Workload-Aware Runtime Energy Management for HPC Systems

Karunakar R. Basireddy, Eduardo W. Wachter, Bashir M. Al-Hashimi and Geoff V. Merrett
University of Southampton
Southampton, United Kingdom
{krb1g15, eww1n17, bmah, gvm}@ecs.soton.ac.uk

***Abstract*—Energy efficiency has become a crucial factor in high-performance computing, mainly due to its effect on operating costs and failure rates of computing platforms. To improve the energy efficiency of such systems, processors are equipped with low-power techniques such as dynamic voltage and frequency scaling (DVFS) and power capping. These techniques have to be controlled carefully as per the workload; otherwise, it may result in significant performance loss and/or power consumption due to system overheads (e.g. DVFS transition latency). Existing approaches are not effective in adapting to workload variations as they do not consider the combined effect of application compute-/memory-intensity, thread synchronization contention, and non-uniform memory accesses (NUMAs) owing to the underlying processor architecture. In this work, we propose a workload-aware runtime energy management technique that takes the aforementioned factors into account for efficient *V-f* control. The proposed technique measures the processor workload using Memory Accesses Per Micro-operation (MAPM), and also considers the thread synchronization contention and latency due to NUMAs to select an appropriate *V-f* setting. This approach also uses workload prediction for pro-active *V-f* control to improve the energy consumption and performance loss. The proposed technique has been implemented on the 12-core (24 threads) Intel Xeon E5-2630 and 61-core (244 threads) Xeon Phi many-core platforms, supporting per-core and system-wide DVFS, respectively. When evaluated with different application scenarios, results show an improvement in energy efficiency of up to 81.2% compared to existing approaches.**

***Index Terms*—Run-time Power/Energy Management, Dynamic Voltage and Frequency Scaling, High-Performance Computing, Non-Uniform Memory Access**

I. Introduction

The Information Technology (IT) industry has evolved over past decades, and changed lifestyles. IT has enabled the revolution of many veteran industries and businesses, and proportioned the disruption of some of the biggest markets. This growth has been made possible by cheaper and more powerful High-Performance Computing (HPC) resources. Such systems can be accessed from anywhere in the world through the cloud computing infrastructure [1]. One of the main design challenges to these HPC systems is energy efficiency [2].

There have been various approaches proposed to improve the energy efficiency of HPC systems [2]–[14] using low-power techniques such as dynamic voltage and frequency scaling (DVFS), power capping or clock gating. These approaches introduced various metrics, such as Instructions Per Cycle (IPC) and Millions Instructions Per Second (MIPS), for determining the application workload on the processor through online and/or offline characterization. These metrics are used to take decisions on the selection of Voltage-frequency (*V-f*) setting. The low-power techniques used for deciding the *V-f* setting have to be controlled carefully, as per the workload, to avoid performance and/or power overheads [4]. Therefore, accurate estimation of workload is a key to achieving energy efficiency.

The applications targeted for HPC systems are usually implemented as multi-threaded to efficiently exploit the available hardware-level parallelism [15]. For multi-threaded applications, the workload not only depends on memory-/compute-intensity but also on thread synchronization contention. In addition to that, modern HPC systems are usually based on Non-Uniform Memory Access (NUMA) architecture, where memory access time depends on the memory location relative to the processor [16]. A thorough analysis of related works [3]–[5], [7]–[9], [11], [12], [17], [18] shows that the existing approaches do not consider the combined effect of the above factors. As a result, they are not efficient for adapting to workload variations, which is essential for improving energy efficiency. Furthermore, reported approaches do not take memory-contention into account when executing applications concurrently, resulting in increased power consumption without any performance benefits.

This work proposes a workload-aware runtime energy management technique that takes the aforementioned factors into account for improving the energy efficiency. Our proposal measures the processor workload using Memory Accesses Per Micro-operation (MAPM) and utilization for estimating the thread synchronization contention. Moreover, latency due to NUMAs is calculated by monitoring the remote and local memory accesses during the application execution. As part of this, we use four hardware performance monitoring counters (PMCs) to compute MAPM, utilization and NUMA latency. To determine the appropriate *V-f* setting, a binning based approach is employed which takes utilization and MAPM as inputs. Furthermore, our approach works on both per-core and system-wide DVFS supporting platforms. Experimental validation is performed on the 12-core (24 threads) Intel Xeon E5-2630 and 61-core (244 threads) Xeon Phi many-core platforms. The former supports per-core DVFS, whereas the latter is based on system-wide DVFS. The main contributions of this paper are:

1) An accurate estimation of processor workload by consid-

ering the combined effect of memory-/compute-intensity, thread synchronization contention and NUMA latency;
2) A binning based approach for efficiently determining the *V-f* setting as per the predicted workload;
3) Validation of the proposed approach on two hardware platforms, the Xeon E5-2630 and Xeon Phi 7620P.

The rest of the paper is organized as follows. A brief review of existing works is presented in Section II. The problem formulation is discussed in Section III. A detailed discussion of the proposed approach and its experimental evaluation are given Section IV and Section V, respectively. Finally, Section VI concludes the paper.

II. Related Work

There have been several approaches proposed for obtaining energy savings in HPC systems. Techniques proposed in [7]–[9] determines bottlenecks during application execution using performance counters. They present a framework for direct, automatic profiling of power consumption for non-interactive and parallel scientific applications to assist the scheduler. Rountree et al. [10] do a critical path analysis to determine which tasks may be slowed down to achieve energy savings while minimizing the performance loss in the parallel execution. This analysis appears beneficial only when applications have computation or communication imbalances among participating processes, which is typically not the case for highly efficient parallel applications [5].

The schemes presented in [11], [12] determine the communication phases to apply DVFS. A technique that applies both DVFS and over-clocking to CPUs to save energy and improve execution time is discussed in [17]. Marathe et al. [18] proposed a runtime system conductor that dynamically distributes available power to different compute nodes and cores based on the available slack to improve performance. The conductor performs both upscaling and downscaling of processor frequency to decrease execution time and to save energy in an indirect manner through power clamping, which differs from the traditional approach of only downscaling to save energy. The authors of [3] proposed a latency-aware DVFS algorithm to avoid aggressive power state transitions. They argue that too frequent DVFS changes are not only unprofitable but also detrimental to performance, due to the extra time and energy costs introduced. This approach divides each application into phases through profiling and uses this information to decide whether changing the *V-f* setting is beneficial or not during the application execution.

Sundriyal et al. [4] present an energy management approach, relying on the Intel Running Average Power Limit (RAPL). It provides a standard interface for measuring and limiting the processor and memory power. This approach uses Memory Access Per Micro-operation (MAPM) and MIPS (millions of instructions per second) metrics to indicate how a change of frequency will affect the performance. A procedure to select power capping thresholds dynamically on the Xeon Phi platform is given in [5]. They noticed that default power capping limits employed on this platform are much higher than the majority of applications would reach. Considering this, different power limits are defined according to the workload characteristics and application performance.

The aforementioned works show how energy savings can be achieved by adapting to the workload at runtime. However, they do not take into account the combined effect of application compute-/memory-intensity, thread synchronization contention and NUMA latency. Furthermore, the memory contention due to concurrent execution of applications is also not addressed. This shows an opportunity for improving the energy consumption, which is exploited in this paper.

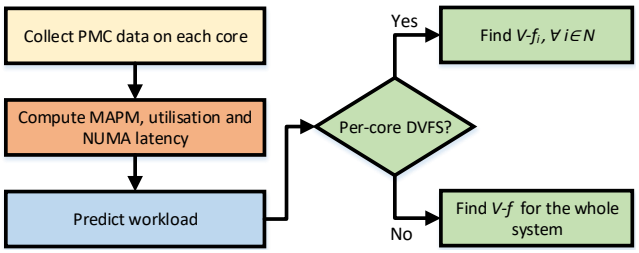

Fig. 1. Illustration of various steps in the proposed approach as a flow-chart.

III. Problem Formulation

The proposed approach considers two kinds of many-core platforms; one supporting per-core DVFS and the other one with system-wide DVFS. The following provides our problem definition, assuming that there are *N* cores in a platform.

Given a set of multi-threaded applications and a many-core platform supporting per-core/system-wide DVFS

Find an efficient *V-f* setting periodically for each core (*V-f*$_i$, where i = 1, 2,..., *N*) or for the whole system such that the overall system energy consumption can be minimized while maximizing performance.

IV. Proposed Approach

The proposed approach is illustrated in Fig. 1, which has the following four steps:

(a) PMC data collection;
(b) Computing MAPM, utilization and NUMA latency;
(c) Workload prediction;
(d) Identification of *V-f* setting.

The PMC data are collected periodically to get the information about the architectural events. The collected data is used to compute the MAPM, utilization and NUMA latency, which are further fed into a prediction algorithm for estimating the future workload. Considering the underlying architecture (per-core or system-wide DVFS), the *V-f* setting is determined according to the predicted workload. The detailed discussion on each step is given in the following sections. As the device firmware automatically adjusts the voltage for a selected frequency in our chosen hardware platforms, we refer to *V-f* and frequency interchangeably throughout the paper.

A. PMC data collection

The modern processors support runtime monitoring of architectural events (e.g. instructions retired, cache misses, etc.) through a set of hardware performance monitoring counters (PMCs). These counters can be configured to count a particular architectural event from a list of supported events by a particular processor. The proposed approach needs `instructions retired`, `Last-Level Cache (LLC) misses` and `active CPU cycles`. It has been already shown in [19], [20] that similar events can be used to efficiently estimate the processor workload at runtime. Further, if the chosen platform has NUMA architecture, `remote memory accesses` are also collected. On our chosen platform (Xeon E5-2630 and Phi), the events matching to the above are the following symbolic names: `UOPS_RETIRED`, `LAST_LEVEL_CACHE_MISSES`, `UNHALTED_CORE_CYCLES`, and `MEM_LOAD_UOPS_LLC_MISS_RETIRED.REMOTE_DRAM`. We use the tool `perfmon` [21] for periodically sampling the PMCs on each core simultaneously. This tool provides routines to configure the PMCs using symbolic names to count a particular event, to initialize, and to terminate the data collection.

B. Computing MAPM, utilization and NUMA latency

To find the appropriate frequency, the processor workload has to be determined accurately. This involves choosing a right metric and taking underlying memory architecture into account. Therefore, we use the metric Memory Accesses Per Micro-operation (MAPM), similar to [4], [20], [22], along with utilization and latency associated with non-uniform memory accesses. MAPM is computed as a ratio between memory accesses and micro-operations retired during the DVFS interval. This one has been used for measuring the memory-intensity of workload during application execution [4], [20]. Usually, a high value of MAPM suggests that the workload is memory-bound and vice versa. The memory accesses and micro-operations retired are measured using the PMCs, `LAST_LEVEL_CACHE_MISSES` and `UOPS_RETIRED`, respectively.

We have observed, especially in case of multi-threaded applications, the actual value of MAPM does not always represent the memory-intensity of the workload accurately. A lower value of MAPM may not always mean a compute-bound workload due to the following reasons. An application might have a lot of synchronization contentions, such as inter-thread locks and barriers or communication contentions, happening on external peripheral devices, e.g. storage devices, keyboard, etc. In such cases, MAPM alone will fail to determine the actual load on the processor. To address this issue, along with MAPM, utilization is also considered for estimating the effect of aforementioned factors. To compute the processor utilisation, we utilise un-halted CPU cycles, monitored using the PMC `UNHALTED_CORE_CYCLES`.

Moreover, if a many-core platform is based on NUMA architecture, then the memory latencies differ depending on the relative distance between processor and memory. Access to the local memory will be much faster than accessing the remote memory. Therefore, to efficiently calculate the effect of MAPM on processor performance, the NUMA latency should also be taken into account to select an appropriate *V-f* setting. To accomplish this, we measure the number of remote memory accesses using the PMC `MEM_LOAD_UOPS_LLC_MISS_RETIRED.REMOTE_DRAM` and use it while computing the MAPM. Assume that μ-*ops*, m_l and m_r represent micro-operations retired, accesses to local and remote memory, respectively. Then, MAPM can be determined as follows,

$$MAPM = \frac{m_l + \theta * m_r}{\mu\text{-}ops} \tag{1}$$

The constant θ depends on the latency associated with the remote memory access, which can be determined from the datasheet of the processor. As discussed in [20], the contention on memory, when applications are memory-bound, is also considered (refer to equation (4)).

C. Workload prediction

To adapt to workload variations and to achieve energy minimization, proactive control of *V-f* is of utmost importance [20]. Therefore, the future workload (t_{i+1}) needs to be predicted at t_i to determine the appropriate *V-f* setting for the time interval $t_i \rightarrow t_{i+1}$. To accomplish this, we use an exponential weighted moving average (EWMA) filter [23] to predict the workload p_{i+1} during the interval $t_i \rightarrow t_{i+1}$,

$$p_{i+1} = \gamma \times a_i + (1 - \gamma) \times p_i + p_e \tag{2}$$

where γ, p_i and a_i are the smoothing factor, predicted and actual workloads for the epoch $t_{i-1} \rightarrow t_i$, respectively. To minimize workload miss-predictions, the predicted workload of the interval $t_{i-1} \rightarrow t_i$ is compared to the actual workload measured from hardware PMCs. Subsequently, computed prediction error p_e (the difference between actual and predicted workloads) is used to improve the workload prediction for $t_i \rightarrow t_{i+1}$. The accuracy of prediction highly depends on γ and a fixed value of γ would result in frequent miss-predictions if there are large workload variations. Therefore, similar to [19], the value of γ is changed as follows,

$$\gamma = \beta + \alpha \times p_e / p_i \tag{3}$$

The values of coefficients α and β are given in Section V-A. We use the Equation (2) for predicting the utilization and MAPM on each core, which will be used to identify the *V-f* setting proactively as explained in the following section.

D. Identification of V-f setting

Determining the appropriate *V-f* setting is key to energy efficiency and to minimize performance loss. We employ a binning based approach [24] for finding the *V-f* setting based on the application workload. This approach consists of the two bins, one utilization bin and other one is MAPM bin. The utilization and MAPM computed from the step explained in Section IV-B acts as inputs to this stage, as shown in Fig. 2.

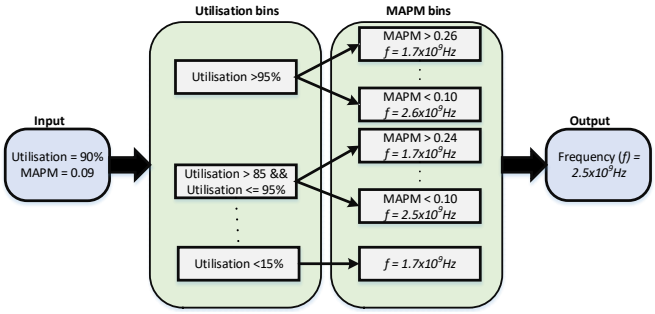

Fig. 2. An example of *V-f* setting selection using binning based approach.

Bins training: We use offline training to determine the bin boundaries. The first part of the training process is obtaining training samples. One training sample consists of the collection of metrics, MAPM and utilisation, along with the application performance at different *V-f* settings available on the chosen platform. In order to generate a diverse range of training samples, we used the applications from different benchmark suites, including SPEC CPU2006 [25], LMBench [26], RoyLongbottom [27], PARSEC 3.0 [28], NAS [29] and Rodinia [30]. These training samples are grouped into bins based on the *utilisation* and *MAPM*, and assigned a *V-f* setting to each bin that is energy efficient with no or little ($<$1%) performance loss. During runtime, the utilization and MAPM are computed to find an appropriate *V-f* setting using the pre-determined bins. We use the utility `cpufreq-set` for changing the *V-f* setting during the application execution.

In case of per-core DVFS supporting platforms, the *V-f* setting of each core can be set as per its workload. But, if the platform supports only system-wide DVFS, the *V-f* setting of the whole system should be chosen in such a way that no application thread experiences the performance loss. Such cases can arise when applications with different workload types (e.g. compute-bound, memory-bound, etc.) are running concurrently. A memory-bound workload can be run at a lower *V-f* setting than a compute-bound, but if both such workloads are executing concurrently, selection of an appropriate *V-f* setting is challenging. We use the similar approach proposed in [20] to address this issue by giving the preference to compute-bound workload over memory-bound one, and considering the impact of memory-contention due to concurrent execution. Therefore, the *V-f* setting is determined by the following MAPM value ($MAPM_t$),

$$MAPM_t = \min\{M_1, M_2, M_3, ..., M_N\} + \zeta * \sum_{i=1}^{N} \frac{M_i}{N} \quad (4)$$

Here, M_i, N and ζ represent MAPM of core i, number of processing cores in the system and coefficient to determine the effect of memory contention on the performance of each application thread, respectively. Furthermore, it can be understood that the result of minimum value computation in Equation (4) is same as the MAPM of the core with maximum utilisation because the most compute-intensive workload has minimum MAPM.

Identification of unused cores: MAPM of idle cores, executing no application thread, is usually low due to fewer memory accesses [19]. This gives a misimpression that such cores are executing a compute-intensive application, leading to the selection of a high *V-f* value for the whole system and thus increasing energy consumption. This becomes prominent when there are more cores than the number of concurrent applications. To address this, the proposed algorithm determines idle cores at runtime using utilization threshold. It is important to note that, for per-core DVFS supporting platforms, identification of idle cores is not required as the *V-f* setting is determined for each core, which is already taken care by the utilization bins, shown in Fig. 2. Based on experimental observation, if the utilization of a core is below 5%, it is identified as an idle core. Subsequently, MAPM of such cores is set to 10 (any value larger than one would be fine as the value of MAPM usually does not exceed one). This nullifies the influence of idle cores on *V-f* setting as it is mostly decided by the minimum MAMP (Equation (4)).

V. Experimental Setup and Results

The proposed approach is validated on an Intel Xeon E5-2630 running Red Hat Linux, and Xeon Phi coprocessor 7120P platforms. The Xeon E5-2630 platform has 2 sockets with 6-cores per socket, i.e. a total of 12 physical cores or 24 logical cores with hyper-threading enabled. It has three levels of cache hierarchy with 32 KB of L1 (I/D), 256 KB of L2 and 15 MB of L3, and 32 GB of main memory running at 1600 MHz. This supports per-core DVFS with 15 levels ranging from 1.2 GHz to 2.6 GHz in 100 MHz steps. The Xeon Phi has an L2 cache size of 30.5 MB and 16 GB of main memory. All cores share a common *V-f* island and the frequency can be varied from 619 MHz to 1238 MHz in nine steps with corresponding voltage ranging from 0.995 V to 1.060 V. To measure the energy consumption of the cores and main memory, the read-only model specific register (MSR) `MSR_PP0_ENERGY_STATUS` and `MSR_DRAM_ENERGY_STATUS` are sampled for every 50 ms, sufficient enough considering the DVFS time slice of 100 ms. These MSRs are updated every $\sim$1 ms with a wraparound time of around 60 secs when power consumption is high and may be longer otherwise.

The Rodinia [30] and NAS parallel [29] OpenMP benchmarks are used to demonstrate the efficacy of the proposed runtime energy management approach. The classes of NAS and input to Rodinia benchmarks are chosen such that their execution times are sufficiently large (more than five seconds). The details of benchmarks are given in Table I. For Xeon E5-2630 and Phi, the number of application threads is set to 24 and 61, respectively. These applications are executed in single, double and triple application scenarios (due to space limitations, we are unable to report the results for four or more concurrent applications). The experimental results are collected by running each scenario for ten times and finally,

TABLE I
DETAILS OF SELECTED APPLICATIONS FROM RODINIA [30] AND NAS PARALLEL [29] BENCHMARKS

Benchmark	Application Name	Domain	Abbreviation
Rodinia	Breadth-First Search	Graph Algorithms	bfs
	HotSpot	Physics Simulation	hs
	K-means	Data Mining	km
	lavaMD	N-body Simulation Algorithms	lmd
	Myocyte	Medical	mc
	Needleman-Wunsch	Bioinformatics	nw
	Particle Filter	Object tracking	pf
	Stream Cluster	Data Mining	sc
	Speckle Reducing Anisotropic Diffusion	Image Processing	srad
NAS	Block Tri-diagonal solver	computational fluid dynamics pseudo-application	bt
	Scalar Penta-diagonal solver		sp
	Lower-Upper Gauss-Seidel solver		lu
	Conjugate Gradient	computational fluid dynamics kernel	cg
	Embarrassingly Parallel		ep
	Multi-Grid		mg
	Data Cube	data movement	dc

their average values (energy and performance) are computed for the comparison.

The proposed technique is compared against Linux's conservative (*CONS*), ondemand (*OD*) and performance (*PERF*) power governors, which are implemented on millions of devices, making them competitive baselines [31]. In addition to that, we also considered the approach presented in [4] for comparison. To demonstrate the advantage of taking the underlying NUMA architecture into account while estimating the workload, two variants of the proposed (*prop*) approach, *prop-NNUMA* (*NNUMA* stands for 'No NUMA') and *prop-NUMA*, are derived. As opposed to *prop-NNUMA*, *prop-NUMA* takes NUMA latency into account while deciding the *V-f* setting. For the better representation, we have normalized the energy consumption of evaluated approaches to the energy consumption obtained by *prop-NUMA*.

A. Estimation of coefficients

The accuracy of the predicted workload as compared to the actual workload of the prior time intervals depends on the coefficients θ, γ and ζ (refer to Equation (1), (2) and (4)). The value of γ is computed from α and β (refer to Equation (3)). These coefficients are experimentally obtained by executing a diverse set of applications individually and concurrently. Finally, considering the relative workload prediction accuracy, θ, ζ, α and β are set to 2, 0.05, 0.3 and 0.6, respectively. The same coefficient values are used for all the application scenarios.

B. Evaluation of Workload Prediction

To evaluate the workload prediction accuracy, various application scenarios (executing single and multiple applications) are considered. For a set of 40 application scenarios, the average error in workload prediction was 5.4% with a minimum and maximum error of 0.5% and 9.3%, respectively.

C. Evaluation on Xeon E5-2630

1) Energy Savings: In the case of a single-application scenario, there is only one active application. The number of cores allocated to each application is 24; the same as the number of logical cores available on Xeon E5-2630. Fig. 3 shows a comparison of the adopted approaches, *prop-NUMA* and *prop-NNUMA*, with existing techniques in terms of normalized energy consumption. It can be observed that the proposed approach *prop-NUMA* outperforms existing approaches, except for the application *sc* executing under the *CONS* power governor. Moreover, *prop-NNUMA* also achieves better energy savings than reported approaches, except for applications *sc* and *ep* executing under *CONS* and *PERF* power governors, respectively. The proposed *prop-NUMA* achieves energy savings up to 57.9%, 60.3%, 61.6% compared to *CONS*, *OD* and *PERF*, respectively. Furthermore, on an average, *prop-NUMA* consumes 3.2% less energy than *prop-NNUMA*.

For double and triple application scenarios, two and three applications are executed concurrently. In this evaluation, available cores on the platform are equally shared among the applications; for example, an application gets 12 cores in the double application scenario. This has been ensured by setting the core affinity of each application at the start of its execution. Concurrent execution increases the contention on memory, which actually suggests scaling down of *V-f* setting to exploit the increased data access latency for energy efficiency. Unlike existing approaches [4], [31], the proposed technique efficiently estimates memory contention, thereby minimizing energy consumption. Fig. 4 gives the normalized energy consumption for various approaches executing two applications concurrently. The proposed *prop-NUMA* approach improves energy consumption by up to 77.1%, 79.4% and 79.5% compared to *CONS*, *OD* and *PERF*, respectively. For the application *bt-nw* scenario, *prop-NUMA* and *PERF* have similar energy consumption values. Furthermore, *prop-NNUMA* achieves an average energy saving of 7.8% with a

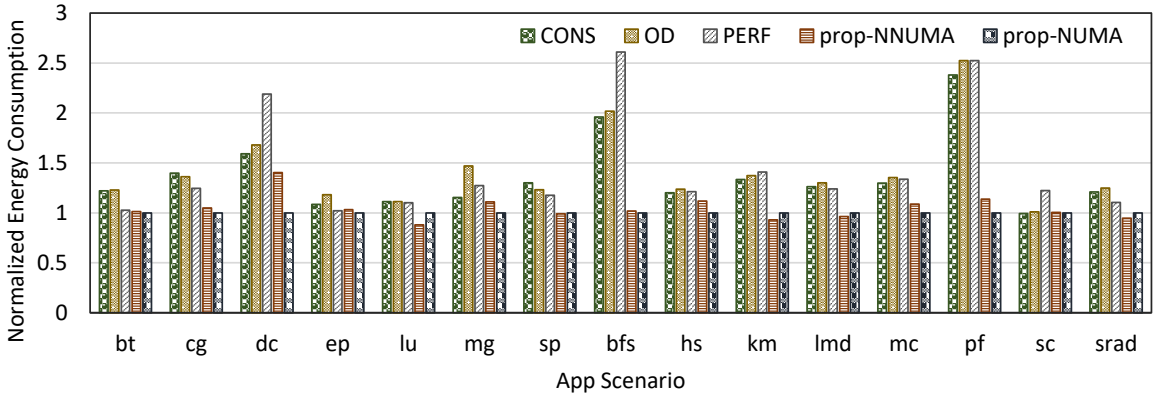

Fig. 3. Comparison of the proposed technique with reported approaches for single application scenario in terms of normalized energy consumption, executing on the Xeon E5-2630.

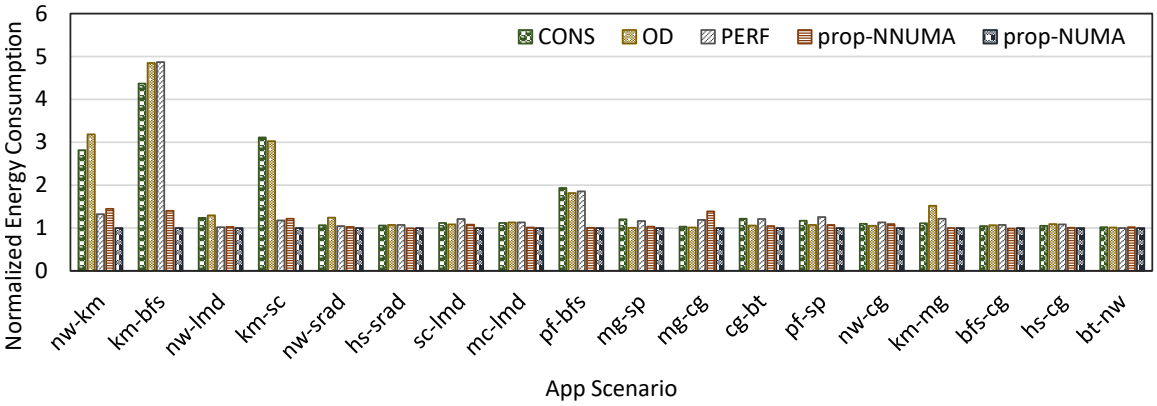

Fig. 4. Normalized energy consumption of various approaches for double application scenario, executing on the Xeon E5-2630.

standard deviation of 10.7% compared to *prop-NUMA*.

The proposed approach is also compared against the reported techniques in terms of normalized energy consumption for a triple application scenario. As shown in Fig. 5, *prop-NUMA* outperforms *CONS*, *OD* and *PERF* by up to 81.2%, 77.9% and 69.8%, respectively. From Fig. 3, 4 and 5, it can be observed that the average energy savings of *prop-NUMA* over *prop-NNUMA* keeps increasing, which is 9.4% with a standard deviation of 8.9%. This suggests that the advantage of considering NUMA latency is more evident when multiple applications are executing concurrently, leading to increased *LLC*-misses and remote memory accesses.

2) Application Performance: The application performance is evaluated for various application scenarios by computing the average execution time over 10 runs. The proposed technique achieves energy savings by scaling down the *V-f* setting if it does not result in performance loss. However, hardware platforms do not usually support fine-grained control of *V-f* setting, leading to performance degradation. Fig. 6 shows the mean and standard deviation of the performance difference (%) between *prop-NUMA* and other approaches. The average execution time for *prop-NUMA*, considering different application scenarios, is 0.66% and 0.47% less compared to *CONS* and *OD*, respectively. However, *PERF*, which runs at the maximum *V-f* setting, outperforms *prop-NUMA* by 4.18% (average). This shows that proposed approach improves energy efficiency with negligible performance loss in the most cases.

D. Evaluation on Xeon Phi

1) Energy Savings: To show the effectiveness of the proposed approach on a platform supporting system-wide DVFS, we also conducted experiments on the Xeon Phi with various application scenarios. It should be noted that the Xeon Phi supports only simultaneous sampling of two PMCs. To address this issue, time multiplexing can be used to monitor more than two events; however, it increases the uncertainties in event count and runtime overheads [32]. Therefore, the `/proc sysfs` is used for measuring the core utilisation. Furthermore, the remote memory accesses, used in *prop-NUMA*, are not considered due to the above limitation. As shown in Fig. 7, on an average, considering all the application scenarios, *prop-NNUMA* improves energy efficiency by up to 54.3%, 60.9%, and 60.4% compared to *CONS*, *OD* and *PERF*, respectively. These energy savings relatively lesser (16%) compared to the ones achieved by per-core DVFS supporting platform (refer to Fig. 3, 4 and 5).

2) Application Performance: The mean and standard deviation of the performance difference between *prop-NNUMA*

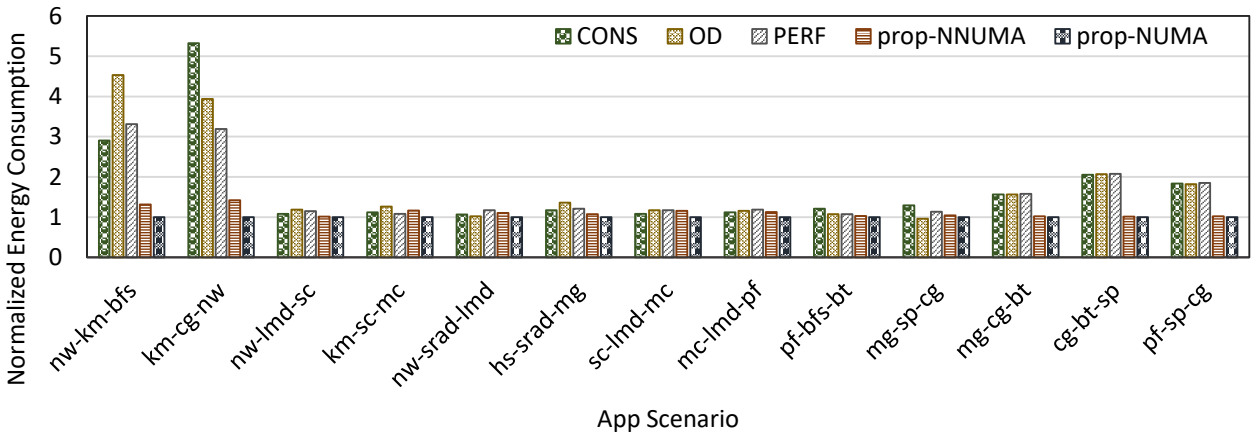

Fig. 5. Comparison of the proposed approach with reported approaches for triple application scenario in terms of normalized energy consumption (evaluated on the Xeon E5-2630).

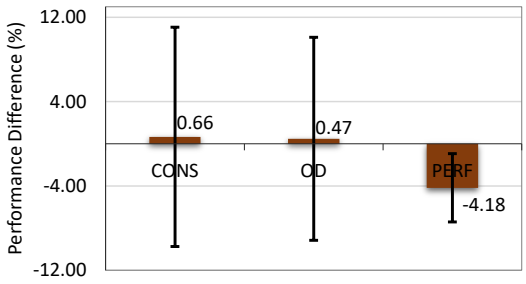

Fig. 6. Mean performance difference between *prop-NUMA* and other approaches, with standard deviation error bars (evaluated on the Xeon E5-2630).

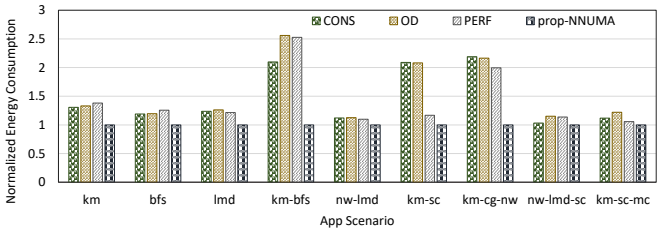

Fig. 7. Comparison of proposed approach (*prop-NNUMA*) with reported approaches in terms of normalized energy consumption for various application scenarios, executing on the Xeon Phi.

and other approaches are given in Fig. 8. Unlike the per-core DVFS platform, in this case the *V-f* setting of the whole system is decided by the most compute-intensive thread of all the executing applications. As a result, performance loss is minimized at the cost of lower energy efficiency, which is evident from the Fig. 7 and 8. The average execution time for *prop-NNUMA*, considering nine application scenarios, is 0.66% and 0.47% better compared to *CONS* and *OD*, respectively. But, as expected, *PERF* outperforms *prop-NNUMA* by 1.19%.

E. Comparison with Sundriyal et al. [4]

Sundriyal et al. [4] presented a model, which aims to predict the micro-operations retired at different frequencies and the memory accesses per micro-operation (MAPM) values by

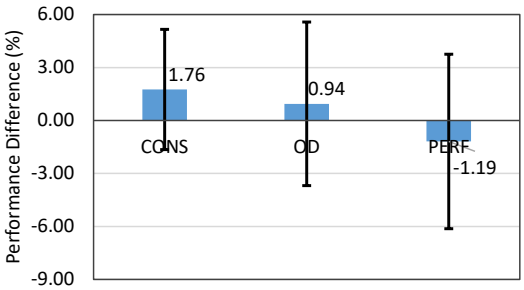

Fig. 8. Mean performance difference between *prop-NUMA* and other approaches, with standard deviation error bars (evaluated on the Xeon Phi).

using a linear regression analysis. It further chooses the scaling frequency at which the energy consumption is minimum by measuring the power using the Intel RAPL technology. This approach is proposed for single application scenario considering per-core DVFS. For comparison, the evaluation is carried out on the Xeon E5-2630 using *ep*, *cg*, *lu*, *mg*, *sp* and *bt* (same applications used in [4]). The approach proposed in [4] gives an average of 7.4% energy savings compared to *PERF* with a performance loss of 5.58%. Whereas, proposed approach (*prop-NUMA*) achieves an average energy savings of 17.8% with a performance loss of 1.8%. It shows that proposed approach outperforms the technique presented in [4] by 10.4% in energy efficiency and 3.7% in performance.

F. Runtime Overheads

Proposed approach involves sampling of four PMCs (only two on the Xeon Phi) on each core and subsequent processing to find and change the *V-f* setting. For all the application scenarios used in the evaluation, we have measured the runtime overhead of our approach on Xeon E5-2630 and Phi by monitoring the time spent in each decision epoch (100 ms). A maximum overhead of 0.52% and 1.3% (in terms of application execution time) is observed for Xeon E5-2630 and Phi, respectively.

VI. Conclusions

The need for energy efficiency in HPC systems has been increasing in recent years, to minimize operating costs and improve lifetime. To address this, we presented a workload-aware runtime energy management technique that takes the combined effect of application compute-/memory-intensity, thread synchronization contention, and non-uniform memory accesses (NUMAs), for controlling the *V-f* setting. Our approach showed that accurate estimation of the processor workload periodically is important for efficient *V-f* control. In the case of concurrent execution, it has been demonstrated that a careful selection of workload and taking the memory contention into account for *V-f* control improve energy efficiency with a low performance loss. Experimental results showed energy savings of up to 81.2% with negligible performance loss when compared to existing approaches. Our future work involves incorporating the performance-awareness and task mapping to adapt to performance variations across different applications.

Acknowledgements

This work was supported in part by the EPSRC under EP/L000563/1 and EP/K034448/1 (the PRiME Programme, www.prime-project.org). Experimental data used in this paper can be found at http://doi.org/10.5258/SOTON/D0517.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *Communications of the ACM*, vol. 53, no. 6, p. 50, 2010.

[2] J. Shuja, K. Bilal, S. A. Madani, M. Othman, R. Ranjan, P. Balaji, and S. U. Khan, "Survey of techniques and architectures for designing energy-efficient data centers," *IEEE Systems Journal*, vol. 10, no. 2, pp. 507–519, 2016.

[3] Z. Lai, K. T. Lam, C.-L. Wang, and J. Su, "Latency-aware dvfs for efficient power state transitions on many-core architectures," *J. Supercomput.*, vol. 71, no. 7, pp. 2720–2747, 2015.

[4] M. S. Vaibhav Sundriyal, "Runtime power-aware energy-saving scheme for parallel applications," in *Iowa State University Computer Science Technical Reports*, 2015, p. 17.

[5] G. Lawson, V. Sundriyal, M. Sosonkina, and Y. Shen, "Runtime power limiting of parallel applications on intel xeon phi processors," in *2016 4th International Workshop on Energy Efficient Supercomputing (E2SC)*, 2016, pp. 39–45.

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

[7] R. Ge, X. Feng, W.-c. Feng, and K. W. Cameron, "Cpu miser: A performance-directed, run-time system for power-aware clusters," in *Parallel Processing, 2007. ICPP 2007. International Conference on*. IEEE, 2007, pp. 18–18.

[8] C.-h. Hsu and W.-c. Feng, "A power-aware run-time system for high-performance computing," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. IEEE Computer Society, 2005, p. 1.

[9] X. Feng, R. Ge, and K. W. Cameron, "Power and energy profiling of scientific applications on distributed systems," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE, 2005, pp. 10–pp.

[10] B. Rountree, D. K. Lownenthal, B. R. De Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: making DVS practical for complex HPC applications," in *Proceedings of the 23rd international conference on Supercomputing*. ACM, 2009, pp. 460–469.

[11] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal, "Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs," in *SC 2006 conference, proceedings of the ACM/IEEE*. IEEE, 2006, pp. 14–14.

[12] V. W. Freeh and D. K. Lowenthal, "Using multiple energy gears in MPI programs on a power-scalable cluster," in *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM, 2005, pp. 164–173.

[13] B. Li, H.-C. Chang, S. Song, C.-Y. Su, T. Meyer, J. Mooring, and K. W. Cameron, "The power-performance tradeoffs of the intel xeon phi on HPC applications," in *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*. IEEE, 2014, pp. 1448–1456.

[14] J. Wood, Z. Zong, Q. Gu, and R. Ge, "Energy and power characterization of parallel programs running on intel xeon phi," in *Parallel Processing Workshops (ICCPW), 2014 43rd International Conference on*. IEEE, 2014, pp. 265–272.

[15] T. Leng, R. Ali, J. Hsieh, V. Mashayekhi, and R. Rooholamini, "An empirical study of hyper-threading in high performance computing clusters," *Linux HPC Revolution*, vol. 45, 2002.

[16] N. Manchanda and K. Anand, "Non-uniform memory access (numa)," *New York University*, vol. 4, 2010.

[17] M. Etinski, J. Corbalan, J. Labarta, M. Valero, and A. Veidenbaum, "Power-aware load balancing of large scale MPI applications," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 1–8.

[18] A. Marathe, P. E. Bailey, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, "A run-time system for power-constrained HPC applications," in *International conference on high performance computing*. Springer, 2015, pp. 394–408.

[19] B. K. Reddy, G. V. Merrett, B. M. Al-Hashimi, and A. K. Singh, "Online concurrent workload classification for multi-core energy management," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 621–624.

[20] B. K. Reddy, A. K. Singh, D. Biswas, G. V. Merrett, and B. M. Al-Hashimi, "Inter-cluster thread-to-core mapping and DVFS on heterogeneous multi-cores," *IEEE Transactions on Multi-Scale Computing Systems*, 2017.

[21] S. Eranian, "Perfmon2: a flexible performance monitoring interface for linux," in *Proc. of Ottawa Linux Symposium*. Citeseer, 2006, pp. 269–288.

[22] B. K. Reddy, A. K. Singh, G. V. Merrett, and B. M. Al-Hashimi, "ITMD: Run-time management of concurrent multi-threaded applications on heterogeneous multi-cores," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017.

[23] S. Sinha *et al.*, "Workload-aware neuromorphic design of the power controller," *IEEE JETCAS*, vol. 1, no. 3, pp. 381–390, 2011.

[24] G. Liu, J. Park, and D. Marculescu, "Dynamic thread mapping for high-performance, power-efficient heterogeneous many-core systems," in *Computer Design, 31st International Conference on*. IEEE, 2013, pp. 54–61.

[25] A. Phansalkar, A. Joshi, and L. K. John, "Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 412–423, 2007.

[26] L. McVoy and C. Staelin, "Lmbench: Portable tools for performance analysis," in *Proc. of the 1996 Annu. Conf. on USENIX Annual Technical Conference*, ser. ATEC '96. Berkeley, CA, USA: USENIX Association, 1996, pp. 23–23.

[27] R. Longbottom, "Roy Longbottom's PC Benchmark Collection," http://www.roylongbottom.org.uk, September 2014, [Online; accessed 2-June-2015].

[28] C. Bienia *et al.*, "The parsec benchmark suite: Characterization and architectural implications," in *Proc. of intl. conf. on Parallel architectures and compilation techniques*. ACM, 2008, pp. 72–81.

[29] H.-Q. Jin, M. Frumkin, and J. Yan, "The OpenMP implementation of NAS parallel benchmarks and its performance," 1999.

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

[31] X. Developers, "XDA-DevelopersForums," accessed 2018-03-21. [Online]. Available: https://forum.xda-developers.com/general/general/ref-to-date-guide-cpu-governors-o-t3048957

[32] D. Zaparanuks, M. Jovic, and M. Hauswirth, "Accuracy of performance counter measurements," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 23–32.