



Dynamic workload-aware DVFS for multicore systems using machine learning

Manjari Gupta¹ · Lava Bhargava¹ · S. Indu² 

Received: 28 January 2020 / Accepted: 24 September 2020 / Published online: 8 October 2020
© Springer-Verlag GmbH Austria, part of Springer Nature 2020

Abstract

With growing heterogeneity and complexity in applications, demand to design an energy-efficient and fast computing system in multi-core architecture has heightened. This paper presents a regression-based dynamic voltage frequency scaling model which studies and utilizes workload characteristics to obtain optimal voltage–frequency (v – f) settings. The proposed framework leverages the workload profile information together with power constraints to compute the best-suited voltage–frequency (v – f) settings to (a) maintain global power budget at chip-level, (b) maximize performance while enforcing power constraints at the per-core level. The presented algorithm works in conjunction with the workload characterizer and senses change in application requirements and apply the knowledge to select the next setting for the core. Our results when compared with two state-of-the-art algorithms *MaxBIPS* and *TPEq* achieve the average power reduction of 33% and 25% respectively across 32-core architecture for PARSEC benchmarks.

Keywords Dynamic voltage frequency scaling · Workload decomposition · Multicore processors · Energy-performance tradeoff · Machine learning

Mathematics Subject Classification 62J05 · 68M20

✉ S. Indu
s.indu@dce.ac.in

Manjari Gupta
2015rec9032@mnit.ac.in

Lava Bhargava
lavab@mnit.ac.in

¹ Malaviya National Institute of Technology, Jaipur, India

² Delhi Technological University, Delhi, India

1 Introduction

There has been an upsurge in the demand for battery-operated devices due to their widespread applications and increased usability across various sectors. Minimizing power consumption and maximizing performance is rapidly becoming a fundamental customer requirement. This has an added advantage of the improvement in chip's reliability and hence longer lifetime which has been an additional concern.

Dynamic voltage frequency scaling (DVFS) has been universally adopted as a low-power technique while fulfilling the performance requirements. Modern-day processors like Intel XScale, Transmeta Crusoe, and AMD Athlon are equipped with in-built DVFS capability [1]. The main objective of DVFS is to supply "just enough" circuit speed for processing system workload whilst attaining the desired throughput and minimizing energy consumption simultaneously [2]. Since the power consumption of a processor is cubically reliant on the operational frequency ($E \propto \text{Capacitance} \times \text{voltage}^2 \times \text{frequency} \times \text{cycles}$), management of clock frequency directly results in energy-savings. Figure 1 illustrates the effect of DVFS on power consumption and execution time of a workload [3]. With no DVFS applied, let t_1 be the time a task takes to complete at the highest frequency (f_1). Let P_{sysf_1} represent the power consumed by this task. P_{idlef_1} signify the CPU/core power consumption in idle state. When DVFS is applied ($f_2 \rightarrow f_1$), the power consumption is curtailed to P_{sysf_2} while the task execution-time amplifies to t_2 . The new task completion time t_2 has an added component of t_{delay} which is a result of the reduction in operating frequency ($f_1 > f_2$) which directly influences the of power-savings. However, the change in frequency does not have a straight forward impact on execution time, which also depends on how the application utilizes the system resources.

Further breaking the system power consumption (P_{sysf_1}) into CPU power (P_{cf_1}) and power consumed by other devices P_d at voltage-frequency setting f_1 , the energy-savings due to DVFS implementation can be represented as Eq. 1:

$$\begin{aligned} E_{DVFS} &= P_{sysf_1}t_1 + P_{idlef_1}t_{delay} - P_{sysf_2}(t_1 + t_{delay}) \\ &= (P_{sysf_1} - P_{sysf_2})t_1 - (P_{sysf_2} - P_{idlef_1})t_{delay} \end{aligned}$$

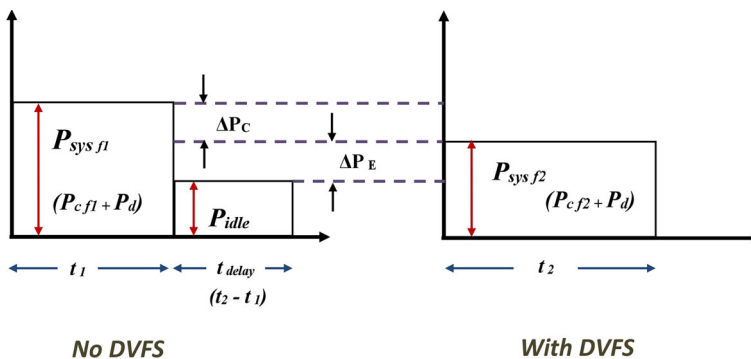


Fig. 1 Impact of DVFS on power consumption and execution-times [3]

$$\begin{aligned}
&= (P_{cf_1} - P_{cf_2})t_1 - (P_{cf_2} + P_d - P_{c_{idle}} - P_{d_{idle}})t_{delay} \\
&= (P_{cf_1} - P_{cf_2})t_1 - ((P_{cf_2} - P_{c_{idle}}) + (P_d - P_{d_{idle}}))t_{delay} \\
&= \Delta P_C t_1 - \Delta P_E t_{delay} \\
&= E_R - E_E \tag{1}
\end{aligned}$$

$\Delta P_C = P_{cf_1} - P_{cf_2}$; represents the scale down in CPU power when frequency setting is changed from f_1 to f_2 . $\Delta P_E = (P_{cf_2} - P_{c_{idle}}) + (P_d - P_{d_{idle}})$; represents the additional energy budget that DVFS brings into the system. It is evident that this extra power can be consumed at the CPU level (P_C) or device resource level (P_d). For memory-driven operations, the value of $P_d - P_{d_{idle}}$ is very high as the system makes frequent trips to memory keeping CPU idle. On the contrary, this value is negligible for CPU-bound workloads. For DVFS to provide energy savings, $E_R > E_E$ is a necessary condition. If this inequality is not satisfied, the energy dissipation of the system operating at f_2 will be greater than the energy consumption at the highest frequency of f_1 . This defeats the purpose of DVFS and additionally results in performance overhead (t_{delay}).

Motivation: The main aim of our work is the joint optimization of IPC and Power which are directly affected by frequency/voltage, thus, DVFS has been chosen. It is a well-known mechanism to make the processor's power to converge to a specified power budget [26]. The applications are a combination of compute-intensive and memory-intensive tasks. Offline analysis of the application misses the opportunity of energy-savings [23]. A run-time workload classifier will be beneficial in adapting to workload variations. A dynamically calculated metric indicating the change in workload will ensure that frequency is scaled only when needed. Reading and evaluating dynamic information from a monitoring system imposes performance overhead and should be minimized. A prediction model can exploit the inherent similarities in the applications. An offline-analysis of performance and power for all possible thread-core mapping corresponding to varying v-f settings will help in identifying the repeated or similar trends. A computationally-inexpensive prediction model such as linear regression is desirable. It constructs a predictor which once trained through training data/programs is capable of handling unseen programs and utilize runtime statistics efficiently. The effectiveness of DVFS varies with the level of granularity. Per-chip DVFS has a single control knob which diminishes the potency of frequency scaling and also restricts extensibility. On the other hand, per-core DVFS supports a wide range of control knobs which facilitate high flexibility but are difficult to design. Figure 2 gives a snapshot of various power supply configurations which depict how granularity is handled in DVFS [4]. The conventional design strategy is portrayed in Fig. 2a where a single control knob is provided as an off-chip regulator. A 2-step voltage switching configuration is sketched in Fig. 2b. To account for inherent degradation in conversions, the off-chip regulator steps-down voltage from 3.7 V to 1.8 V. This 1.8 V supply voltage wheels the on-chip regulator which further steps down the voltage to a range of 0.6–1 V and distributes it across the 4-cores. The last configuration (Fig. 2c) extends Fig. 2b by providing individual on-chip regulators for each core. The first two configurations are *coarse-grained* while the last one is *fine-grained* DVFS. It is observed that with the augmentation in the number of cores, per-core DVFS holds the key to low-power design in the future.

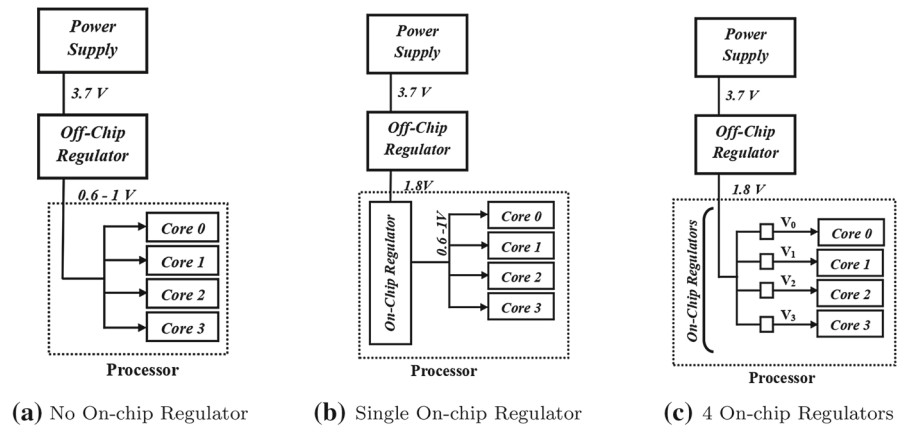


Fig. 2 Power-supply organizations for a quad-core system [4]

To ensure deadline-conformance, the DVFS technique should have prior knowledge of tasks such as arrival time, execution time, priority, etc. This critical information is relatively difficult to obtain and store in limited space-constrained embedded systems. The proposed DVFS system is designed to control the selection of best suited v - f setting (amongst the set of predefined values) based on task characteristics. The runtime statistics of application and architecture are gathered to analyze the task model. The algorithm is responsible to accurately characterize task behavior and then apply optimal v - f values while accommodating the global power budget. The fine-grained power manager provides modification at a per-core level while the coarse-grained allocator ensures there is no power budget overshoot. The prediction-based controller chooses to increase/decrease the frequency to the next specified value based on the prediction model. Reduction in operational frequency leads to increment in the makespan of a task, affecting the real-time performance of the system. Thus, achieving a balance between energy and performance becomes essential while making DVFS decisions.

Section 2 highlights the road map of DVFS techniques explored in the past. It also lists the paper contributions. Details of the methodology followed and the algorithms used are explained in Sect. 3. Experimental implementation and analysis of the obtained results are mentioned in Sect. 4. Finally, the paper is concluded in Sect. 5.

2 Related work and paper contribution

Low-power design techniques using DVFS and Dynamic power management (DPM) have been an active research area in the past. Long-researched DVFS approaches can be cataloged into three segments. The first section constitutes methods that perform DVFS at either application-level or are compiler-assisted. Since the operating system-based techniques use varied heuristics, they can only provide control at the coarse-grain level. A regression-based learning model is used in [5] to map tasks on a heterogeneous system. Based on the application performance requirement, it is dynamically allotted

a computing resource. This technique performs an energy-performance tradeoff efficiently but doesn't consider task parallelism. Mapping in conjunction with DVFS is considered jointly in [6] for a heterogeneous architecture. Applications with their timing constraints are represented through the task-graph. It then performs probabilistic task allocation and frequency scaling through heuristics. Since the approach uses offline heuristics, it lacks run-time adjustment to the dynamic nature of the workload.

The second category consists of system-level DVFS techniques that don't consider task characteristics. These approaches depend on the runtime statistics provided by the platform or architecture to replicate application characteristics. An IPC (Instruction per cycle) guided DVFS technique is presented in [1]. A multivariate linear regression model is used to predict IPC and accordingly perform voltage scaling. The algorithm works at a micro-architectural level and chooses from various processor configurations (pipeline gating, in-order, and out-of-order issue) in conformance with performance requirements. To promote portability across applications, performance monitoring counters (PMCs) built into the hardware architecture have been used in [7]. A multinomial logistic regression (MLR) based controller makes thread packing and DVFS control decisions. The inclusion of thread packing with DVFS improves the average power-range considerably. Reinforcement learning based DVFS method in [28] presents power optimization in mobile devices. The method handles scalability by maintaining two Q-tables and two corresponding update functions. This method provides an alternate to supervised learning, however suffers from overestimation.

The third category comprises of DVFS-techniques that use prior known information on the application model to make control decisions. Choi et al. [8] proposed a workload decomposition technique which uses performance counters to divide task into *on-chip* and *off-chip* parts. The on-chip part represents the clock cycles needed for a CPU operation while the off-chip segment signifies memory access. The DVFS control manager reduces the operational frequency for memory-intensive jobs while increasing it for CPU-intensive jobs. The technique can utilize workload characteristics but doesn't consider multi-tasking. Hardware performance counters utilizing the clustering approach to handle large number of cores has been explored in [27]. This approach facilitates scalability but does not correlate application characteristics with observed hardware events. Weissel et al. in [9] use the performance monitoring unit (PMU) to select optimal v-f settings while maintaining the performance. The PMU captures memory access and cache hit & miss ratio statistics to model the dynamic program behavior. Though this process considers multi-tasking, it fails to provide runtime energy-performance tradeoff control. Extending on the same lines, Dhiman et al in [2] presented an online-learning based scheduling method using task characterization. It verifies that CPU-intensive tasks suffer while memory-bound phases benefit when executed at a lower frequency. Since the solution was implemented on a single-core system it is unable to exploit the advantages of DVFS functionality. This paper is an extension to [2] but takes global power budget and per-core power capping into consideration. The regression-model takes the next interval IPC and Power prediction value into consideration along with task characteristics to obtain optimal v-f settings.

The operating system-based techniques use varied heuristics, they can only provide control at the coarse-grain level and not at the fine-grain level. These techniques lack exploiting the full capability of the DVFS technique. With increasing complexity in

applications and variation in task characteristics unaccounted for, system-level DVFS techniques are unable to adopt new applications. DVFS supervision at a fine-grain level doesn't take overall power consumption for the chip into consideration. Our method addresses all these limitations by including both global and per-core budget while considering workload variation collectively.

Further, the proposed method has been compared to two state-of-the-art methods: MaxBIPS [16] and TPEq [17] for power-reduction and performance degradation. Similar to our work, Isci et al. [16] uses workload characteristics to obtain optimal DVFS policy. To induce variation in workload, the method scales memory and L2 access cycles and hard-wires the relations at design-time. Our method provides a novel method for calculation of application characteristics. Unlike the presented method, MaxBIPS only considers single-threaded applications, hence limiting the practical feasibility. Additionally, our method evaluates 6 frequency modes contrary to MaxBIPS which uses only 3 modes. TPEq is an improvement of MaxBIPS and uses thread progress equalization (TPEq) to achieve a power-performance tradeoff. It uses Cycles-per-instruction (CPI) to identify thread imbalance and utilizes it to dynamically select voltage/frequency settings per core. Since the main aim of TPEq is to reduce thread imbalance, it is inclined towards performance improvement rather than reducing power consumption.

Following are the contributions made by this paper:

- Introduce a Global power allocator, which collects per-core power and performance statistics at periodic intervals. It enforces adherence to the chip power budget by assigning optimal operating voltage–frequency (v – f) levels.
- Develop a workload analyzer which characterizes the application-phase as CPU-intensive or Memory-intensive. Since an application can have both these phases, the value is calculated dynamically.
- The presented technique is divided into *on-line* and *off-line* phases, wherein the offline phase machine learning technique (Multi-variate Linear Regression) is used to develop models based on characterization data collected. The best-fitting voltage–frequency settings for each core are determined dynamically using application and architecture statistics.
- We demonstrate that the proposed method successfully reduces power consumption and minimizes performance degradation under power capping. When compared with *MaxBIPS*, 25% improvement in power-reduction and 33% with *TPEq* is achieved.

3 Methodology

3.1 Architecture model

Considering a homogeneous architecture composed of m identical processors $P = P_1, P_2, \dots, P_m$. All processors in P support DVFS and each processor can operate on different voltage/frequency which can be scaled independently. However, we assume that the processor can select only amongst a finite number of discrete operat-

ing frequencies. To understand the impact of frequency on the core’s utilization, we must understand their relationship. Before formulating the problem, we introduce the following notations:

- τ_i : i th task
- e_i : Execution time of the i th task
- e_i^* : Task portion independent of core frequency
- e_i' : Task portion dependent on core frequency
- d_i : Relative deadilne of the i th task
- $U_i(k)$: Utilization of core P_i during k_{th} DVFS epoch k
- $f_i(k)$: Operating frequency of core P_i during k_{th} DVFS epoch

Utilization of P_i core at fixed frequency $f_i(k)$ is given by [10]:

$$U_i(k) = \sum_{\tau_j \in k} \frac{e_j}{d_j} = \sum_{\tau_j \in k} \frac{e_j^* + e_j' / f_i(k)}{d_j} \tag{2}$$

Rewriting 2 in terms of frequency changes where $F_j(k)$ is clock-period($1/f_j(k)$).

$$U_i(k + 1) = U_i(k) + g_j(k) X \Delta F_j(k) \tag{3}$$

where $\Delta F_j(k) = F_j(k + 1) - F_j(k)$.

Equation 3 clearly shows that transition in the frequency has a significant effect on the utilization of a core. Thus, if we have information about the next interval utilization of $U_i(k + 1)$ of the core, we can manipulate the operating frequency accordingly.

3.2 Application model

An application is a collection of tasks with its inherent properties. Throughput is basically how quickly an application can run on a system. The benchmark selection is a crucial step as it replicates the application model which is one of the input parameters. The characteristics of the chosen workload should be closely related to real-time applications. We have used industry-standard PARSEC benchmark [11] which has a plethora of parallel, multithreaded applications. Table 1 lists the varying characteristics and details of the benchmarks used. A baseline-study for each benchmark has been conducted and detailed in Sect. 4.2.

3.3 DVFS-strategy

The overview of the DVFS framework is depicted in Fig. 3. The presented prediction model uses Linear-regression [21] to predict the next interval IPC and Power values. For analysis, we chose a linear regression model for its simplicity, low computational overhead, and high accuracy even with a small population of data. Additionally, this

Table 1 Characteristics of PARSEC benchmarks [11]

Benchmark name	Working set	Granularity	Load balancing	Locks	Barriers	Data usage	
						Sharing	Exchange
Blackscholes	Small	Coarse	Static	No	Low	Low	Low
Bodytrack	Medium	Medium	Dynamic	High	High	High	Medium
Ferret	Unbounded	Medium	–	High	No	High	High
Swaptions	Medium	Coarse	Static	Low	No	Low	Low
x264	Medium	Coarse	–	High	No	High	High

model is relatively easier to scale when compared with other non-linear or probability-based regression models. The offline-regression learning model uses performance monitoring counters—IPC, Power consumption, and operational frequency as an input parameter. Both IPC and power are directly related to clock rate (cycles per second, given in Hz). IPC has a linear relation with frequency i.e. high frequency leads to high IPC. Similarly, lowering the frequency leads to a reduction in energy dissipation. Since IPC and Power counterbalance the effect of change in frequency, both need to be evaluated jointly. The linear regression model predicts the next interval values for both IPC and Power when considered simultaneously. This facilitates obtaining a solution that optimizes both power and performance. Since frequency is the single control knob for both the optimization parameters, the DVFS manager is chosen. The regression algorithm calculates a function through training data which can predict the value of a dependent variable through a set of independent variables (also known as features). Consider a set of independent variables $x_1, x_2, x_3, \dots, x_n$ and y as dependent variable. Ordinary Least Squares (OLS) is used as the basis of the presented prediction model and calculates the weights β for each feature x and error e using Eq. 4:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \dots + \beta_k x_{ki} + e_i \quad (4)$$

where y_i , i th output ($IPC_{predicted}$ or $Power_{predicted}$); x_{ji} , j th feature ($IPC_{lastinterval}$ or $Power_{lastinterval}$ and frequency f) calculated at i th observation; e_i , i th error; k , number of features.

The goal is to obtain optimal values of weights/coefficients $\beta_1, \beta_2, \beta_3, \dots, \beta_k$ such that the variance between observed and estimated values is minimized.

At the start of the simulation, system configuration for Intel Gainstown (as per Table 2) is set as a Python script in the Sniper simulator. The pre-defined voltage-frequency values are set in the `dvfs.py` file. Benchmarks are run with no DVFS on Sniper to generate core statistics and Power traces from McPAT. Additionally the application statistics such L1 cycles stalled and busy cycles are passed to the python scripted “Workload Characterizer”. Here the value of μ is calculated for each interval per benchmark. The Linear regression prediction model is used to estimate continuous value of next interval IPC ($IPC_{predicted}$) and core-power ($Power_{predicted}$) [15]. To reduce complexity, both IPC and Power values are divided into five bins which have been determined according to their statistical distribution as performed in [14]. This

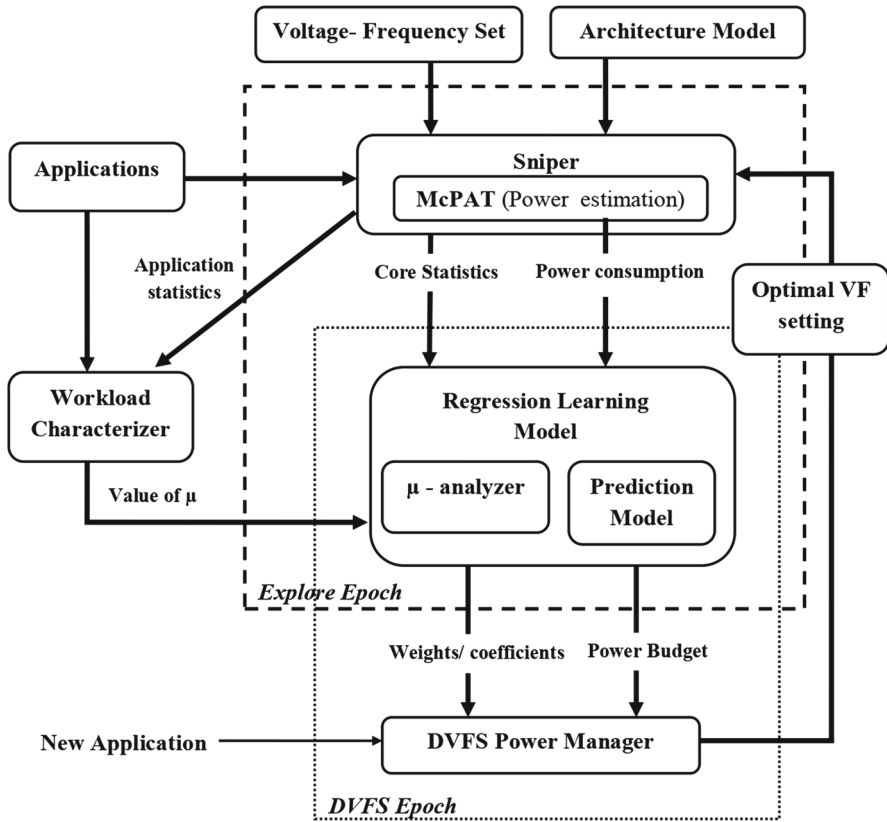


Fig. 3 Overview of the proposed workload-aware DVFS framework

data is grouped as a vector and has been explained in Sect. 3.4. The DVFS power manager decides to increase/decrease the next optimal voltage-frequency setting for the core by taking inputs from *Prediction Model* and μ -analyzer. The component μ -analyzer determines the change in application behavior and sets the flag according to Algorithm 2.

At each *explore epoch* (500 μ s), the chip-wide global power manager is evoked. During this period, the power and performance statistics for the current phase are evaluated. This is to prevent budget overshoot at the chip-level. A coarse-grained power allocator enforces budget adherence based on threshold power calculated dynamically (Sect. 4.3). This chip-wide DVFS manager is light-weight and has low overhead as it is operated at $\frac{1}{5}$ th time interval as compared to DVFS epoch.

At each *DVFS epoch* (100 μ s), the implemented fine-grained power manager is triggered. It is during this time, new power level request (following Algorithm 2) is generated based on performance, core-level power and change in CPU-intensiveness of the application. The termination condition is the completion of benchmark.

The metrics used to invoke the designed Power level request algorithms are Gain and change in μ which are defined in equations 5 and 6 respectively.

$$Gain = \frac{IPC * Clockrate}{Power} \quad (5)$$

$$\Delta\mu = \mu_i(k + 1) - \mu_i(k) \quad (6)$$

Since IPC depends on the type of core, Gain represents instructions executed per unit energy eliminating the effect of core-type from the evaluation. At every DVFS epoch, first Algorithm 1 is called to identify change in application-behavior. As illustrated in equation 3, any fluctuation in expected behavior will trigger frequency scaling. Since the change in the value of μ is a computed value and not an estimated value, it represents the real-time change in application requirement. The μ -analyzer calculates change in workload behavior by comparing the value of μ of last interval ($\mu(k)$) and current interval ($\mu(k + 1)$). Based on these values a *flag* is set to signify the change in CPU-intensiveness, which in turn triggers the change in the v-f setting by the Power Manager.

Algorithm 1 : CPU-intensiveness change

- 1: Input Variables : $\mu_i(k + 1), \mu_i(k)$
 - 2: $flag_i = \begin{cases} -1 & \text{if } \Delta\mu_i < 0 \ \backslash \ \text{Next phase Memory-intensive} \\ 0 & \text{if } \Delta\mu_i = 0 \ \backslash \ \text{No change} \\ +1 & \text{if } \Delta\mu_i > 0 \ \backslash \ \text{Next phase CPU -intensive} \end{cases}$
 - 3: If $flag_i \neq 0$, call Power level request $\backslash \ \text{change in } \Delta\mu_i$
-

The problem-statement has been illustrated in Algorithm 2. The suggested power manager should be able to boost G_{max} (Total gain) whilst ensuring chip-level budget adherence (P_{budget}). The DVFS manager is subjected to two levels of constraints: (1) Select either current power or predicted power under budget constraints; (2) Ensure there is an increase in the expected throughput for the next interval.

Algorithm 2 : Power Level Request

- 1: Input Variables : $P_{current}, P_{predicted}, P_{budget}$
 - 2: Output Variable : P_{change}
 - 3: $x_i = \begin{cases} 0 & P_{current} \\ 1 & P_{predicted} \end{cases}$
 - 4: Maximize $G_{max} = \sum_j G_{next}(j).X_j + \sum_j G_{current}(j).(1 - X_j)$
 - 5: Subject to : $\sum_j P_{predicted}(j).X_j + \sum_j P_{current}(j).(1 - X_j) \leq P_{budget}$
and $IPC_{current}(j) < IPC_{predicted}(j)$
-

3.4 Data collection

Each benchmark is run for combinations of pre-defined discrete voltage-frequency levels and the number of active cores. The data-collection program scripted in Python gathers core statistics, power estimates, and application statistics (parameters required for calculation of μ). This constitutes of the training data which represents different core behavior and safeguards the predictive model from any bias.

The data gathered is grouped in one matrix per frequency with the following values:

1. Core Power (Static+Dynamic) [in Watts]
2. Core IPC (Instructions per cycle)
3. CPU-intensiveness (μ)

Core IPC and Power (in Watts) are extracted from Snipersim along with McPAT. The value of CPU-intensiveness (μ) is calculated as follows [14]:

$$\mu_i = 1 - \frac{\text{number of } L_1 \text{ stalled}}{\text{number of busy cycles}} \quad (7)$$

It has been observed that discretization gives improved prediction accuracy [24]. Discretized features provide stability and prevent overfitting. The value of μ is discretized in 3 bins (0, 0.5, 1) where $\mu = 0$ depicts memory-intensive while $\mu = 1$ signifies CPU-intensive. For a benchmark with $\mu = 0.5$, it shows it has both memory and CPU-demanding phases. We only need to identify that the new task is compute-intensive/memory-intensive/mixed, 3 levels were chosen. Discretizing the value of μ into more levels will result in additional computation without any significant improvement in precision. However, since a task can have varying memory-bound and CPU-intensive phases, the value of μ has to be calculated dynamically. A workload characterizer to calculate the value of μ has been implemented in Python and is called at every DVFS epoch (taken as 100 μ s). The calculated value is one of the input parameters of the regression learning model.

The data is grouped according to the operating frequency. The data matrix looks like:

(Core ID, IPC, Power, CPU-intensiveness): $(i, IPC_i, Pow(i, VF_i), \mu_i(k))$

where $i \in (1, 32)$, IPC_i and $pow(i, VF_i)$ denotes IPC and Power respectively at core i when operating at VF_i voltage-frequency level. $\mu_i(k)$ represents CPU-intensiveness of the application in the current DVFS epoch k .

For each benchmark, we obtain 62 data points per frequency. The total data set of 1860 ($62 * 6 * 5$) points is partitioned in the ratio of 7:3 to obtain training- set and test-data respectively.

4 Experimental evaluation and results

4.1 Experimental setup

The effectiveness of the proposed technique has been implemented on the Sniper simulator which provides core statistics and power consumption (Dynamic+Static). The

Table 2 Simulated architecture configuration

Number of cores	2, 4, 8, 16, 32
Architecture	Intel Processor Gainestown
Technology node	45 nm
L1-I/D Cache	32 KB, 4-way, LRU
L2 Cache	256 KB, 8-way, LRU
L3 Cache	8 MB, 16-way, LRU
Frequency levels (GHz)	1, 1.4, 1.8, 2.1, 2.7, 3.3
Voltage (V)	0.8, 0.9, 1.1, 1.2, 1.4, 1.6
Explore epoch	500 μ s
DVFS epoch	100 μ s

simulated configuration is mentioned in Table 2. The presented method is designed, but not limited to the 45 nm technology node. It provides the groundwork for future technology nodes (such as 32 nm, 22 nm, etc.) and can be extrapolated. However, future process technologies (under 22nm) will be severely power and thermal constrained. Thus, thermal balancing should be considered in next-generation designs [26]. The voltage-frequency settings have been extracted from the datasheet of Xeon 5550 processor [18] and have been set in `dvfs.py` script in Sniper. Due to the McPAT constraint, the lowest frequency chosen is 1000 MHz. The data-collection program, regression model, and workload analyzer are created in Python script and integrated to Snipersim through API.

In an ideal scenario, DVFS should instantaneously transition from one frequency to another. But in reality, due to internal PLL(phase lock loop), locking times, and capacitances there is a context-switching time. The Intel Xeon processor has a switching latency of 70 μ s at 1600 MHz for a 4-core system [19]. Additionally, we need to consider the DVFS overheads which are approximately 100 μ s for a 1 GHz system [16]. Taking both the factors into account, we have chosen the DVFS epoch which can accommodate both switching latency and DVFS overhead.

4.2 Workload characterizer

Cores react differently to low- voltage operation due to process variation [22]. To incorporate varying task characteristics, we are using parallel, multi-threaded applications from PARSEC benchmark suite [11]. To construct the experimental framework, workload characteristics of applications are gathered on Intel's Sniper simulator [12] and McPAT [13]. Half an hour-long traces of each benchmark were run to generate utilization and power profiles. Simulations conducted for the benchmarks (*blackscholes*, *bodytrack*, *swaptions*, *ferret*, *x264*) running on a 4-core MPSoC are shown in Fig. 4. The power consumption and execution time plots form the baseline of the analysis.

From the graph, it is evident that *ferret* which is highly CPU-intensive continuously burns CPU cycles without accessing memory locations. To achieve better performance, it requires higher processing power. On the other hand, *blackscholes* and *swaptions* which are memory-intensive applications with minimum communication

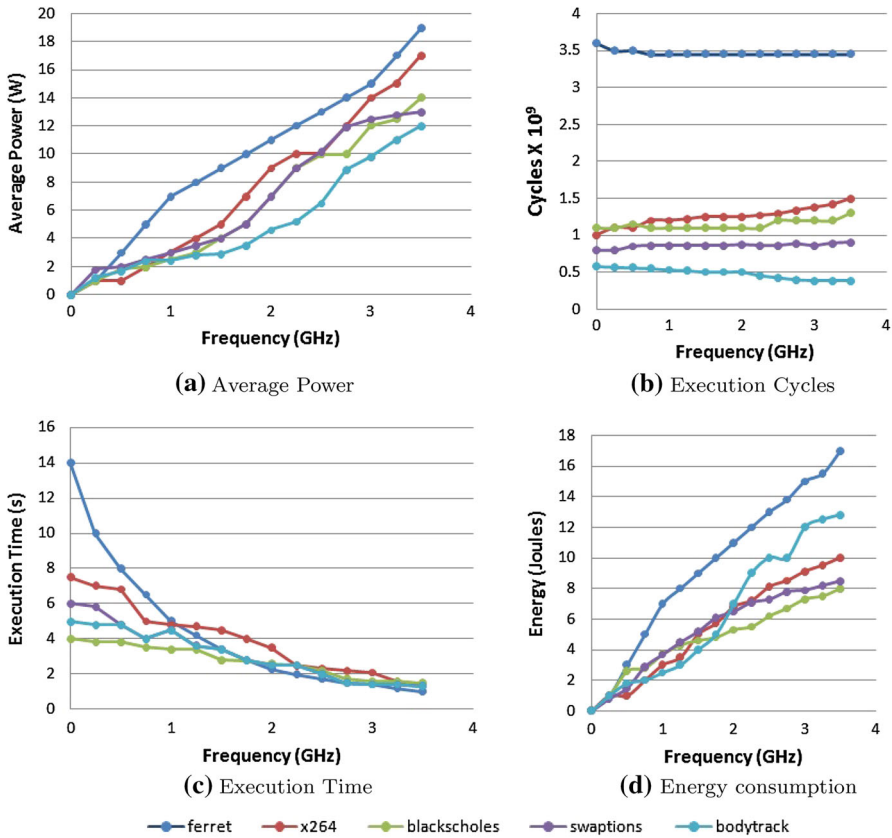


Fig. 4 PARSEC benchmark suite when run for one thread on a 4-core system with *simsmall* input

execute faster and are energy-efficient. Both *x264* and *bodytrack* are medium-coarse but *x264* requires higher power to overcome its lock-barrier. Since *bodytrack* has an inherent load balancing feature, the execution time and power dissipation across cores don't show variation.

4.3 Chip-wide DVFS

A power-allocator program is implemented in Python and acts as a Global-power manager. At first, the power budget corresponding to each core configuration (2/4/8/16/32) is collected. Since the threshold power budget varies with the number of cores and operating frequency, we calculate it dynamically. To obtain this value, we run each benchmark (without DVFS) for each combination of active core configuration and frequency values. The data is collected for all the pre-defined frequency values (Refer Table 2). Illustrating the data collection process, consider operational frequency as 1200 MHz. *swaptions* benchmark is executed for a 2-core, 4-core, 8-core, 16-core, and 32-core architecture separately where each core is operating at 1200 MHz. The

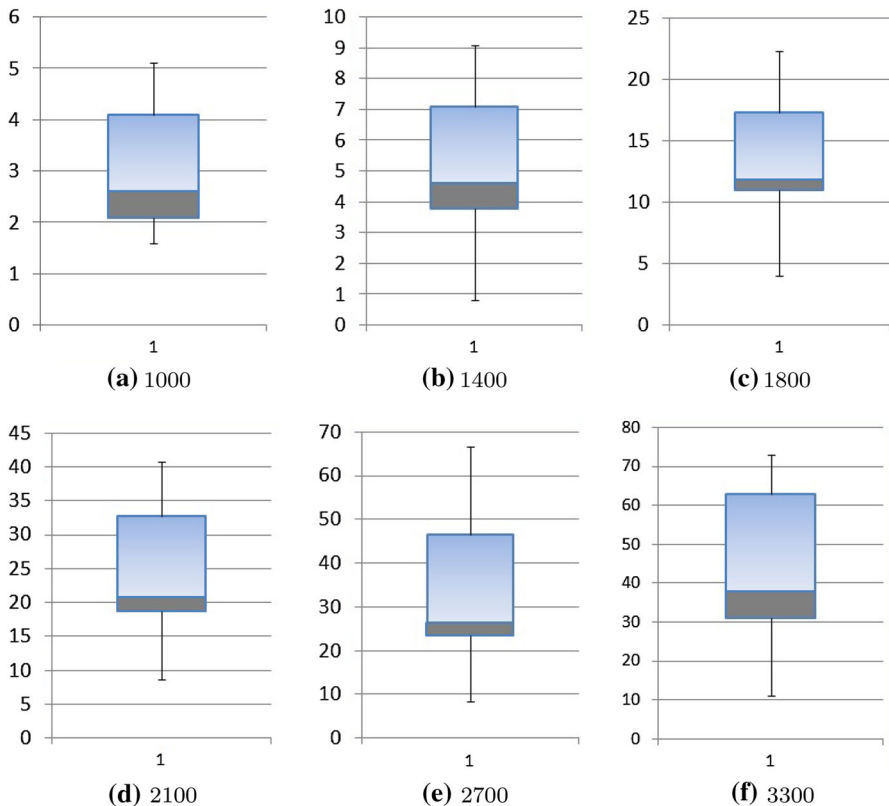


Fig. 5 Boxplots of power consumption (W) with frequency (MHz) at pre-defined DVFS levels

power consumed by each core is collected corresponding to the operational frequency. The same process is repeated for all 6 pre-defined operational frequencies and each benchmark independently. The power consumption is collated per frequency and analyzed through box plots (Fig. 5). A threshold value is set for each of the selected frequency levels (Table 3) and is calculated by taking the median of power consumption for every frequency.

The box plots corresponding to each frequency value have been charted in Fig. 5 for further analysis and calculation. The power consumption value between the **third quartile** and **median** have been used as threshold. This method makes the power-consumption dependent only on frequency thus, making the proposed technique benchmark and core configuration agnostic. However, the main purpose of this value is to provide safety against budget-overshoot as the global manager will not scale frequency if the core has the potential of reaching the power budget. Based on this data, the power threshold for each frequency has been tabulated in Table 3.

The actual power range for Xeon 5550 is not available for more than 4 cores. The collected data, when compared with the actual power range of 95 W for 4 cores [18], is

Table 3 Power threshold (per-core)

Frequency level (MHz)	Power threshold (W)
1000	2.5
1400	4.4
1800	11.2
2100	19.8
2700	24.2
3300	33.6

fairly accurate. Since 3300 MHz is the “Turbo-boost” mode in Xeon it is implemented to study the impact of over-clocking in the system.

The proposed global power allocator is triggered at *explore epoch*. At this instant, the power manager gathers the core statistics to identify any potential budget overshoot. This is to ensure that any increase in the frequency of a core by the fine-grained manager in the last *DVFS epoch* has not led to impractical power values. Each explore epoch consists of system profiling and calculates cumulative chip power consumption. During the profiling phase, any transition in core frequency or memory access is halted and PLLs and DLLs are resynchronized. During the core frequency transition, the core does not execute instructions while the other cores operate normally. This leads to the core transition overhead of the order of tens of microseconds. This penalty is necessary to avoid the oscillation and over-correction of frequency. Since the chip-wide manager is called at every 500 μ s while the per-core manager at every 100 μ s the effect of transition overhead is diminished.

4.4 Per-core DVFS

The DVFS power manager gathers application and architecture statistics through the data collection program. The fine-grained power manager uses linear regression to predict the next interval IPC and Power values corresponding to each operating frequency. Distinct regression models have been developed to estimate continuous values of $IPC_{predicted}$ and $Power_{predicted}$ for the next interval. The obtained continuous values have been discretized to reduce complexity as mentioned in Sect. 3.4. At every *DVFS epoch(k)*, the remaining budget per-core ($delta_{budget}$) is calculated. The cores with the highest $delta_{budget}$ are arranged in max heap-sort. Inclusion of this step has two-fold benefits: (1) Arranging tuples in maximum heap-sort facilitates quick identification of core(s) which have the maximum potential for frequency scaling and; (2) in case there is no change in frequency, the system doesn't need to recalculate the statistics for all cores and can simply pick the next child-node from the graph.

The Power and performance models are generated as a function of PMU (frequency, $IPC_{lastinterval}$, $Power_{lastinterval}$, $\mu_{lastinterval}$). The value of $Power_{budget}$ is passed by the global manager. The power manager ensures that the total power allocated doesn't exceed the power budget. The residual power budget signifies the available power for the DVFS assignment. The core with the highest $delta_{budget}$ is identified from heap-sort. The manager verifies variation in the value of μ and triggers the change

Table 4 Root mean square error

Frequency level (MHz)	Root mean square error	
	IPC	Power
1000	0.32	0.458
1400	0.35	0.576
1800	0.41	0.651
2100	0.44	0.87
2700	0.50	1.36
3300	0.82	2.38

in voltage-frequency requests. Based on predicted values of IPC and Power for the next interval, new voltage-frequency settings are assigned.

An indicator of the accuracy of the predictive model is Root Mean Square Error (RMSE) which is defined as in equation 8. RMSE represents the deviation between the observed and estimated/predicted values. It is desirable to have lower RMSE values which are indicative of an accurate model.

$$J(\theta) = \frac{1}{m} \sqrt{\sum_{j=1}^m (h_{\theta}(x^{(j)}) - y^{(j)})^2} \quad (8)$$

where m , training samples statistic; $x^{(j)}$, j th value of predictor variables; $y^{(j)}$, j th measured value; $h_{\theta}(x^{(j)})$, j th predicted value.

The average value of RMSE corresponding to different operating frequencies is tabulated in Table 4.

We observe that the prediction models work fairly for both IPC and Power estimation. For Dual-core and quad-core systems, the RMSE is very small as the applications behave expectedly. The power allocation algorithm can adapt to changes in workload behavior. At the start of the simulation, high RMSE values were encountered especially since the model was in the learning phase. However, the model shows a large deviation for higher frequency. This is attributed to the fact that power and frequency are not linearly related. Thus, a non-linear or probability-based model may give better results in the future. Since the suggested model is fairly able to predict IPC for the next interval, it prevents performance degradation.

4.5 Results and analysis

The proposed method has been compared with two state-of-the-art algorithms: MaxBIPS and TPEq. Below, we briefly describe our implementation of these methods: The algorithm **MaxBIPS** [16] uses three power modes: *Turbo*, *Efficient 1(Eff1)* and *Efficient 2(Eff2)*. In Turbo mode, the system works on the highest frequency and neglects power savings. The Eff1 and Eff2 mode aims to achieve energy-performance tradeoff in different proportions in terms of power-savings and performance degradation. The method targets a 3:1 power/performance tradeoff. Unlike MaxBIPS which

uses 3 modes, this work considers 6 frequency levels providing greater fine-tuning. The 3300 MHz setting can be considered equivalent to ‘Turbo’ mode of MaxBIPS. Although MaxBIPS has been implemented only for up to 8 cores, we have scaled it to 32 cores for the comparison. Since *bodytrack* needs at least a 4-core system (as it has 2 extra threads), its 2-core comparison has not been included.

TPEq algorithm [17] focusses on multi-thread workloads with barrier and/or lock synchronization. It defines a metric ‘*progress*’ which is inversely related to the weighted CPI of each thread and helps to establish the most lagging thread. The core configuration for this thread is changed to improve the progress metric while abiding the power and performance constraints. Both MaxBIPS and the proposed method use DVFS epoch as 500 μ s while TPEq uses 1ms as epoch length which reduces the granularity of temporal adaptation of the thread. The default power settings for MaxBIPS are 90% of power budget while TPEq starts with the lowest frequency level. However, our method initiates the system with the highest frequency under capping constraints.

Considering that the main aim of this method is to achieve joint optimization of power and makespan, we evaluate the proposed technique for both reductions in power consumption and performance degradation. The metric used to calculate power reduction/energy-savings is given by equation 9 and have been plotted in Fig. 6:

$$\frac{\text{mean}(Power_{noDVFS}) - \text{mean}(Power_{method})}{\text{mean}(Power_{noDVFS})} \quad (9)$$

Here $Power_{noDVFS}$ signifies the power consumption of each benchmark in normal execution, without any DVFS. $Power_{method}$ represents power values for either the proposed solution or state-of-the-art algorithm (MaxBIPS or TPEq). Another metric used to estimate throughput is *Performance degradation*. It is quantified with elapsed execution time for individual benchmarks [16] and the normalized performance degradation (with respect to MaxBIPS) is plotted in Fig. 7.

Power Reduction: At the outset it is visible from Fig. 6, both MaxBIPS/ TPEq and the prospective method can achieve similar energy-savings for a 2-core system. However, for computationally-intensive tasks like *ferret* and *x264*, the presented solution shows better results. *ferret* shows the highest power reduction across all the cores because it has both CPU and memory-demanding phases. The proposed method can adapt better than MaxBIPS which only relies on the precision of V^2f rule. Inefficient power actions in TPEq lead to lower energy savings. Since *swaptions* is a no-barrier workload, the power manager is easily able to predict its characteristics and hence make suitable decisions. As a memory-intensive, medium-sized workload the suggested algorithm can utilize the newly added cores and thus improve energy-efficiency. Since it is a low-lock application with very few unexpected fluctuations, all the methods show comparable results.

However, *blackscholes* which is a small scale, the balanced workload does not benefit much from power balancing techniques especially for a lower number of cores. However, as the cores scale *blackscholes* improves in energy-efficiency. This is because it relies on neighboring cores that consume lower power by borrowing from their power budgets. *bodytrack* being compute-intensive shows significant power reduction

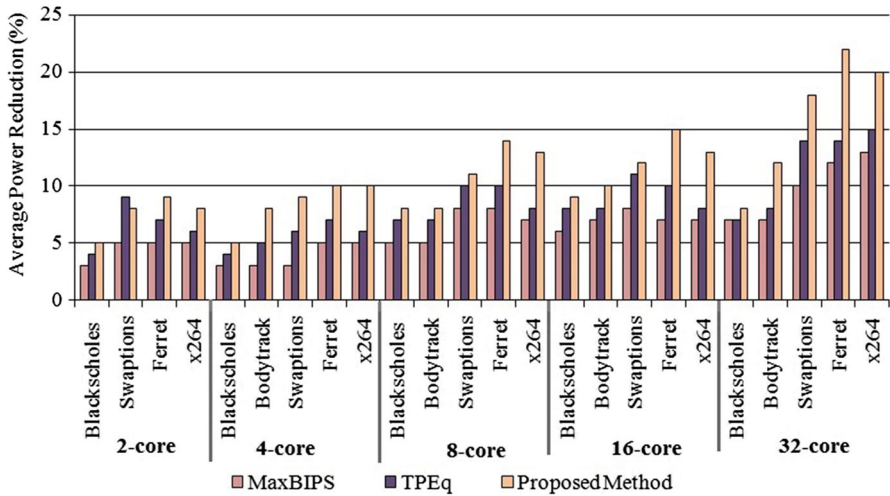


Fig. 6 Average power-reduction (%) achieved across benchmarks

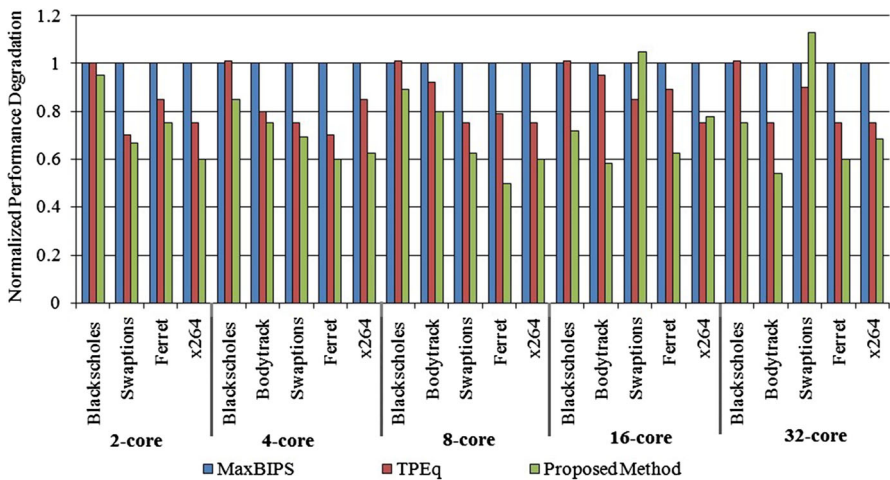


Fig. 7 Normalized performance degradation across benchmarks (with respect to MaxBIPS)

especially for cores greater than 8. This is attributed to low communication and medium granularity of workload which translates to higher energy-efficiency.

Performance Degradation: Since this work aims adaptation to workload characteristics, performance degradation becomes an important criterion. This metric demonstrates the prediction capability of the per-core manager. Lower value shows the precision of the decision strategy. *blackscholes* being a balanced workload requires an almost equal amount of power across cores. It has inherent static load balancing due to which it can maintain the throughput. With the augmentation of the cores, it is unable to fully utilize the additional processing capacity hence fails in performance improvement. Since it is a no-lock and low-barrier workload, simply increasing the

number of cores does not translate into higher performance. *bodytrack* performs better than *blackscholes* in terms of performance degradation. The high-lock, high-barrier aspect of *bodytrack* prevents the manager from reaching low-performance values. As the number of cores increase, the built-in dynamic load balancing ensures that all the additional processing capacity is converted to higher performance. Thus, for 16 and 32 core configuration the performance degradation remains almost constant. Our predictor can accomplish comparable results with TPEq and outperform MaxBIPS even for higher core configurations. Both *ferret* and *x264* don't have load balancing capability and are computationally expensive. *ferret* exhibits lower performance degradation in proposed method as opposed to MaxBIPS. This is because our technique dynamically calculates workload phases while MaxBIPS achieves it through scaling of memory and cache. *x264* is similar to *ferret* but has a smaller working set which facilitates higher performance without any significant loss. *swaptions* works as per expectations up to 8-core configuration. However, for a higher number of cores, it shows an increase in performance degradation as the static load balancing is unable to match the augmentation. TPEq outperforms the presented method by around 15% for a 16-core system and 18 % for a 32-core configuration.

Budget overshoot accuracy It is essential to evaluate that the power consumed by each method is constrained by the global power budget. It is observed that a good power budgeting and balancing method can lower processor temperature gradient without degrading the performance of the system [26]. Table 5 displays the accuracy of each algorithm in selecting v-f setting within a fixed power constraint. We studied the accuracy of matching the power budget (p_b) for each benchmark for two fixed values -130W and 140 W. The tabulated accuracies reflect the fraction of intervals for which the power consumption at optimal setting selected, matched/stayed below than the allotted power budget.

Our technique outperforms both MaxBIPS and TPEq in maintaining power capping while selecting v-f levels. As MaxBIPS aims to achieve high performance it continuously pushes towards power budget while considering the dynamic characteristics of the application. Thus, the accuracy suffers for benchmarks like *blackscholes* and *swaptions*. While TPEq, favors balanced benchmarks such as *bodytrack*. Since *x264* is compute-intensive and requires high power, both MaxBIPS and TPEq can perform identical to our method. Our method performs better for lower budgets thus keeping the power consumption further minimized.

We summarize the observations as follows:

(1) For a dual-core system, the proposed method works identical to the state-of-the-art algorithms. (2) As the cores are augmented from 4 to 8, performance degradation decreases as global budgeting becomes effective. (3) The benefit of power and performance prediction becomes dominant with the scaling of cores. From Figs. 6 and 7, it is clear that the proposed approach can identify and handle changes in application characteristics better than *MaxBIPS* and *TPEq*. The power reduction result shows that the implemented per-core manager can identify variations in application. For compute-intensive tasks (*ferret* and *x264*) we are able to reduce power by 30% and improve performance degradation by 35%. (4) The presented approach can adapt well with a boost in the number of cores and increase overall power reduction by 33% from MaxBIPS and 25% from TPEq while minimizing performance degradation across

Table 5 Accuracy of DVFS techniques in selecting the optimal operating points under Power budget constraints

Benchmark name	MaxBIPS		TPEq		Proposed method	
	$p_b < 130\text{ W} (\%)$	$p_b < 140\text{ W} (\%)$	$p_b < 130\text{ W} (\%)$	$p_b < 140\text{ W} (\%)$	$p_b < 130\text{ W} (\%)$	$p_b < 140\text{ W} (\%)$
<i>blackscholes</i>	23.70	44.60	27.80	66.20	100	88.80
<i>bodytrack</i>	85.00	99.90	89.80	94.70	96	80.20
<i>ferret</i>	51.00	97.00	36.80	66.30	98.60	79.20
<i>swaptions</i>	21.20	32.20	25.40	45.20	87.20	86.10
<i>x264</i>	97.40	81.80	100	99.90	100	94
Average	55.66	71.10	55.96	74.46	96.36	85.66

applications and configurations. Additionally, the proposed method can keep operating power under power budget and prevent overshoot more efficiently than both MaxBIPS and TPEq.

Further analyzing the *complexity* and *overhead* associated, we examine the considered method in contrast with both *MaxBIPS* and *TPEq*. The algorithm complexity in MaxBIPS is $O(N.\alpha^N)$ where α signifies the number of VF levels as it employs exhaustive search to find optimal v-f values. The presented method and TPEq employ heap-sort with respect to IPC value, hence has complexity $O(N.\log(N))$ in the worst case. In the proposed method, the linear regression-based model augments the complexity by $O(\frac{mn^2}{p} + \frac{n^3}{p} + n^2\log(p))$ where p is number of cores [20]. Thus, a fast, scalable learning-based model has been identified as future work which has a limited number of features (n) and learning samples (m).

5 Conclusion

This paper presents a novel perspective on the dynamic power management technique in multi-core systems. The presented approach demonstrates an efficient performance-energy tradeoff technique that incorporates application behavior in decision strategy while capping power at both the chip and per-core level. The presented DVFS control algorithm allocates the power budget at a coarse granularity. At the fine-grain level, suitable voltage-frequency settings for each core are determined using workload characteristics while maximizing the performance. This work also explores the usage of using two-different control times (*explore epoch* and *DVFS epoch*) for *chip-wide* and *per-core* manager respectively. This inclusion reduces the overhead for global budget allocator by 5X.

The prospective approach evaluated for various configurations (2–32 cores), achieves 33% and 25% power reduction compared to MaxBIPS and TPEq respectively. It can consistently minimize performance degradation giving overall 23% improvement. The advantage of workload analyzer is visible for CPU-intensive applications where the energy savings increase by 30% and performance degradation diminishes by 35%. The method shows the potential of combining learning-based DVFS with workload characteristics of an application. Though there are unexplored aspects of a task behavior like deadline, priority, etc., this method gives a baseline to analyze aggressive scaling strategies. Additionally, the applicability of our method for different processor designs such as 22nm die size or single-ISA heterogeneous multi-core architecture [25] may be explored in the future.

References

1. Ghiasi S, Casmira J, Grunwald D (2000) Using IPC variation in workloads with externally specified rates to reduce power consumption. In: Complexity-effective design at ISCA27
2. Dhiman G, Rosing TS (2007) Dynamic voltage frequency scaling for multi-tasking systems using online learning. In: Proceedings of international symposium on low power electronics and design, pp 207–212

3. Dhiman G, Pusukuri KK, Rosing T (2008) Analysis of dynamic voltage scaling for system level energy management. *USENIX HotPower*, 8
4. Cebrin JM, Snchez D, Aragn JL, Kaxiras S (2013) Efficient inter-core power and thermal balancing for multicore processors. *Computing* 95(7):537–566
5. Reddy BK, Singh AK, Biswas D, Merrett GV, Al-Hashimi BM (2017) Inter-cluster thread-to-core mapping and DVFS on heterogeneous multi-cores. *IEEE Trans Multi-Scale Comput Syst* 4(3):369–82
6. Kim W, Gupta MS, Wei GY, Brooks D (2008) System level analysis of fast, per-core DVFS using on-chip switching regulators. In: 2008 IEEE 14th international symposium on high performance computer architecture. IEEE, pp 123–134
7. Yang S, Shafik RA, Merrett GV, Stott E, Levine JM, Davis J, Al-Hashimi BM (2015) Adaptive energy minimization of embedded heterogeneous systems using regression-based learning. In: 2015 25th international workshop on power and timing modeling, optimization and simulation (PATMOS), pp 103–110
8. Qiu M, Sha EHM (2009) Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems. *ACM Trans Des Autom Electron Syst* 14(2):25:1–25:30
9. Cochran R, Hankendi C, Coskun AK, Reda S (2011) Pack & cap: adaptive DVFS and thread packing under power caps. In: Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture, MICRO-44, (New York, NY, USA), ACM, pp 175–185
10. Huang H, Lin M, Yang LT, Zhang Q (2020) Autonomous power management with double-Q reinforcement learning method. *IEEE Trans Ind Inf* 16(3):1938–1946. <https://doi.org/10.1109/TII.2019.2953932>
11. Choi K, Soma R, Pedram M (2004) Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times. *IEEE Trans Comput Aided Des Integr Circuits Syst* 24(1):18–28
12. Choi J, Park G, Nam D (2020) Interference-aware co-scheduling method based on classification of application characteristics from hardware performance counter using data mining. *Clust Comput* 23(1):57–69
13. Weissel A, Belloso F (2002) Process cruise control: event-driven clock scaling for dynamic power management. In: Proceedings of the 2002 international conference on compilers, architecture, and synthesis for embedded systems
14. Isci C, Buyuktosunoglu A, Cher CY, Bose P, Martonosi M (2006) 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, 2006 Dec 9. IEEE Computer Society, pp 347–358
15. Turakhia Y, Liu G, Garg S, Marculescu D (2016) Thread progress equalization: dynamically adaptive power-constrained performance optimization of multi-threaded applications. *IEEE Trans Comput* 66(4):731–744
16. Ma Y, Chantem T, Dick RP, Hu XS (2017) Improving system-level lifetime reliability of multicore soft real-time systems. *IEEE Trans Very Large Scale Integr VLSI Syst* 25(6):1895–1905
17. Bienia C, Kumar S, Singh JP, Li K (2008) The PARSEC benchmark suite: characterization and architectural implications. In: Proceedings of the 17th international conference on parallel architectures and compilation techniques. ACM, pp 72–81
18. Cohen J, Cohen P, West S, Aiken L (2013) Applied multiple regression/correlation analysis for the behavioral sciences. Taylor & Francis, Milton Park
19. Gupta M, Bhargava L, Indu S (2019) Dynamic voltage frequency scaling in many-core systems using adaptive regression model. Presented in international conference on signal processing, VLSI and communication engineering (ICSPVCE-2019)
20. Wang Z, Tian Z, Xu J, Maeda RK, Li H, Yang P, Wang Z, Duong LH, Wang Z, Chen X (2017) Modular reinforcement learning for self-adaptive energy efficiency optimization in multicore system. In: 2017 22nd Asia and South pacific design automation conference (ASP-DAC). IEEE, pp 684–689
21. Ren S, He L, Li J, Chen Z, Jiang P, Li CT (2019) Contention-aware prediction for performance impact of task co-running in multicore computers. *Wirel Netw* 13:1–8
22. <http://ark.intel.com/products/37106>
23. Mazouz A, Laurent A, Pradelle B, Jalby W (2014) Evaluation of CPU frequency transition latency. *Comput Sci Res Dev* 29:187–195

24. Bacha A, Teodorescu R (2013) 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, pp 297–307
25. Carlson TE, Heirmant W, Eeckhout L (2011) Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In: SC'11: proceedings of 2011 international conference for high performance computing, networking, storage and analysis. IEEE, pp 1–12
26. Li S, Ahn JH, Strong RD, Brockman JB, Tullsen DM, Jouppi NP (2009) McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In: Proceedings of the 42nd annual IEEE/ACM international symposium on microarchitecture. ACM, pp 469–480
27. Chu CT, Kim SK, Lin YA, Yu Y, Bradski GR, Ng AY, Olukotun K (2006) Map-reduce for machine learning on multicore. In: Proceedings of the 20th annual conference on neural information processing systems (NIPS '06), pp 281–288
28. Sjlinder M, Martonosi M, Kaxiras S (2014) Power-efficient computer architectures: Recent advances. *Synth Lect Comput Archit* 9(3):1–96

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.