Online Concurrent Workload Classification for Multi-core Energy Management

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

Amit Kumar Singh
University of Essex, United Kingdom
a.k.singh@essex.ac.uk

***Abstract*—Modern embedded multi-core processors are organized as clusters of cores, where all cores in each cluster operate at a common Voltage-frequency (*V-f*). Such processors often need to execute applications concurrently, exhibiting varying and mixed workloads (e.g. compute- and memory-intensive) depending on the instruction mix and resource sharing. Runtime adaptation is key to achieving energy savings without trading-off application performance with such workload variabilities. In this paper, we propose an online energy management technique that performs concurrent workload classification using the metric Memory Reads Per Instruction (MRPI) and pro-actively selects an appropriate *V-f* setting through workload prediction. Subsequently, it monitors the workload prediction error and performance loss, quantified by Instructions Per Second (IPS) at runtime and adjusts the chosen *V-f* to compensate. We validate the proposed technique on an Odroid-XU3 with various combinations of benchmark applications. Results show an improvement in energy efficiency of up to 69% compared to existing approaches.**

I. Introduction and Motivation

To achieve energy efficiency, modern embedded multi-core platforms support dynamic voltage and frequency scaling (DVFS) which enables on-the-fly linear reduction of frequency (f) and voltage (V), yielding a cubic reduction in dynamic power consumption ($\propto V^2 f$). Considering the complexity of hardware design, such architectures with a fixed number of cores are organized as clusters, where all cores in a cluster operate at the same *V-f* (cluster-wide DVFS) [1], [2].

Multi-cores often execute applications concurrently, where each application exercises the hardware differently based on its instruction mix. This, coupled with resource sharing results in varying and mixed workloads (compute-intensive, memory-intensive, etc.). Managing such workload variations on multi-core architectures supporting cluster-wide DVFS is a challenging task, as each core may execute a different type of workload, and thus an appropriate *V-f* for the entire cluster has to be determined.

Fig. 1 shows the variation in workloads when multiple applications are run in two different configurations - individually (left) and concurrently (right) on the A15 cluster of the Odroid-XU3 platform. Here we consider three applications having different workload profiles from SPEC CPU 2006: *astar*, *lbm* and *mcf*, and their various combinations *astar-lbm*, *astar-mcf*, *lbm-mcf* and *astar-lbm-mcf*. Further, a metric called Memory Reads Per Instruction (MRPI = last level cache misses/instructions retired) is derived for classifying the workload. The reason

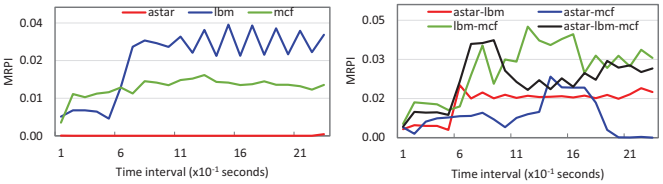

Fig. 1: Variation in MRPI for individual (left) and concurrent (right) execution of applications.

for using MRPI as opposed to the commonly used CPU cycles, Instructions Per Cycles (IPC), or utilisation [3]–[5] is as follows. CPU cycles, utilisation and IPC are usually affected not only by number of memory accesses but also by other statistics, such as lower-level cache misses, branch mispredictions, etc., whose penalty (measured in cycles) does not get affected by frequency scaling. Therefore, to efficiently select the *V-f* setting based on the memory-intensiveness of the application, the MRPI metric is chosen. A high load on the processing core indicates a low MRPI and vice versa.

It can be observed from Fig. 1 that the different workload types (of the applications astar, lbm and mcf) can clearly be classified (e.g. memory-or compute-intensive) when run individually, which is completely different in the case of concurrent execution having greater workload variability. This necessitates an efficient online classification of concurrent workloads to understand the above effects and select an appropriate *V-f* setting for achieving energy efficiency. To this end, this paper makes the following contributions:

1) An approach for workload selection, prediction and classification that proactively controls the *V-f*;
2) A low overhead technique for identifying cores, executing no application, to avoid selection of a high *V-f*;
3) Implementation and validation of the proposed approach on a real hardware platform, the Odroid-XU3 [1].

II. Related Work

Various energy optimization approaches have been adopted for single and multiple applications executing individually by controlling the *V-f* setting with respect to workload behaviour [6]–[8]. A simulation based approach for energy savings due to the reduced cache miss penalties at lower clock frequencies is proposed in [6]. Compile-time techniques, such as [7], have exploited the stall periods. On the other hand, online algorithm utilizing the hardware PMCs is proposed in [8] to achieve energy saving without recompiling the applications.

978-3-9819263-0-9/DATE18/©2018 EDAA

The aforementioned approaches consider single application at a time and thus cannot be applied to concurrent applications. Moreover, they are applicable for per-core DVFS based multi-core architectures. Identification of *V-f* setting on per-core DVFS based multi-cores is relatively simple compared to cluster-wide DVFS multi-cores, as *V-f* of one core does not affect the workload running on another core.

Approaches proposed in [9], [10] consider cluster-based architectures. Kong et al. [9] proposed energy-efficient scheduling of real-time tasks and identification of appropriate *V-f* setting, which was validated on a simulation platform. In [10], an offline regression-based technique for task mapping and DVFS is proposed for concurrent applications. However, this approach heavily depends on offline analysis results, which is non-scalable. Furthermore, none of the aforementioned approaches consider online concurrent workload classification and the *V-f* level is not adjusted during execution, which is beneficial for adapting to workload variations.

III. Proposed Approach

The proposed approach addresses the following problem:

Given a set of concurrent applications and a multi-core platform supporting cluster-wide DVFS

Optimize energy consumption while maximizing the performance by selecting efficient *V-f* setting

An overview of the proposed approach is shown in Fig. 2 having the following stages:

(a) workload selection and prediction
(b) workload classification and frequency selection
(c) performance evaluation and compensation

A detailed discussion on each stage is presented in the following sections. As the device firmware automatically adjusts the voltage for a selected frequency, we refer to *V-f* and frequency interchangeably throughout the paper.

A. Workload Selection and Prediction

An appropriate *V-f* setting depends on the application workload. When applications execute individually, *V-f* setting is mostly guided by a single workload [3]. However, for concurrent execution, it depends on multiple workloads. It is important to note that, for concurrently executing applications on a cluster-based multi-core, *V-f* of each cluster should be chosen in such a way that all the applications meet their performance requirements. Furthermore, these applications generate varying and mixed workloads due to resource sharing (e.g. cache and memory). Therefore, a representative *V-f* setting has to be chosen for achieving energy efficiency without degrading application performance.

1) Workload Selection: Let us assume that there are *N* concurrently executing applications on a multi-core and $\mathtt{mrpi}_{ni}$ be the MRPI of an application *n* for time interval $t_{i-1} \rightarrow t_i$. There will be *N* different workloads at every time interval of the execution. The workload is quantified by the MRPI, where a low value represents a high load on the processing core and vice versa. Due to limited resource availability, concurrent execution of applications creates contention on shared resources, especially on memory, impacting performance of individual application, i.e. increases the execution time.

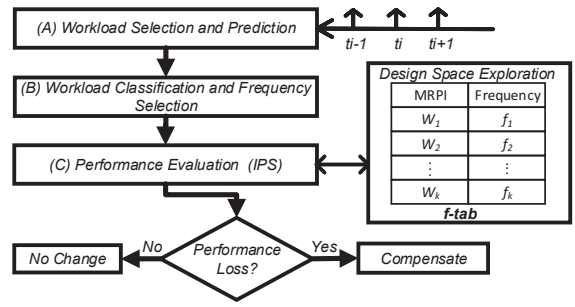

Fig. 2: Proposed online energy management technique.

Contention on memory can be exploited to scale down the frequency to minimize the wasted cycles. We take this into account by increasing the MRPI of each core by δ, calculated at runtime based on the average MRPI of the running applications. If all the running applications are memory-intensive, then the value of δ will be high due to increased memory traffic. Considering the above and to maximize the application performance, *V-f* setting for time interval $t_{i-1} \rightarrow t_i$ (considering cluster-wide DVFS) is influenced by the workload with minimum MRPI ($\mathtt{mrpi}_{target\_i}$),

$$\mathtt{mrpi}_{target\_i} = \min\{mrpi_{1i}, mrpi_{2i}, ..., mrpi_{Ni}\} + \delta_i \quad (1)$$

2) Workload Prediction: To adapt to workload variations and to achieve energy minimization, proactive control of *V-f* is of utmost importance. Therefore, the future workload (t_{i+1}) needs to be predicted at t_i to set the appropriate *V-f* value for the time interval $t_i \rightarrow t_{i+1}$. To accomplish this, we use an exponential weighted moving average (EWMA) filter [11] to predict MRPI (p_{i+1}) for the time interval $t_i \rightarrow t_{i+1}$,

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

where γ, p_i and a_i are the smoothing factor, predicted and actual MRPI values respectively during the interval $t_{i-1} \rightarrow t_i$. It is to be noted that $\mathtt{mrpi}_{target\_i}$ computed from Equation 1 represents the actual workload a_i. To minimize miss-predictions, the predicted MRPI of the interval $t_{i-1} \rightarrow t_i$ is compared to the actual MRPI measured from hardware PMCs. Subsequently, computed prediction error P_e (difference between actual and predicted MRPI values) is used to improve the 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, the value of γ is changed in proportion to p_{wc} (=P_e/p_i),

$$\gamma = \alpha \times P_{wc} + \beta \quad (3)$$

The values of coefficients α and β are given in Section IV.

B. Workload Classification and Frequency Selection

Classification of the predicted workload is important for identifying an appropriate *V-f* setting for achieving energy savings without any performance loss. For online workload classification, two hardware PMCs (L2 data cache refills and instructions retired) are used for periodically computing the

MRPI during application execution. The modified `perfmon` tool [11] is used for accessing the PMCs.

To minimize the runtime overhead, workload types are predetermined through a custom program, generating varying number of memory accesses. The custom program copies varying amount of data from one memory location to another, like memcpy(), after doing addition and multiplication operations on two large arrays. At different number of memory accesses, the variation in MRPI and execution time is recorded by sweeping the frequency from 0.2 GHz to 2 GHz on A15 cluster of Odroid-XU3. The offline profiling results contain MRPI ranges and corresponding appropriate *V-f* settings (`f-tab` in Fig. 2), which are used at runtime to set the operating frequency to a desired value through the utility `cpufreq-set`. The range of MRPI values having little (<1%) or no effect on execution time for the same frequency are grouped into a single class (same workload type). Workloads with large MRPI are assigned to a low frequency and it is decided by the speed of memory (933 MHz in our case). Similarly, workload having a significantly low MRPI, i.e. execution time scales linearly with frequency, is assigned a maximum available frequency.

1) Identification of unused cores: MRPI of unused cores, i.e. no application is executing on those cores, is usually low due to fewer memory accesses. As a result, if such cores are not identified, it gives a miss-impression that the cores are executing a compute-intensive application, leading to selection of a high *V-f* and thus increasing energy consumption. This becomes prominent when there are more cores than number of concurrent applications in a cluster. To address this, the proposed algorithm determines unused cores at runtime using an IPS threshold. If IPS of a core is below the threshold value, it is identified as unused core and subsequently, its MRPI is set to 10 (any value larger than one would be fine as the value of MRPI usually does not exceed one). This eliminates the influence of unused cores on *V-f* setting, which is decided by the minimum MRPI of the applications (Equation 1). The IPS threshold value is experimentally identified as 2.9×10^6 for the cores in A15 cluster of Odroid-XU3, which is used for experimental validation. Determining unused cores using IPS does not need extra PMCs as the number of instructions retired during each time interval is already made available for computing MRPI. Otherwise, checking unused status with generally used CPU utilisation [5] needs an extra PMC (number of active CPU cycles) or accessing `/proc` virtual filesystem (`procfs`), leading to increased runtime overhead.

C. Performance Evaluation and Compensation

Considering the dynamic resource availability and interference between concurrent applications, it is important to evaluate the performance during execution to ensure that no application experiences any performance loss. Therefore, we use instructions per second (IPS) as a metric for quantifying the runtime performance of each application for every elapsed time interval T_s. The performance loss is calculated by comparing the IPS_n for every time interval with the maximum IPS (IPS_{max}) achieved by executing the application at the highest available frequency (f_{max}). If there is a performance loss of λ% during the interval $t_{i-1} \rightarrow t_i$, the selected *V-f* is increased by $\lambda\times f_{max}$ for subsequent time interval ($t_i \rightarrow t_{i+1}$) to compensate it. Furthermore, the frequency is modified only when λ is significant to minimize the overheads associated with DVFS. We experimentally verified and set the value of λ to 1% by taking the variations in PMC data into account.

IV. Experimental Results

The proposed approach is validated on an Odroid-XU3 platform running Ubuntu Linux Kernel 3.10.96. The Odroid-XU3 has eight cores, which are organized into big and LITTLE clusters with four A15 and A7 cores respectively, and 2 levels of cache hierarchy. Moreover, each cluster operates in a different power domain. As part of our experiments, the A15 cluster only is considered, which supports 19 *V-f* pairs (200 MHz - 2000 MHz with 100 MHz steps). However, without the loss of generality, similar experiments can be carried out on LITTLE cluster and any other cluster-based architectures.

To show the effectiveness of the proposed approach, applications *lbm* (lb), *milc* (mi), *mcf* (mc) and *bwaves* (bw) from SPEC CPU2006 [12], and *swaptions* (sw) and *freqmine* (fr) from PARSEC [13] are considered. These applications are executed concurrently in single, double and triple combinations. The power is measured from the on-board power sensors of Odroid-XU3 every 100ms. The proposed technique is compared against Linux's conservative, ondemand and interactive power governors, which are implemented on millions of smartphones, making them competitive baselines [14]. These governors come under utilisation-based approaches as they scale the frequency based on utilisation threshold. Further, we also considered IPC-based [5] and exhaustive search-based, similar to [10], approaches for the comparison. For exhaustive search-based approach, each application scenario is executed at all available frequencies and the one with minimum energy consumption is selected while having the same or better performance than the proposed approach. Energy consumption values of evaluated approaches are normalized to the energy consumption obtained by running the proposed approach.

A. Energy Savings

Energy consumption of various approaches for single, double and triple application scenarios is shown in (a), (b) and (c) of Fig. 3 (A), respectively. In case of single application scenario, proposed approach achieves up to 68% energy savings compared to reported approaches.

The proposed approach efficiently adapts to concurrent workload variations compared to existing techniques and selects an appropriate *V-f* setting, as shown in Fig. 3 (B). Further, it also considers the latency due to memory contention through δ (Equation 1), whose value is experimentally identified as 4.5% of average MRPI of all the cores in a cluster. For multi-application scenario, proposed approach improves the energy efficiency by up to 69% when compared with the existing techniques.

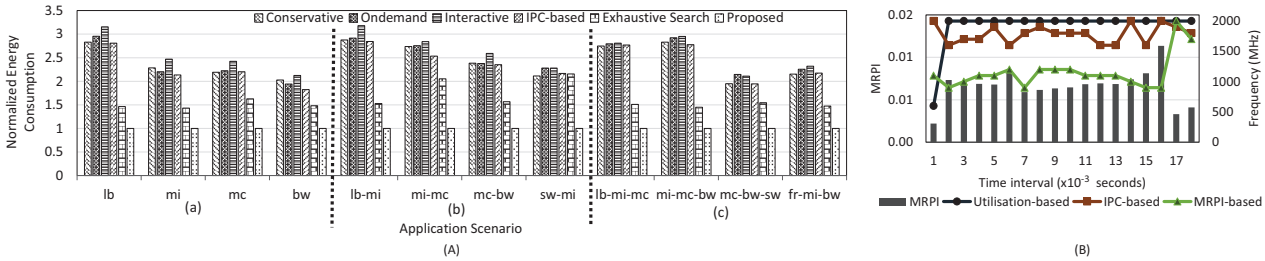

Fig. 3: Comparison of proposed approach with the existing approaches. (A) Normalized energy consumption for single, double and triple application scenarios. (B) MRPI and frequency at different time intervals of the application scenario *lb-mi*.

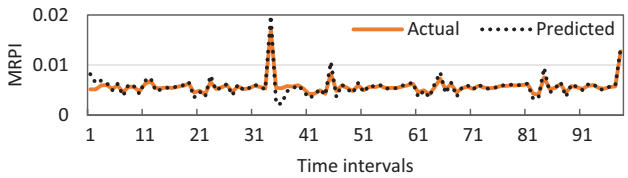

Fig. 4: Workload prediction for the application scenario *mi-mc*.

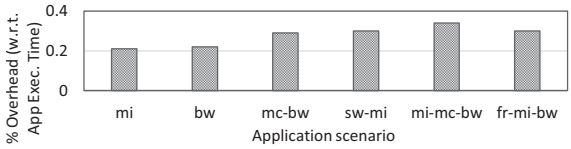

Fig. 5: Runtime overhead of the proposed approach.

B. Application Performance

The application performance is evaluated for various application scenarios by computing the average execution time over several runs. The proposed technique continuously monitors the application performance in terms of IPS for every time interval and consequently, modifies the chosen frequency if there is a significant performance loss ($> 1\%$). As a result, the proposed approach achieves significantly better energy savings with a little performance loss. The average execution time for the proposed approach, considering different application scenarios, is 4.85%, 3.62%, 3.10%, 3.22% and 0.9% slower compared to interactive, conservative, ondemand, IPC-based and exhaustive search-based approaches, respectively.

C. Workload Prediction

The values of α and β in Equation 3 were experimentally obtained by sweeping them between 0 and 1, and observing the corresponding workload miss-predictions (under/over) for various application scenarios. Finally, a value of 0.3 and 0.6 are chosen as it resulted in relatively accurate workload prediction. Fig. 4 shows the actual and predicted MRPI for the application scenario *mi-mc* at different time intervals. The average error in workload prediction, considering all application scenarios used in evaluation, is 4.2%.

D. Runtime Overheads

Fig. 5 shows the runtime overhead of the proposed technique for various application scenarios as a percentage of application execution time. A maximum overhead of 0.35% is observed for *mi-mc-bw*, having a long execution time of 111 seconds. This shows that proposed approach has negligible runtime overhead.

V. Conclusions

We have proposed an online energy minimization approach for concurrently executing applications on a multi-core platform using workload classification and prediction for appropriately selecting the *V-f* setting. Validation on a real multi-core hardware platform for various application scenarios shows an improvement of up to 69% in energy efficiency compared to existing approaches. Our future work includes incorporating task mapping and DVFS techniques for concurrent multi-threaded applications on heterogeneous architectures.

Acknowledgements

This work was supported in parts by the EPSRC Grant EP/L000563/1 and the PRiME Programme Grant EP/K034448/1 (www.prime-project,org). Experimental data used in this paper can be found at http://doi.org/10.5258/SOTON/D0308.

References

[1] "Odroid-XU3," www.hardkernel.com/main/products.
[2] "Mediatek helio X20," http://www.96boards.org/product/mediatek-x20/.
[3] A. Das, B. M. Al-Hashimi, and G. V. Merrett, "Adaptive and hierarchical runtime manager for energy-aware thermal management of embedded systems," *ACM TECS*, vol. 15, no. 2, p. 24, 2016.
[4] V. Pallipadi and A. Starikovskiy, "The ondemand governor," in *Proc. of the Linux Symposium*, vol. 2. sn, 2006, pp. 215–230.
[5] A. S. Bischoff, "User-experience-aware system optimisation for mobile systems," Ph.D. dissertation, University of Southampton, 2016.
[6] D. Marculescu, "On the use of microarchitecture-driven dynamic voltage scaling," in *Workshop on Complexity-Effective Design*, vol. 42, 2000.
[7] C.-H. Hsu and U. Kremer, "Compiler-directed dynamic voltage scaling for memory-bound applications," *Technical Report DCS-TR-498, Department of Computer Science, Rutgers University*, 2002.
[8] A. Weissel and F. Bellosa, "Process cruise control: event-driven clock scaling for dynamic power management," in *Proc. of CASES*. ACM, 2002, pp. 238–246.
[9] F. Kong, W. Yi, and Q. Deng, "Energy-efficient scheduling of real-time tasks on cluster-based multicores," in *DATE*. IEEE, 2011, pp. 1–6.
[10] A. Aalsaud *et al.*, "Power-aware performance adaptation of concurrent applications in heterogeneous many-core systems," in *Proc. of ISLPED*. ACM, 2016, pp. 368–373.
[11] S. Sinha *et al.*, "Workload-aware neuromorphic design of the power controller," *IEEE JETCAS*, vol. 1, no. 3, pp. 381–390, 2011.
[12] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.
[13] 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.
[14] "XDA-developersforums," https://forum.xda-developers.com/general/general/ref-to-date-guide-cpu-governors-o-t3048957.