5GHz
[25mV, 35mV)	1, 2 PMDs	1T, 2T, 4T(clustered)	780 mV	770 mV
[35mV, 45mV)	4 PMDs	8T(clustered), 4T(spreaded)	800 mV	780 mV
[45mV, 55mV)	8 PMDs	16T(clustered), 8T(spreaded)	810 mV	790 mV
[55mV, 65mV)	16 PMDs	32T, 16T(spreaded)	830 mV	820 mV

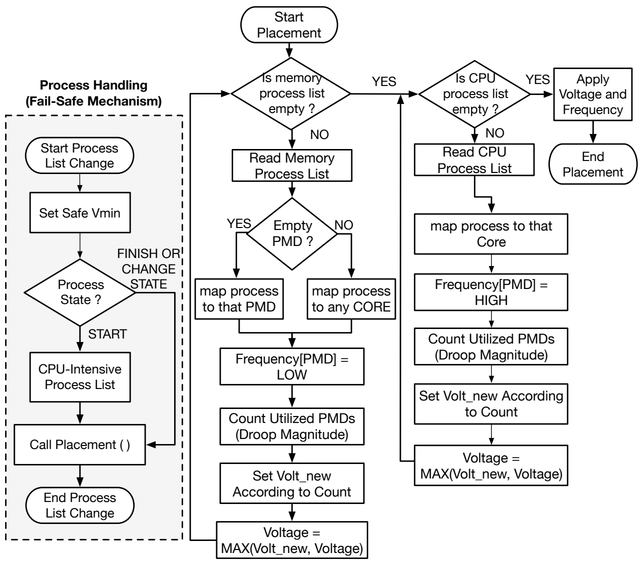
Process Handling (Fail-Safe Mechanism)
Start Process List Change
Set Safe Vmin
Process State ?
FINISH OR CHANGE STATE
START
CPU-Intensive Process List
Call Placement ()
End Process List Change
Start Placement
Is memory process list empty ?
YES
NO
Read Memory Process List
Empty PMD ?
YES
NO
map process to that PMD
map process to any CORE
Frequency[PMD] = LOW
Count Utilized PMDs (Droop Magnitude)
Set Volt_new According to Count
Voltage = MAX(Volt_new, Voltage)
Is CPU process list empty ?
YES
NO
Apply Voltage and Frequency
End Placement
Read CPU Process List
map process to that Core
Frequency[PMD] = HIGH
Count Utilized PMDs (Droop Magnitude)
Set Volt_new According to Count
Voltage = MAX(Volt_new, Voltage)

Figure 13. Process handling and Placement flow chart.

daemon will set the voltage accordingly (as shown in Figure 13). By following this policy, there is a minimal increase of the total power consumption, however, this guarantees the reliable execution on a real system. The Placement part uses the classification performed by the Monitoring part to guide its decisions, and is invoked upon every process list or classification change.

The online monitoring daemon is minimally intrusive and has no impact on the safe V_{min}. Its performance overhead is also negligible, as it is only running periodically to read the performance counters and upon every process list change, to invoke the placement process, which has equal impact as a process migration of the Linux kernel. The daemon is invoked only after (a) either a new process is issued to the system or when a process finishes its execution (to check if a process migration is required), or (b) when a process changes its state (from CPU-intensive to memory-intensive and vice versa). In case (a), it reads the process mapping table and estimates the result, and in case (b), it periodically counts the L3C accesses (as we described in subsection IV-B). Note that, in case (b) the utilized PMDs cannot be changed. Utilized PMDs can only be changed when a new process is invoked, or when a process finishes its execution. To count the L3C accesses, we leverage the built-in performance monitoring counters of the microprocessors. To do so, we developed a kernel module able to provide access to the performance counters from user-space in order to be part of our online monitoring daemon. It is lightweight (thus fast) because it acts in the kernel space and provides near zero overhead to the total microprocessor's operation. We did not use tools like Perf [20] or PAPI [32] because these tools impose an extra overhead in measurements ($\pm 3\%$), while we need very accurate values to take correct decisions. To count the L3C accesses, only one read of one PMU counter and one read of the same register after 1M cycles are required. Afterwards, the kernel module subtracts these two values to produce the final result.

B. Evaluation Results

For the purposes of our experimental evaluation, we also developed a "workload generator" which creates a typical server workload from a "pool" of programs (which includes all the 29 SPEC CPU2006 and the 6 NPB benchmarks; in total 35 different programs). The generator can generate workloads of configurable duration by randomly selecting benchmarks from this pool and randomly defining the timeslot in which each benchmark will be invoked. The workload includes heavy load periods, average load periods and light periods, including also a few idle periods, resembling a typical server workload on a given time window. The generator is configured to guarantee that the number of active processes is never more than the available cores of the microprocessor. The generated workload can be then invoked multiple times, allowing multiple experiments under the same load conditions, using different policies or configurations. Our exploration has revealed potential energy efficiency improvements by adjusting the CPU voltage, frequency and the core allocation. In this subsection we present the results for a simple realistic experimentation we performed in our two systems. To provide detailed results for a comprehensive analysis, we ran 4 different configurations for the same workload sequence:

Baseline: in this configuration we run the workload sequence with the default microprocessor's frequency scaling and scheduler settings (ondemand governor is enabled).

Safe V_{min}: in this configuration we change the nominal voltage of the microprocessor to the safe V_{min}, according to Table II, (again with ondemand governor enabled). With this configuration we evaluate the impact of pessimistic voltage guardbands in energy.

Placement: in this configuration we run the simple online monitoring daemon to perform the proposed frequency and core allocation (the ondemand governor is now disabled), but keeping the voltage at its nominal value. With this configuration we evaluate the impact of the proposed frequency and core allocation options in energy and performance.

Optimal: in this configuration we run the simple online monitoring daemon and adapting the voltage and frequency settings (with ondemand governor disabled). This is the best scenario which integrates all the conditions targeting the best energy efficiency.

Figure 14 shows the average power and Figure 15 the average system load, as well as the number of running processes for a 1-hour execution of randomly generated workload for the default microprocessor's settings (Baseline) and the Optimal scheme, for X-Gene 3 (we omit the corresponding X-Gene 2 graph due to space limitations, however, the observations are the same). We generated 2 workloads, one for X-Gene 2 using a maximum constraint of 8 cores, and one for X-Gene 3 with a constraint of 32 cores. Each workload was executed under both Baseline and Optimal settings.

In Figure 14 that compares the Baseline and the Optimal configuration, we can see that the average power consumption for the optimal configuration compared to the default one (for the same 1-hour workload) is significantly reduced. The average system load, shown in Figure 15 is presented as a moving average of 1 minute with samples of 1 second. As presented in Figure 15, the system has a load with phases of high utilization and others with low utilization, resembling a typical server workload. Moreover, in Figure 15 we can see how many memory-intensive (orange line) and CPU-intensive (dark line) processes are running for each second of the 1-hour execution. We can see that some peaks reach the maximum capability of the system, indicating that the system was occasionally pushed towards its limits.

In Table III and Table IV we compare the 4 different configurations described above. We can see a significant reduction in the total energy: 25.2% in X-Gene 2 (Table III) and 22.3% in X-Gene 3 (Table IV). This was achieved without compromising the performance of CPU-intensive programs, while slightly reducing the performance of memory-intensive programs, as described in Section III, targeting the highest ED2P efficiency. The use of ED2P metric guarantees that the selected policies do not violate high performance constraints and at the same time, significantly reduce the energy consumption. The Optimal scheme with the online monitoring daemon did also slightly shift the completion time of the workload as a result of the small performance impact on some memory-intensive programs (as described in Section III). The total time was shifted by 3.2% in X-Gene 2 and 2.5% in X-Gene 3. We can also notice in Table III and Table IV that the frequency and core allocation options (Placement) are the major contributors on the total energy reduction (18.3% in X-Gene 2 and 13.4% in X-Gene 3), compared to the voltage reduction with the ondemand governor (Safe V_{min}). In any case, the Optimal scheme can achieve more than 22% energy savings with a minimal performance penalty.

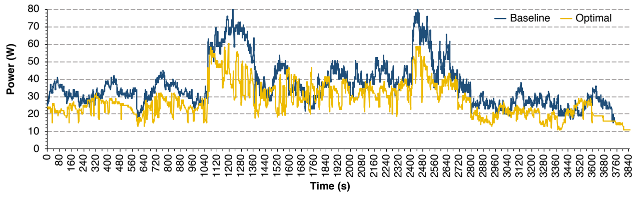

Figure 14. Average power for the Baseline and the Optimal configurations in X-Gene 3 during 1-hour execution.

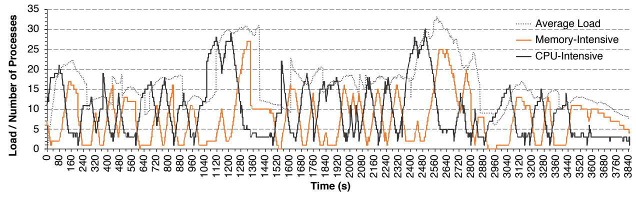

Figure 15. Average System Load (grey dotted line) and number of running processes. The dark line represents the CPU-intensive processes and the orange line the memory-intensive ones.

VII. Related Work

Characterization: Bacha *et al.* [1], [2] focus on monitoring the hardware-reported errors manifested in the caches of a commercial Intel Itanium processor during the execution of benchmarks in off-nominal voltage conditions. Authors in [3], [5], [18], [19], and [31] measure single-core voltage margins in several commercial microprocessor chips to study not only the pessimistic voltage guardbands of the chips, but also the core-to-core and chip-to-chip variations for single-core executions. Sasaki *et al.* [23], study the prevalence of power capping when multiple processes in a multicore microprocessor compete for power, while the power management system attempts to mitigate the contention (reduce the power consumption) by slowing down the processor. Several characterization studies have been presented for off-nominal voltage conditions operation of commercial microprocessor chips with up to 8 cores (e.g.: [1], [2], [6] [18], [19], [25], [31]–[35]). In this paper we present the first work characterizing multi-threaded workloads on high-end microprocessors with significantly larger core counts (up to 32-core). To our knowledge, the observations we make for the safe V_{min}, voltage droop magnitude and the reduced workload variation in actual hardware (not simulated models) of 8-core and 32-core microprocessors, have not been presented in the literature before.

TABLE III
X-Gene 2 Results for the 4 Configurations.

	Baseline	Safe V_{min}	Placement	Optimal
Time (s)	3707	3707	3829	3829
Avg. Power (W)	6.90	6.10	5.46	5.00
Energy (J)	25578.30	22612.07	20906.34	19145.00
Energy Savings	–	11.6%	18.3%	25.2%
ED2P (workload)	351 x 10^9	311 x 10^9	307 x 10^9	281 x 10^9
ED2P Savings	–	11.6%	12.8%	20.1%

TABLE IV
X-Gene 3 Results for the 4 Configurations.

	Baseline	Safe V_{min}	Placement	Optimal
Time (s)	3748	3748	3846	3846
Avg. Power (W)	36.49	32.51	30.78	27.63
Energy (J)	136773.26	121847.48	118379.88	106283.56
Energy Savings	–	10.9%	13.4%	22.3%
ED2P (workload)	19 x 10^{11}	17 x 10^{11}	17 x 10^{11}	15 x 10^{11}
ED2P Savings	–	10.9%	8.9%	18.2%

Voltage Noise Characterization and Mitigation: Ketkar *et al.* [36] and Kim *et al.* [37], [38] propose methods to maximize voltage droops in single core and multicore chips in order to investigate their worst-case behavior due to the generated voltage noise effects. Bertran *et al.* [39] present a voltage noise characterization study in multi-core microprocessors, in which they use a systematic methodology to generate noise stressmarks. Studies of Gupta *et al.* [40] and Reddi *et al.* [30] focus on the prediction of critical execution points of benchmarks, in which large voltage noise glitches are likely to occur, leading to system malfunctions. In the same context, several studies either in the hardware or in the software level have been presented to mitigate the effects of voltage noise [17], [35], [41]–[47] or to recover from them after their occurrence [48]. Some works [49]–[52] focus on the development of electrical simulation framework for power-delivery analysis Reddi *et al.* [17] show that different workloads have different voltage margins and different voltage droop activity based on a 2-core microprocessor. Unlike these previous studies, in this work we are based on 8-core and 32-core microprocessors and show that as the number of active cores increases (4 or more active cores), the emergency voltage droops are massive and lead to workload-independent V_{min}.

Workload Scheduling and DVFS: Energy Aware Scheduling (EAS) project [53], is trying to solve power-management limitations on big.LITTLE designs by balancing the load across all CPUs, while saving power by scaling down the frequency of the CPUs or idling them. Li and Martinez [54] optimize a parallel workload running on a simulated 16-core microprocessor by dynamically changing the number of processors and the V/F levels. Isci *et al.* [55] consider a simulated 4-core microprocessor with per-core DVFS and examine different DVFS policies for high performance and power efficiency. Their solutions are not scalable to large systems, and to this end, Teodorescu and Torrellas [56] consider a simulated 20-core microprocessor, again assuming a per-core DVFS approach, to propose variation-aware algorithms for power management. However, in this paper we show with actual measurements on multicore microprocessors that, as the number of active threads increases, there is a minimal variation across different workloads and cores. Vega *et al.* [57] propose a coordinated power management algorithm, which dynamically detects situations in which the program may be bound by single-thread-performance or throughput and actuates the power management accordingly. Miftakhutdinov *et al.* in [58] propose a low-cost mechanism that accurately predicts the performance impact of frequency scaling in the presence of a realistic memory system. Castillo *et al.* in [59] propose a hardware mechanism that dynamically adapts the computational power of a task depending on its criticality. Authors in [21], [22] have exposed the phases of a program which are memory intensive and proposed this feature as a proxy in a single-core microprocessor to guide the DVFS to reduce the frequency in that specific program phases. Unlike these previous studies, in this paper, we are based on real implementations of 8-core and 32-core microprocessors, whose frequency can be set per pair of cores and all cores have the same voltage, and present a different approach for V/F settings, according to workload characteristics (not program phases) and voltage droop magnitude, for the dynamic scheduling and migration of the programs on the available microprocessor cores.

VIII. Conclusion

We have presented a comprehensive analysis of how different process placement and V/F settings can deliver different energy and ED2P behavior of multithreaded workloads on two state-of-the-art multicore server-grade CPUs: 8-core X-Gene 2 and 32-core X-Gene 3. We explored how optimal energy consumption and ED2P values can be achieved and what are the main factors that affect these metrics the most. All our measurements are on actual server hardware and we report the variability among the different chips and workloads. Our analysis findings were employed in a lightweight online monitoring daemon which can decide the optimal core allocation and frequency to achieve higher energy efficiency by monitoring the utilized PMDs (which affect the droop emergency events) and L3C accesses. We show that as the number of threads increases, the workload V_{min} variation is virtually eliminated and we explain the reason of this behavior by showing that in multicore executions all programs have the same maximum voltage droop magnitude for different core allocation and frequency settings. Our system-level evaluation demonstrates that these findings can deliver on average 25.2% energy savings on X-Gene 2, and 22.3% energy savings on X-Gene 3.

Acknowledgment

This work is funded by the H2020 Framework Program of the European Union through the UniServer Project (Grant Agreement 688540) — http://www.uniserver2020.eu.

References

[1] A. Bacha and R. Teodorescu, “Dynamic reduction of voltage margins by leveraging on-chip ecc in itanium ii processors,” in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA ’13, (New York, NY, USA), pp. 297–307, ACM, 2013.

[2] A. Bacha and R. Teodorescu, “Using ecc feedback to guide voltage speculation in low-voltage processors,” in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, (Washington, DC, USA), pp. 306–318, IEEE Computer Society, 2014.

[3] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, C. Magdalinos, and D. Gizopoulos, “Voltage margins identification on commercial x86-64 multicore microprocessors,” in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, jul 2017.

[4] A. Bacha and R. Teodorescu, "Authenticache: Harnessing cache ecc for system authentication," in *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, (New York, NY, USA), pp. 128–140, ACM, 2015.

[5] M. Kaliorakis, A. Chatzidimitriou, G. Papadimitriou, and D. Gizopoulos, "Statistical analysis of multicore CPUs operation in scaled voltage conditions," *IEEE Computer Architecture Letters*, vol. 17, pp. 109–112, jul 2018.

[6] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, D. Gizopoulos, P. Lawthers, and S. Das, "Harnessing voltage margins for energy efficiency in multicore cpus," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, (New York, NY, USA), pp. 503–516, ACM, 2017.

[7] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, HotPower'10, (Berkeley, CA, USA), pp. 1–8, USENIX Association, 2010.

[8] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "Rapl: Memory power estimation and capping," in *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '10, (New York, NY, USA), pp. 189–194, ACM, 2010.

[9] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, (New York, NY, USA), pp. 13–23, ACM, 2007.

[10] L. Gwennap, *Performance Arms X-Gene 3 for Cloud*, 2017 (Accessed: July 25, 2018). http://www.linleygroup.com/uploads/x-gene-3-for-cloud.pdf.

[11] *NAS Parallel Benchmarks Suite*, v3.3.1. https://www.nas.nasa.gov/publications/npb.html.

[12] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, pp. 1–17, Sept. 2006.

[13] 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 '08, (New York, NY, USA), pp. 72–81, ACM, 2008.

[14] B. Lepers, V. Quéma, and A. Fedorova, "Thread and memory placement on numa systems: Asymmetry matters," in *Proceedings of the 2015 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '15, (Berkeley, CA, USA), pp. 277–289, USENIX Association, 2015.

[15] M. Curtis-Maury, *Improving the Efficiency of Parallel Applications on Multithreaded and Multicore Systems*. PhD thesis, Virginia Tech, 2008.

[16] *Advanced Configuration and Power Interface (ACPI) Specification*, 2014 (Accessed: Aug 1, 2018). https://www.uefi.org/sites/default/files/resources/ACPI_5_1release.pdf.

[17] V. J. Reddi *et al.*, "Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling," in *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE, dec 2010.

[18] G. Papadimitriou, A. Chatzidimitriou, M. Kaliorakis, Y. Vastakis, and D. Gizopoulos, "Micro-viruses for fast system-level voltage margins characterization in multicore CPUs," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, apr 2018.

[19] G. Karakonstantis *et al.*, "An energy-efficient and error-resilient server ecosystem exceeding conservative scaling limits," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, mar 2018.

[20] *Perf: Linux Profiling with Performance Counters*. https://perf.wiki.kernel.org/index.php/Main_Page.

[21] C. Isci, G. Contreras, and M. Martonosi, "Live, runtime phase monitoring and prediction on real systems with application to dynamic power management," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, (Washington, DC, USA), pp. 359–370, IEEE Computer Society, 2006.

[22] Q. Wu *et al.*, "A dynamic compilation framework for controlling microprocessor energy and performance," in *Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 38, (Washington, DC, USA), pp. 271–282, IEEE Computer Society, 2005.

[23] H. Sasaki, A. Buyuktosunoglu, A. Vega, and P. Bose, "Characterization and mitigation of power contention across multiprogrammed workloads," in *2016 IEEE International Symposium on Workload Characterization (IISWC)*, IEEE, sep 2016.

[24] D. M. Brooks *et al.*, "Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors," *IEEE Micro*, vol. 20, pp. 26–44, Nov. 2000.

[25] J. Leng, A. Buyuktosunoglu, R. Bertran, P. Bose, and V. J. Reddi, "Safe limits on voltage reduction efficiency in gpus: A direct measurement approach," in *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, (New York, NY, USA), pp. 294–307, ACM, 2015.

[26] W. Jia, K. A. Shaw, and M. Martonosi, "Stargazer: Automated regression-based GPU design space exploration," in *2012 IEEE International Symposium on Performance Analysis of Systems & Software*, IEEE, apr 2012.

[27] P. Joseph, K. Vaswani, and M. Thazhuthaveetil, "Construction and use of linear regression models for processor performance analysis," in *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, IEEE.

[28] B. C. Lee and D. M. Brooks, "Accurate and efficient regression modeling for microarchitectural performance and power prediction," in *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XII, (New York, NY, USA), pp. 185–194, ACM, 2006.

[29] A. Biswas, N. Soundararajan, S. S. Mukherjee, and S. Gurumurthi, "Quantized AVF: A means of capturing vulnerability variations over small windows of time," in *Silicon Errors in Logic - System Effects (SELSE)*, 2009.

[30] 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 *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, IEEE, feb 2009.

[31] K. Tovletoglou *et al.*, "Measuring and exploiting guardbands of server-grade ARMv8 CPU cores and DRAMs," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, IEEE, jun 2018.

[32] P. J. Mucci, S. Browne, C. Deane, and G. Ho, "Papi: A portable interface to hardware performance counters," in *In Proceedings of the Department of Defense HPCMP Users Group Conference*, pp. 7–10, 1999.

[33] C. R. Lefurgy *et al.*, "Active management of timing guardband to save energy in power7," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, (New York, NY, USA), pp. 1–11, ACM, 2011.

[34] S. Sundaram *et al.*, "Adaptive voltage frequency scaling using critical path accumulator implemented in 28nm CPU," in *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, IEEE, jan 2016.

[35] Y. Zu, C. R. Lefurgy, J. Leng, M. Halpern, M. S. Floyd, and V. J. Reddi, "Adaptive guardband scheduling to improve system-level efficiency of the power7+," in *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, (New York, NY, USA), pp. 308–321, ACM, 2015.

[36] M. Ketkar and E. Chiprout, "A microarchitecture-based framework for pre- and post-silicon power delivery analysis," in *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, (New York, NY, USA), pp. 179–188, ACM, 2009.

[37] Y. Kim and L. K. John, "Automated di/dt stressmark generation for microprocessor power delivery networks," in *IEEE/ACM International Symposium on Low Power Electronics and Design*, IEEE, aug 2011.

[38] Y. Kim *et al.*, "Audit: Stress testing the automatic way," in *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-45, (Washington, DC, USA), pp. 212–223, IEEE Computer Society, 2012.

[39] R. Bertran *et al.*, "Voltage noise in multi-core processors: Empirical characterization and optimization opportunities," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, (Washington, DC, USA), pp. 368–380, IEEE Computer Society, 2014.

[40] 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 '09, (3001 Leuven, Belgium, Belgium), pp. 160–165, European Design and Automation Association, 2009.

[41] M. S. Gupta, K. K. Rangan, M. D. Smith, G.-Y. Wei, and D. Brooks, "Towards a software approach to mitigate voltage emergencies," in *Proceedings of the 2007 international symposium on Low power electronics and design - ISLPED 2007*, ACM Press, 2007.

[42] R. Joseph, D. Brooks, and M. Martonosi, "Control techniques to eliminate voltage emergencies in high performance processors," in *The Ninth International Symposium on High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings.*, IEEE Comput. Soc.

[43] T. N. Miller, R. Thomas, X. Pan, and R. Teodorescu, "Vrsync: Characterizing and eliminating synchronization-induced voltage emergencies in many-core processors," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, (Washington, DC, USA), pp. 249–260, IEEE Computer Society, 2012.

[44] M. D. Powell and T. N. Vijaykumar, "Pipeline muffling and a priori current ramping: Architectural techniques to reduce high-frequency inductive noise," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, ISLPED '03, (New York, NY, USA), pp. 223–228, ACM, 2003.

[45] J. Leng, Y. Zu, and V. J. Reddi, "GPU voltage noise: Characterization and hierarchical smoothing of spatial and temporal voltage noise interference in GPU architectures," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, feb 2015.

[46] R. Thomas, K. Barber, N. Sedaghati, L. Zhou, and R. Teodorescu, "Core tunneling: Variation-aware voltage noise mitigation in GPUs," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, mar 2016.

[47] R. Thomas, N. Sedaghati, and R. Teodorescu, "EmerGPU: Understanding and mitigating resonance-induced voltage noise in GPU architectures," in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, apr 2016.

[48] 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 *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, IEEE, feb 2008.

[49] P. N. Whatmough, S. Das, Z. Hadjilambrou, and D. M. Bull, "14.6 an all-digital power-delivery monitor for analysis of a 28nm dual-core ARM cortex-a57 cluster," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, IEEE, feb 2015.

[50] P. N. Whatmough, S. Das, and D. M. Bull, "Analysis of adaptive clocking technique for resonant supply voltage noise mitigation," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, IEEE, jul 2015.

[51] S. Das, P. Whatmough, and D. Bull, "Modeling and characterization of the system-level power delivery network for a dual-core ARM cortex-a57 cluster in 28nm CMOS," in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, IEEE, jul 2015.

[52] P. N. Whatmough, S. Das, Z. Hadjilambrou, and D. M. Bull, "Power integrity analysis of a 28 nm dual-core ARM cortex-a57 cluster using an all-digital power delivery monitor," *IEEE Journal of Solid-State Circuits*, vol. 52, pp. 1643–1654, jun 2017.

[53] V. Wool, *EAS – Energy Aware Scheduler: An unbiased look*, (Accessed: Oct 11, 2018). https://elinux.org/images/6/69/Eas-unbiased1.pdf.

[54] J. Li and J. Martinez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," in *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, IEEE.

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

[56] R. Teodorescu and J. Torrellas, "Variation-aware application scheduling and power management for chip multiprocessors," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, (Washington, DC, USA), pp. 363–374, IEEE Computer Society, 2008.

[57] A. Vega, A. Buyuktosunoglu, H. Hanson, P. Bose, and S. Ramani, "Crank it up or dial it down: Coordinated multiprocessor frequency and folding control," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-46, (New York, NY, USA), pp. 210–221, ACM, 2013.

[58] R. Miftakhutdinov, E. Ebrahimi, and Y. N. Patt, "Predicting performance impact of dvfs for realistic memory systems," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-45, (Washington, DC, USA), pp. 155–165, IEEE Computer Society, 2012.

[59] E. Castillo *et al.*, "CATA: Criticality aware task acceleration for multicore processors," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, IEEE, may 2016.