Dynamic Power Management in a Heterogeneous Processor Architecture

Frehiwot Melak Arega, Markus Haehnel, and Waltenegus Dargie

Faculty of Computer Science, TU Dresden, 01062 Dresden, Germany
Email:{frehiwot_melak.arega, markus.haehnel1,
waltenegus.dargie}@tu-dresden.de

Abstract. Emerging mobile platforms integrate heterogeneous, multicore processors to efficiently deal with the heterogeneity of data (in magnitude, type, and quality). The main goal is to achieve a high degree of energy-proportionality which correspond with the nature and fluctuation of mobile workloads. Most existing power and energy consumption analyses of these architectures rely on simulation or static benchmarks neither of which truly reflects the type of workload the processors handle in reality. By contrast, we generate two types of stochastic workloads and employ four types of dynamic voltage and frequency scaling (DVFS) policies to investigate the energy proportionality and the dynamic power consumption characteristics of a heterogeneous processor architecture when operating in different configurations. The analysis illustrates, both qualitatively and quantitatively, that knowledge of the statistics of the incoming workload is critical to determine the appropriate processor configuration.

Keywords: Dynamic Power Management, DVFS, multicore processor, heterogeneous processor architecture, workload

1 Introduction

The volume of data generated and processed by mobile platforms has shown a rapid and sustained increment in the recent years, and evidence suggests that the trend will continue so in the near future. The driving forces behind this development are the introduction of advanced processor architectures and an improving communication infrastructure. At the same time, however, the complexity of the applications which run on the mobile platforms, generating and consuming some of these data, and their corresponding power consumption are increasing with comparable proportion. The implication is that both to meet users' requirements and provide sustainable services, there is a need for efficient resource management strategies which take the peculiar aspects of emerging mobile platforms (specifically, processor architectures) into consideration. One of these resources which should be managed is the energy consumption associated with the computation and communication demand of mobile platforms. Indeed, the research community is endeavouring to improve energy efficiency in

different ways, including noble dynamic voltage and frequency scaling, dynamic thermal management, workload-aware task scheduling, efficient thread-to-core mapping, and seamless runtime task migration [1], [2], [3], [4], [5]. Complementary to the runtime adaptation strategies, effort is also being made both by the academia and the industry to develop energy-efficient and energy-proportional processor architectures including (1) the efficient integration of multicore and heterogeneous processors, (2) fast and efficient simultaneous multi-threading, (3) non-uniform cache architecture, and (4) advanced branch prediction strategies, among others. The energy-proportionality (the ratio of consumed energy to work done) of these architectures has in general been improving at each generation, as their power consumption can be managed at core, processor, socket, and platform levels. The purpose of this paper is to qualitatively and quantitatively analyse the energy proportionality of runtime or dynamic power management strategies in emerging heterogeneous processor architectures. Most existing approaches rely on either simulation or static benchmarks neither of which truly reflects the type of workload these architectures process in reality. In contrast, we make an extensive and experimental investigation using stochastic workload (video transcoding) and four types of dynamic voltage and frequency scaling policies. The rest of the paper is organised as follows: In Section 2, we summarise related work; in Section 3, we introduce our experiment setup; in Section 4, we present and discuss experiment results and share the insight we obtained from the experiment results. Finally, in Section 5, we give concluding remarks and outline future work.

2 Related Work

Improving the performance and energy consumption of emerging heterogeneous processor architectures is an active research area. Some of the proposed approaches target one of the following goals: (1) Determining the optimal task-to-processor assignment strategy, (2) identifying effective and workload sensitive dynamic voltage and frequency scaling strategy for a specific processor configuration, and (3) managing the heat dissipation of processors by adapting operation frequencies to workload. Julio et al. [6] propose a model for estimating the execution duration of individual tasks under a specific clock frequency in a heterogeneous processor architecture without the need for analysing any source code or hardware specification. The resource consumption characteristics of a task is analysed at runtime during the first "hyperperiod" of the execution of a task using performance monitoring counters, computation time, waiting for memory, and "overlapping time" between computation and memory access. An overlap time is defined as the time the processor is executing non-dependent instructions while a memory request is being served. Petrucci et al. [7] propose a mechanism for (a) the optimal mapping of threads to specific cores in a heterogeneous architecture during task allocation and (b) periodic reassignment of threads with the aim on improving the runtime energy consumption while meeting performance requirements of application threads. The resource consumption

characteristic and the different execution phases of running threads is analysed using performance monitoring counters (Instructions Per Cycle (IPC) and Last Level Cache (LLC) miss rate). Similarly, Liu et al [8] propose a generic approach to formulate the map of threads to cores in a heterogeneous architecture using Integer Linear Program (ILP) for any number of threads, cores, and types of cores. Their main goal is to maximise throughput while keeping total power under a given budget. The approach first assigns threads to cores such that the assignment can achieve the highest possible throughput and, then, it performs virtual swapping of threads between adjacent core types. Pricopi et al. [9] investigate the relationship between the performance and power consumption of a heterogeneous architecture by using a large number of performance monitoring counters. Their approach focuses on accounting the way this relationship is affected when executions are migrated from one type of cores to another within the same processor architecture. Hanumaiah [2] aims to minimise the effects of high thermal design power (TDP) by combining different approaches such as performance-per-watt efficiency as a trade-off between performance and power consumption, DVFS, thread migration, and active cooling. Maiti et al. [1] investigate the consequences of configuration mismatch between core frequencies and core states and propose a framework for core selection, thread-to-core mapping and DVFS. The aim is to select the best distribution of jobs and the appropriate DVFS. Unlike the approach in [2], here optimisation serves one of the two purposes: either the energy of a server is minimised under performance constraints or its performance is maximised under power constraints. Likewise, Singla et al. [10] and Prakash et al. [11] propose thermal management strategies as a mechanism to minimise the energy consumption and to maximise the performance of emerging processor architecture. In both cases, the temperature of a processor is determined as a function of its power consumption and maximum operation frequency. From this relationship it was possible to determine the maximum operation frequency which sets the temperature of a processor below a set limit.

Complementary to state-of-the-art, this paper presents a comprehensive investigation of the power and energy consumption characteristics of different processor configurations and dynamic voltage and frequency scaling possibilities in a heterogeneous processor architecture given that the incoming workload of the processor architecture is of stochastic nature.

3 Experiment Setup

Processor Architecture. We employ the Odroid-XU4[1] processor architecture with Ubuntu 14.04 operating system installed for our experiment which consists of eight "big.LITTLE" cores. Four of these are energy-efficient ARM Cortex-A7 cores which are suitable for executing non-time-critical workloads. So they are denoted as "LITTLE" cores. The other four, which are high performance ARM

[1] `http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825`

Cortex-A15 cores, are suitable for executing compute intensive and time critical workloads. They are denoted as "big". The performance difference between the two types of cores is a result of their difference in peak operation frequency (which is 2.0 GHz for the Cortex A15 whilst it is 1.4 GHz for the Cortex-A7) as well as in size. We supplied one and the same task (transcoding the same video) for the two types of cores and let them process the task at different frequencies. On the same Cortex the transcoding duration is more or less inversely proportional to the frequency. But when we compared the transcoding duration between processor types for comparable operation frequency, the difference is considerable. For example, a given transcoding took 33 s on a Cortex-A7 core when it was clocked at 1.4 GHz but it took 20 s with a smaller frequency (1.2 GHz) on the Cortex-A15. The additional performance improvement comes from a higher degree of out-of-order execution as well as a large L2 cache (2 MB of Cortex-A15 vs. 512 kB of Cortex-A7). The little cores have only one in-order execution pipeline. In contrast, the big cores have an out-of-order three-way execution pipeline, so that they can reorder the necessary instructions to utilise the subsets of logic more efficiently.

Table 1: The transcoding time of a single test video file on different cores with different frequencies.

Cortex-A7	68 s @ 600 MHz	33 s @ 1.4 GHz
Cortex-A15	20 s @ 1.2 GHz	13 s @ 2.0 GHz

The performance gain of the Cortex-A15 cores comes at a price of high power consumption because of its more complex and larger circuit architecture (pipeline and caches, for instance). The big cores consume approximately four times more power than the LITTLE cores. On the other hand, the Cortex-A15 are not four times faster (as can be seen in Tab. 1); therefore, they should be used only when their performance is really needed. Since the Cortex-A7 and Cortex-A15 cores have a different clocking range and separate phase-locked-loops (PLLs), their bias voltage and clock frequency can be set independently. Nevertheless, as cores of the same type share one and the same PLL, they run at the same frequency. Because an idling core consumes a certain amount of energy, it may be reasonable to prolong the execution of tasks to minimise idle durations. Alternatively, the load of an underutilised core can be migrated somewhere in order to switch it off entirely. But the wake-up latency of a disabled core is significant in case of load balancing.

Workload. The power consumption characteristic of a processor depends on the workload characteristic. Most existing power consumption analysis techniques [9], [10] rely on static benchmarks which, in reality, do not reflect the workload of typical mobile platforms. Since the magnitude of the workload of mobile platforms experiences fluctuation over time, we generated stochastic workloads in order to reflect this fluctuation.

The process can be described as follows: First of all, using the `Python` statistical tool, we generated random numbers of predetermined probability density functions, means, and variances. Then for each random number belonging to a

given probability density function, we generated a video the size of which corresponds to the random number (hence, the distribution of the video size follows the distribution of the random numbers). Secondly, we picked up the longest video from each probability density function and transcoded it using FFmpeg[2] and registered the time a processor core requires to complete the task. A processor core is 100 % utilised when it undertakes a transcoding task and becomes idle when it completes its task. Thirdly, we divided time into slots. The duration of a time slot equals the time the processor required to complete the longest video. Fourthly, at the beginning of each time slot we randomly picked up a video for each activated core from a known probability density function and transcoded it. Notice that now the time the processor requires to transcode this video is random, as the size of the video is random. As a result, the processor experiences a random idle duration in each slot. Furthermore, the interference of several tasks within a time slot as a result of the competition for L2 cache and memory bandwidth is included. The reason is that they run concurrently on different cores of the same type (L2 cache) as well as different types (memory bandwidth). An example is illustrated in Fig. 1 where the maximum transcoding time and, therefore, the slot duration is 30 s. Thus, we produced the following workload distributions:

- Exponential – $\mathbf{E}(\lambda = 15\,\text{MB})$: 3.6 % of the videos have the maximum video size, which is 50 MB for all the experiments.
- Uniform – $\mathbf{U}(0, 30)$: The size of the videos assigned to a core varies uniformly between 0 and 30 MB.

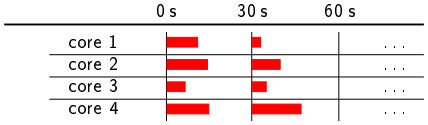

Fig. 1: An example of the workload distribution in a given time slot. A request to transcode videos of different size arrives in the beginning of each time slot; a time slot, for this example is 30 s.

Dynamic Voltage and Frequency Scaling. Most existing dynamic power management strategies aim to minimise the idle states of computing resources, because they consume a significant amount of power even when they are idle. In general, a computing system enters into an idle state due to two stochastically independent phenomena: (1) the interval between the arrival of any two compute tasks is a random process, and (2) there is a discrepancy between the time allocated by a scheduler to process a task (which is usually over-provisioned) and the time the processor requires to complete this task. The idle state can be minimised by either putting a computing resource into a deep sleep mode (or by turning it off altogether) whenever it is idle, by deliberately slowing down its processing speed so that the idle interval is minimised, or by adaptively varying its execution speed, so that execution speed and task completion deadline can overlap. Each approach has its own merits and demerits. One of the merits of the deep-sleep and the slow execution strategies is a significant reduction of idle

[2] `https://www.ffmpeg.org/` (version 2.6.2)

time, but for both strategies this comes at a potential cost of increased execution latency. The adaptive execution frequency strategy foresees the potential change in the interval between the time a task is completed and the arrival of the next task and, therefore, is able to estimate the optimal execution frequency, but for that it requires sufficient task execution and task arrival statistics, as a result of which the gain in power consumption may not be appreciable. For our investigation we selected four types of DVFS policies, namely, *power-save*, *on-demand*, *conservative*, and *performance* policies [12], [13]. The *power-save* policy operates cores at the lowest frequency while the *on-demand* and *conservative* policies adapt the clock frequency to the change in the workload of the servers. The last two policies estimate the utilisation of the processor using a moving average prediction technique, predict its future workload for the next time slot, and scale-down or scale-up the processor's speed accordingly. The essential difference between the two is that the *on-demand* policy scales up the CPU frequency to the maximum whenever an increment in the core's activity is predicted whereas this is done gradually in the *conservative* policy. But both strategies scale down the clock frequency gradually whenever they perceive the future workload as decreasing. The *performance* policy operates a core at its maximum clock frequency. The aim is to complete a task as fast as possible and to put the core into a deep sleep state. On the system side, the frequency of the LITTLE cores can be varied between 200 MHz and 1.4 GHz in step of 100 MHz whereas the frequency of the big cores can be varied between 200 MHz and 2 GHz in step of 100 MHz. Nevertheless, the frequency of an individual core in either group cannot be managed independently, as all the cores in the same group share the same phase-locked loop which generates the clock frequency of the group. Consequently, the frequency estimated by a DVFS policy for a particular core and the frequency assigned to the same core can be different.

Measurement. Our analysis involves three different configurations for the two groups of processors, four DVFS policies, and two types of stochastic workloads. Altogether we conducted 24 experiments. We run each experiment for 1 hour and employed YOKOGAWA WT210 digital power analysers[3] to measure the power consumption of the Odroid-XU4 board at a rate of approximately 10 Hz. To get statistics pertaining to the CPU utilisation of each core, we used DSTAT[4].

4 Evaluation

In order to carry out reproducible experiments, we generated the workloads based on underlying probability density functions (uniform and exponential). These workloads are then processed in different configurations: LITTLE, big, and big.LITTLE configuration. In the LITTLE configuration we deactivated the A15 cores and transcoded videos only with the LITTLE cores. Likewise, in the big configuration, we deactivated the A7 cores. In the big.LITTLE configuration,

[3] http://tmi.yokogawa.com/

[4] `http://dag.wiee.rs/home-made/dstat/`

all cores are activated. In each experiment, a specific DVFS policy determines at any given time a suitable operation frequency for all cores sharing a single PLL. For each experiment, we specify time intervals (the duration) which is determined by (1) the largest video which should be transcoded and (2) the speed of transcoding this video under the DVFS policy we selected for that particular experiment. Tab. 2 shows the time required to transcode the longest video for the three configurations under the different DVFS policies. This latency determines the task arrival interval during video transcoding. Except for the *power-save* policy, we discovered that the time required to transcode the longest video in a given processor configuration is almost the same for all the scaling policies. At the beginning of each interval, each active core is supplied with a video to transcode. The video is chosen randomly from a pool of videos; hence, a processor may not spend the entire time slot transcoding a video, in which case, it may spend some time idling. The idle time statistics is used by a DVFS policy to estimate the appropriate clock frequency of the core.

Table 2: Transcoding latency (in seconds) to determine task arrival interval.

Configuration	Conservative	On-demand	Performance	Power-save
LITTLE	45	45	45	105
big	30	30	30	90
big.LITTLE	45	45	45	-

Since the task arrival time is approximately equal for all the DVFS policies (excepting the power-save policy) in a specific processor configuration, the number of videos which should be transcoded within a fixed experiment duration is the same as well for all the policies. This enables to compare the power consumption characteristics of the different policies given the same workload statistics. In all our analysis we employ the cumulative (probability) distribution function which expresses the probability that the value of a given quantity ($\mathbf{x}$) is equal to or below a specified value (x): $\mathrm{F}(x) = P\{\mathbf{x} \leq x\} = p$, where $F(x)$ is the cumulative distribution function(CDF), and p is the probability associated with $F(x)$.

CPU Utilisation vs. Power Consumption. The power vs. CPU utilisation of the three processor configurations exhibit distinct features. In order to plot the relationship between the two quantities, we normalised the CPU utilisation by the maximum CPU utilisation. Likewise, the overall power consumption is normalised by the maximum power consumption. This way, both quantities have values ranging from 0 to 1. Fig. 2 displays an example of the CDFs of the normalised power and the normalised CPU utilisation for the LITTLE cores under the *ondemand* DVFS policy. As can be seen, the CDF of the power consumption remains more or less unaffected by the change in the statistics of the CPU utilisation. In terms of the diversity of CPU utilisation (following the change in the statistics of the size of transcoding workload), the conservative and performance policies show responsiveness whereas the other two policies do not appear to be responsive, which indicates that the latter are slow to adapt to change. Fig. 3

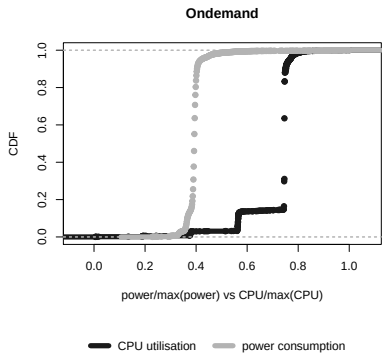

Fig. 2: Normalised CPU-Utilisation vs. Power for the LITTLE cores under *ondemand* DVFS policy for an exponential workload.

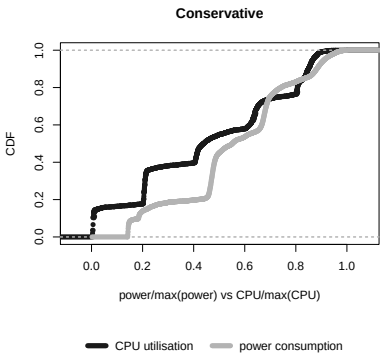

Fig. 3: Normalised CPU-Utilisation vs. Power for the big cores under *conservative* DVFS policy and for an exponential workload.

displays the CDFs of the normalised power and the normalised CPU utilisation under the *conservative* DVFS policy as an example for the big cores. Unlike the case with the LITTLE cores, here there is an almost ideal linear relationship between the two quantities. An exception is observed with the *power-save* policy which is not unexpected, as this policy operates the processors with the lowest frequency all the time. An interesting feature we observed is the pattern of the relationship. The CDF of the power consumption is always below or on the right-side of the CDF of the CPU utilisation which indicates that the probability of consuming more power is always slightly larger than the probability of utilising a corresponding amount of CPU (which we understand as an indication of energy disproportionality). The difference in pattern is bigger for the performance policy; since this policy operates the CPU with the maximum frequency, it is to be expected that it is not the most adaptive policy. Fig. 4 displays an example

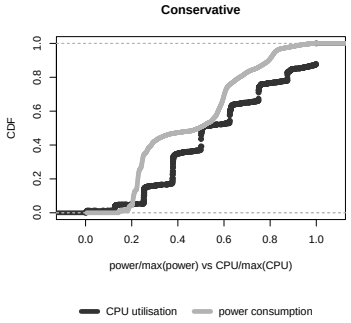

Fig. 4: Normalised CPU-Utilisation vs. Power for the big.LITTLE configuration under *conservative* DVFS policy for an exponential workload.

of the CDFs of the normalised power and the normalised CPU utilisation under *conservative* DVFS policy for the big.LITTLE configuration. We excluded the *power-save* policy for this configuration because the completion time for the

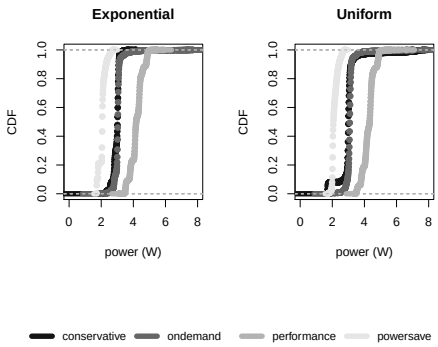

Fig. 5: Comparison of the power consumption of LITTLE cores under different DVFS policies.

transcoding task becomes too big. The rest demonstrate the features that can be inherited from both types of cores when the system operates in a hybrid mode. The almost linear relationship between the two quantities is visibly inherited from the big cores, but the pattern of the relationship is taken from the LITTLE cores, for the CDF of the normalised CPU utilisation is always below or on the right-side of the CDF of the normalised power (it is the other way round for the big cores). This indicates that power can be saved in this mode, but it must be recalled from Tab. 2 that the performance (in terms of transcoding latency) of this configuration is comparable to the performance of the LITTLE cores.

Power Consumption vs. Workload. Figs. 5-7 display the CDFs of the power consumption of the three configurations under the four DVFS policies for the two different types of workloads. In Fig. 5 the usefulness of DVFS is apparent, because the two graphs in the middle show the adaptiveness of the *conservative* and *on-demand* policies to a change in the workload statistics whereas the two extremes show the cost of operating the processors at fixed frequencies. While the reduced power consumption of *power-save* is a consequence of a lower throughput (more time is needed to complete transcoding all the video files), the higher power consumption of the *performance* policy is unjustifiable, for there is no improved throughput. Both the *conservative* and *on-demand* policies completed transcoding all the video files within the same time period as the *performance* policy. For the big configuration, all except the *power-save* policy performed comparatively the same for both types of workloads, showing a broader range of values (from 1.8 W to 12 W) which suggests that they were comparatively adaptive to the fluctuation in the workloads. For both types of workloads the *conservative* policy was the most efficient policy (the gradual scaling up of clock frequencies in response to a perceived workload). The CDFs of the power consumption of the big.LITTLE configuration (as a trade-off between increased transcoding latency for reduced power consumption) can be seen in Fig. 7. For example, in Fig. 6 the probability that the overall power consumption of the A15 cores is equal to or below 6 W for all workload types and for all DVFS policies is approximately

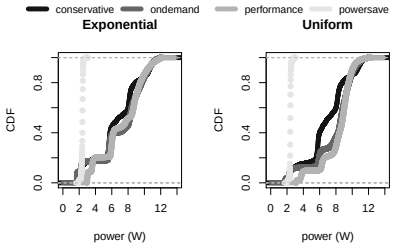

Fig. 6: Comparison of the power consumption of the big cores when they processed different workloads under different DVFS policies.

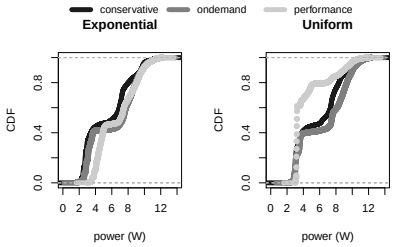

Fig. 7: Comparison of the power consumption of the big.LITTLE configuration when processing different workloads under different DVFS policies.

0.2 whereas for the heterogeneous configuration the figure is approximately equal to or even greater than 0.4. Indeed, for the uniform workload, the *performance* policy in this configuration yields a probability of approximately 0.8.

Power Consumption vs. Processor Configuration. Figs. 8 and 9 compare the CDFs of the power consumption of the three configurations for the two types of workloads using *ondemand* and *conservative* DVFS policies as example. Both figures consistently place the big.LITTLE configuration's power consumption between the LITTLE and the big configuration, regardless of the type of DVFS policy which manages the runtime power consumption of the processors. Moreover, both figures indicate that a wide range of dynamic power can be achieved in the big and big.LITTLE configurations by DVFS whereas this is not the case with the LITTLE configuration. This characteristic is more visible with the exponential workload.

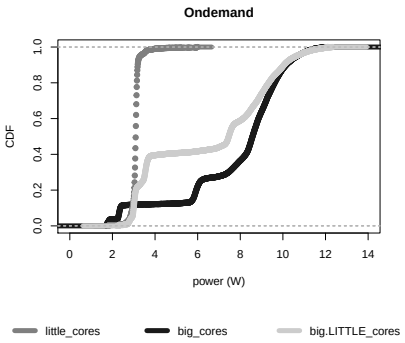

Fig. 8: An example of comparison of the power consumption characteristics of the three processor configurations (LITTLE, big, big.LITTLE) when they process a uniform workload under *ondemand* DVFS policy.

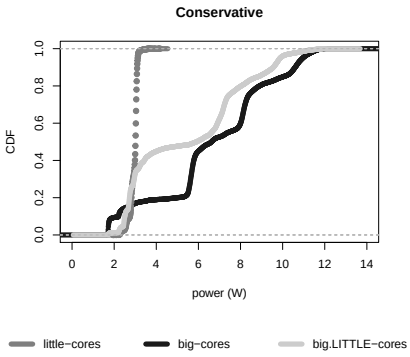

Fig. 9: An example of comparison of the power consumption characteristics of the three processor configurations (LITTLE, big, big.LITTLE) when they process an exponential workload under *conservative* DVFS policy.

Energy. Except for the *power-save* policy, the number of videos which can be transcoded in a specific amount of time for a specific processor configuration is fixed. Since the workload statistics is the same for all the DVFS policies for a

specific configuration, it is possible to make an objective comparison between the energy consumption of the different policies for a given configuration. Tab. 3 and 4 display the energy consumption (energy as the integration of power consumed with respect to time) of the different configurations. Interestingly, the *conservative* policy, which gradually scales the clock frequency of the processors as a function of the perceived change in the workload, performed best for all the configuration yielding the minimum amount of energy consumption. Exception to this is the big.LITTLE configuration where the *performance* policy produces the minimum energy consumption.

Table 3: Cores energy consumption (in Watt-hour) under uniform workload with different DVFS

Configuration	Conservative	Ondemand	Performance	Powersave
LITTLE	3.01	3.11	4.27	2.15
big	6.89	7.78	8.23	2.41
big.LITTLE	5.92	6.55	4.56	-

Table 4: Cores energy consumption (in Watt-hour) under exponential workload with different DVFS

Configuration	Conservative	Ondemand	Performance	Powersave
LITTLE	2.95	3.05	4.21	2.07
big	6.69	7.17	7.31	2.40
big.LITTLE	5.54	6.23	6.63	-

5 Conclusion

In this paper we experimentally investigated the energy-proportionality and the variation in the dynamic power of a heterogeneous processor architecture consisting of two quad-core CPUs. The CPUs are different in capacity as well as in the range of operation frequencies they support. We generated two stochastic video transcoding workloads (uniformly and exponentially distributed) and implemented four different types of dynamic voltage and frequency scaling policies. The processor architecture can be configured as LITTLE, big, and big.LITTLE. In the LITTLE configuration only the smaller and the slower of the two quad-core processors is active; in the big, only the bigger and the faster quad-core processor is active, and in the big.LITTLE all the processors can be active. The experiment results show that the three configurations have distinct energy-utility characteristics. The LITTLE configuration has the minimum average power consumption, but the range of its dynamic power is narrow and does not mirror the variation in the workload. The big configuration, on the other hand, exhibits a wide variation in its dynamic power and mirrors the variation in the workload. The dynamic power of the big.LITTLE configuration mirrors the dynamic power characteristic of the big configuration, but the magnitude of variation is smaller than the big configuration. When it comes to dynamic scaling of frequency and voltage, the *conservative* policy, which gradually increases and decreases the clock frequency

of a processor in response to a perceived change in the workload, consistently produces the best performance in the big configuration, regardless of the workload type. The same scaling policy was the best policy for the exponential workload in the big.LITTLE configuration. For the LITTLE configuration, the *conservative* and the *on-demand* policies have comparable performance whereas in the other two no conspicuous saving or consumption of power can be observed without a corresponding sacrifice or gain in the job completion time. In general, it can be concluded that the implementation of a dynamic power management policy makes sense if the processor architecture has a wide range of dynamic power and the workload statistic is known. The latter can be estimated by taking samples at runtime. Hence, knowledge of this statistics can be used by a scheduler to determine both the configuration of the computing platform and the suitable DVFS policy.

References

1. S. Maiti, N. Kapadia, and S. Pasricha, "Process Variation Aware Dynamic Power Management in Multicore Systems with Extended Range Voltage/Frequency Scaling," in *MWSCAS*, pp. 1–4, 2015.
2. V. Hanumaiah and S. Vrudhula, "Energy-efficient operation of multicore processors by DVFS, task migration, and active cooling," in *IEEE Transactions on Computers*, vol. 63, pp. 349–360, 2014.
3. C. Möbius, W. Dargie, and A. Schill, "Power consumption estimation models for processors, virtual machines, and servers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1600–1614, 2014.
4. W. Dargie, "A stochastic model for estimating the power consumption of a processor," *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1311–1322, 2015.
5. W. Dargie, "Analysis of the power consumption of a multimedia server under different dvfs policies," in *CLOUD*, pp. 779–785, IEEE, 2012.
6. J. Sahuquillo, H. Hassan, S. Petit, J. L. March, and J. Duato, "A dynamic execution time estimation model to save energy in heterogeneous multicores running periodic tasks," *Future Generation Computer Systems*, vol. 56, pp. 211–219, 2016.
7. V. Petrucci, O. Loques, D. Mossé, R. Melhem, N. A. Gazala, and S. Gobriel, "Energy-efficient thread assignment optimization for heterogeneous multicore systems," *ACM Trans. on Embeded Computing Systems*, vol. 14, no. 1, p. 15, 2015.
8. G. Liu, J. Park, and D. Marculescu, "Dynamic thread mapping for high-performance, power-efficient heterogeneous many-core systems," in *ICCD*, pp. 54–61, IEEE, 2013.
9. M. Pricopi, T. S. Muthukaruppan, V. Venkataramani, T. Mitra, and S. Vishin, "Power-performance modeling on asymmetric multi-cores," in *CASES*, sep 2013.
10. G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras, "Predictive dynamic thermal and power management for heterogeneous mobile platforms," in *DATE*, 2015.
11. A. Prakash, H. Amrouch, M. Shafique, T. Mitra, and J. Henkel, "Improving mobile gaming performance through cooperative CPU-GPU thermal management," in *DAC*, 2016.
12. V. Pallipadi and A. Starikovisky, "The ondemand governer," in *Proceedings of the Linux Symposium (volume two)*, 2006.
13. A. Brihi and W. Dargie, "Dynamic Voltage and Frequency Scaling in Multimedia Servers," in *AINA*, pp. 374–380, 2013.