Efficient Programming for Multicore Processor Heterogeneity: OpenMP versus OmpSs

Anastasiia Butko, Florent Bruguier, Abdoulaye Gamatié, Gilles Sassatelli

Efficient Programming for Multicore Processor Heterogeneity: OpenMP versus OmpSs

Anastasiia Butko, Florent Bruguier, Abdoulaye Gamatié, and Gilles Sassatelli
LIRMM (CNRS and University of Montpellier), Montpellier, France
Email: {firstname.lastname}@lirmm.fr

Abstract. ARM single-ISA heterogeneous multicore processors combine high-performance big cores with power-efficient small cores. They aim at achieving a suitable balance between performance and energy. However, a main challenge is to program such architectures so as to efficiently exploit their features. In this paper, we study the impact on performance and energy trade-offs of single-ISA architecture according to OpenMP 3.0 and the OmpSs programming models. We consider different symmetric/asymmetric architecture configurations in terms of core frequency and core count between big and LITTLE clusters. Experiments are conducted on both a real Samsung Exynos 5 Octa system-on-chip and the gem5/McPAT simulation frameworks. Results show that OmpSs implementations are more sensitive to loop scheduling parameters than OpenMP 3.0. In most cases, best OmpSs configurations significantly outperform OpenMP ones. While cluster frequency asymmetry provides uninteresting results, asymmetric cluster configuration with single high-performance core and multiple low-power cores provides better performance/energy trade-offs in many cases.

1 Introduction

Energy-efficiency is one major challenge to be addressed for next generation high-performance computing systems, especially for those foreseen for Exascale computing. Among the ongoing investigations conducted by the concerned research community, we can mention [13] that suggest the massive usage of low-power embedded cores to build energy-efficient supercomputers. Heterogeneous platforms have become a promising direction to make architectures providing the required compromise in terms of performance and power dissipation.

Heterogeneous multicore architectures usually consist of different cores which differ in each other from their instruction set architectures (ISAs), their execution paradigms, e.g. in-order and out-of-order, and other fundamental characteristics. Among them, single-ISA heterogeneous multicore [11] has an important advantage due to the fact that all processor cores execute the same ISA, which allows the whole system to benefit from the symmetric multiprocessor (SMP) execution model with a single operating system (OS) running on top of it. Therefore the workload can be efficiently distributed among high-performance and low-power cores to achieve better performance and energy. However, such execution flexibility requires complex support from the programming environment.

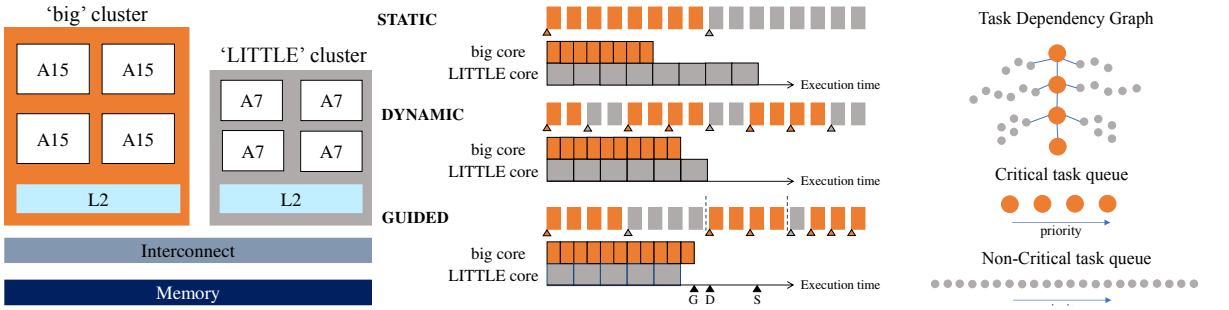
The objective of this paper is to study the impact of OpenMP and OmpSs programming approaches. We evaluate the sensitivity of OpenMP with *dynamic* scheduler and OmpSs with *CATS* depending on the one hand of granularity, i.e. chunk size or block size and on the other hand on cluster frequency asymmetry. Experiments are conducted using real Exynos 5 Octa (5422) SoC. Finally, an exploratory work is proposed studying asymmetry in terms of number of cores per cluster using simulation models in gem5 and McPAT frameworks accurately configured using the board. Experimental software environment including big.LITTLE simulation models as well as disk image with integrated ompss runtime and benchmarks are aimed to be freely available online [1].

The rest of the paper is organized as follows: Section 2 presents the background on the relevant topics such as ARM big.LITTLE technology and OpenMP/ OmpSs runtime scheduling policies. Section 3 describes our exploration methodology including the description of Odroid XU3 board setup and gem5/McPAT full-system simulation. Evaluation of the cluster frequency asymmetry and core configuration asymmetry are presented in Section 4. Finally Section 5 gives concluding remarks and perspectives.

2 Motivation and Background

Heterogeneous ARM big.LITTLE Architecture. The ARM big.LITTLE is a single-ISA heterogeneous multicore system made of two sets of cores: a low power cluster referred to as the "LITTLE" cluster, and a higher performance called "big" cluster. One existing implementation of such system is the Samsung Exynos 5 Octa shown in Figure 1 (a), combining four ARMv7 Cortex-A7 cores and four Cortex-A15 cores. Clusters operate at

[1] http://www.lirmm.fr/ADAC



(a) *ARM big.LITTLE SoC* (b) *OpenMP scheduling policies* (c) *OmpSs task-based programming*

Fig. 1: Background.

independent frequencies, from 200MHz up to 1.4GHz for the LITTLE one and up to 2GHz for the big one. Each core has its private L1 instruction (I) and data (D) caches. And each of both clusters has its own L2 cache shared among all cluster cores. The L2 sizes differ, the Cortex-A7 cluster has a smaller 512kB L2 cache whereas the Cortex-A15 has 2MB L2 cache. L2 caches are connected to the DRAM memory via the 64-bit Cache Coherent Interconnect (CCI) 400. The SoC incorporates its own system memory in the form of 2GB LPDDR3 RAM. It runs at 933MHz frequency and achieves 14.9GB/s memory bandwidth with 2x32-bit bus.

OpenMP and OmpSs Programming Models. *OpenMP* [3] is a popular shared memory parallel programming interface. OpenMP v3 features thread-based fork-join task allocation model. It consists of a set of compiler directives, library routines and environment variables for developing parallel applications. The OpenMP loop scheduling allows determining the way in which iterations of a parallel loop are assigned to threads. Iterations can be assigned in *chunks*, e.g. the number of contiguous iterations. Three general loop scheduling types are available: static, dynamic, and guided as shown in Figure 1 (b).

OmpSs is a task-based programming model [9], which supports asynchronous parallelism and heterogeneity using task directives and dependency tracking mechanisms. OmpSs environment is built on top of Mercurium compiler, which translates the OmpSs annotation clauses to source code and Nanos++ runtime system that manages task execution. Nanos++ supports several task scheduling policies, which define execution order and resource allocation for ready-to-execute tasks, i.e. tasks whose dependencies have been satisfied. The Criticality-Aware Task Scheduler (CATS) is of particular interest, since it targets single-ISA heterogeneous architectures, such as ARM big.LITTLE [7]. The scheduler dynamically detects the longest path of the task dependency graph using bottom-level longest-past priorities as shown in Figure 1 (c). The tasks, which belong to the longest path, are determined as critical. There are two queues for ready tasks: (i) critical task queue that is intended to big cores and (ii) non-critical task queue that is intended to LITTLE cores. The percentage of critical tasks depends on the application nature and granularity. Authors in [7] demonstrated a consistent performance improvement of CATS over the default scheduling policies, which reaches up to 30%.

Motivational example. The following example provides some insights in performance variation within thread-based OpenMP and task-based OmpSs programming models. We consider three parallel implementations of *Cholesky Factorization* problem, i.e. Pthread, OpenMP and OmpSs. Figure 2 shows execution time while running implementations on real Samsung Exynos 5 Octa (5422) processor.

Pthread and OpenMP with static scheduling provide single value. While OpenMP with dynamic or guided scheduling and OmpSs with CATS show execution variation depending on the chosen application granularity, i.e. chunk size or block size. OpenMP implementation runs vary significantly depending on the chunk size within units and tens of seconds. However, OmpSs implementation shows much higher sensitivity to block size configuration and provides execution time variation of three orders of magnitude. Thereby, detailed analysis of implementation behavior together with accurate selection of suitable runtime parameters is crucial for efficient use of complex programming solution such as OmpSs. Moreover, the degree of asymmetry defined by cluster frequency or unequal number of cores per cluster also affects the performance and energy trade-offs. That makes complex software/hardware co-design strongly required.

Related Work. There are a large number of works focused on efficient schedulers for single-ISA heterogeneous multicore systems. Yu et al. in [17] evaluate ARM big.LITTLE power-aware task scheduling, via power

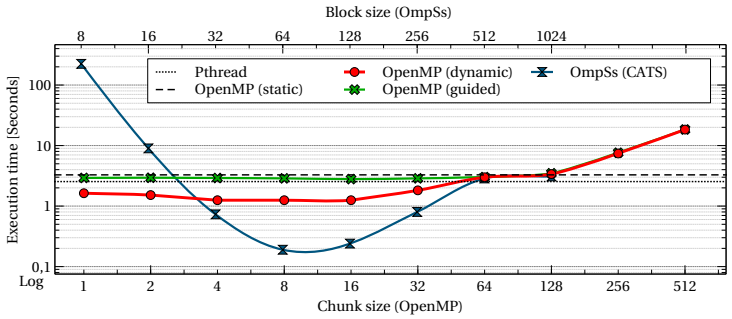


Fig. 2: Motivational example.

saving techniques such as dynamic voltage and frequency scaling (DVFS) and dynamic hot plug. Tan et al. in [14] implement a computation approximation-aware scheduling framework in order to minimize energy consumption and maximize quality of service, while preserving performance and thermal design power constraints. In [12], authors propose a hierarchical power management framework for asymmetric multicores, in particular for ARM big.LITTLE architecture, in order to minimize energy consumption within the thermal design power constraint. Topcuoglu et al. in [15] propose two scheduling algorithms called the Heterogeneous Earliest-Finish-Time (HEFT) and the Critical-Path-on-a-Processor (CPOP). These algorithms aim to meet high performance and fast scheduling time at the same time. The presented results show that the proposed algorithms significantly outperformed the other algorithms. In [7], authors describe and evaluate the OmpSs criticality-aware dynamic task scheduling. The evaluation results illustrate that CATS surpasses HEFT and the earlier proposed breadth-first OmpSs scheduler by up to 2.7x.

Here, we evaluate the impact of cluster frequency asymmetry and asymmetry in terms of cluster multicore architecture while running OpenMP and OmpSs workloads. Our work advances state-of-the-art by combining real board experiments with cycle-approximate full-system simulations thereby enriching the explored design space with not-existing architectural configurations.

3 Exploration Methodology

The first set of experiments aims to study the impact of block size and chunk size on the performance. cluster frequency scaling. Fixing cluster frequency in one of three chosen constant values, i.e. *Low*, *Medium* or *High*, we achieve different levels of heterogeneity. Thus, setting big cluster in High mode and LITTLE cluster in Low mode, we enhance system asymmetry. These experiments are conducted using Odroid XU3 development board, which contains Samsung Exynos Octa (5422) processor.

The second set of experiments focus on the impact of asymmetry in terms of number of different cores instead of performance difference. The base architecture integrates equal number of big and LITTLE cores. We change the given equality by combining one big and seven LITTLE cores and seven big and one LITTLE. These experiments are performed using gem5 and McPAT simulation frameworks.

Platform Experimental Setup. In the first experimental scenario performed on the Odroid XU3 board, we measure workloads execution time and power consumption. The power consumption is measured by means of querying internal sensors. Results are calculated over five consecutive runs.

Denoted experiments require the following environmental setup: (1) individual frequency fixing per cluster, (2) definite thread assignment for OpenMP execution and (3) Nanos++/Mercurium runtime support for OmpSs execution.

(1) The Odroid XU3 board runs Ubuntu 14.04 OS on Linux kernel LTS 3.10. For a better power saving, the Linux kernel offers a set of CPU frequency scaling features. The general frequency scaling policy for CPU is defined by the *scaling governor* thanks to a dedicated power scheme [8]. For our experiments, we fix the cluster frequency by means of the *performance* governor thereby disabling the DVFS feature. We consider three frequency levels: *Low*, *Medium* and *High*. Due to the fact that the LITTLE and the big clusters have different clock limits, the chosen frequency sets are 200MHz, 800MHz and 1.4GHz for the LITTLE cluster and 200MHz, 800MHz and 2GHz for the big cluster. By combining big and LITTLE cluster frequencies, we define five system operating modes:

Table 1: OpenMP 3.0 and OmpSs workloads implementations.

Workload	*blackscholes*	*fluidanimate*	*freqmine*	*cholesky*	*LUD*
OpenMP (Chunk size)	{1 .. 1024}	{1 .. 256}	{1 .. 4096}	{1 .. 512}	{1 .. 256}
OmpSs (Block size)	{256 .. 16384}	default	default	{1 .. 512}	{8 .. 512}
Problem size	Large (64K)	Medium (100K)	Small (250K)	1024x1024	512x512

- big low/LITTLE Low (l/L)
- big low/LITTLE High (l/H)
- big high/LITTLE Low (h/L)
- big high/LITTLE High (h/H)
- big medium/LITTLE Medium (m/M)

(2) To ensure correct thread execution [6], master thread is assigned to the most performance-efficient core, i.e. Cortex-A15, by means of the `GOMP_CPU_AFFINITY` variable.

(3) We follow the provided guide of Nanos++ usage [5] to install and run OmpSs workloads. To set CATS scheduling policy and define the performance-efficient cores, the `NX_SCHEDULE` and `NX_HP_FROM/_TO` environment variables are used.

Full-System Simulation. gem5 is an event-driven cycle-approximate simulator [2]. It has a modular structure that enables flexible configuration of various multicore architecture components. gem5 supports several simulation modes, which differ in their speed and accuracy. *Full-System (FS)* mode is able to run an unmodified operating system, as if this was running on the real hardware. gem5 further produces statistical information enabling to estimate power consumption and footprint area with the Multicore Power, Area, and Timing (McPAT) modeling framework [10]. We implemented performance and power simulation models of ARM big.LITTLE multicore architecture. Our recent study evaluates the accuracy of the proposed models against the real Samsung Exynos Octa (5422) SoC [6]. On an average, the gem5 model predicts performance with less than 20% error. The average error percentage of total power is around 12%.

We consider three architecture configuration groups:

- Symmetric HMP that represents the baseline ARM big.LITTLE processor with four Cortex-A7 and four Cortex-A15 cores (4A7/4A15);
- SMP that includes eight Cortex-A7 (8A7) or eight Cortex-A15 (8A15) cores;
- Asymmetric HMP that introduces alternative big.LITTLE processors with single Cortex-A7 and seven Cortex-A15 cores (1A7/7A15) or seven Cortex-A7 and single Cortex-A15 cores (7A7/1A15).

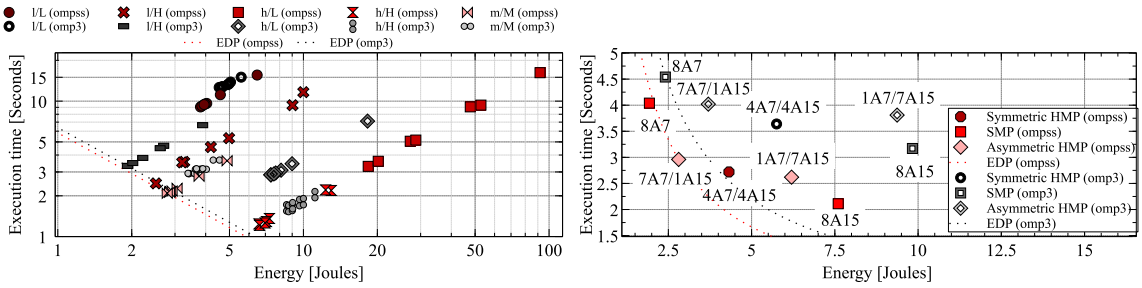
Considering asymmetric HMP and SMP configurations, we adapt the size of L2 cache per cluster according to the number of cores. Namely, we use 128kB per single Cortex-A7 core and 512kB per single Cortex-A15 core. For example, the baseline 4A7/4A15 configuration possesses L2 caches of 512kB and 2MB for LITTLE and big cluster respectively. Thus an alternative asymmetric 1A7/7A15 configuration contains L2 caches of 128kB and 3.5MB for LITTLE and big cluster respectively.

Workloads. Table 1 lists the chosen workload set that contains parallel applications and kernels from different benchmarks, such as PARSEC [1], SPLASH-2 [16] and Rodinia [4]. The set consists of *blackscholes*, *fluidanimate*, *freqmine*, *cholesky*, and *LU decomposition*. For each workload we provide OpenMP and OmpSs execution parameters, i.e. chunk size or block size, and problem size. OmpSs implementations are taken from the BSC Application Repository (BAR) and PARSEC-ompss benchmark suite [5]. Due to the implementation, several workloads, such as *fluidanimate* and *freqmine* are executed with a 'default' block parameter.

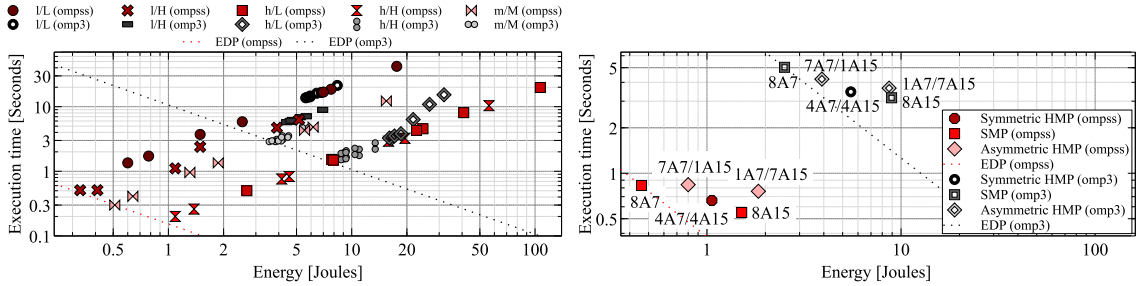
4 Exploration Results and Discussion

We present the results of two previously described experimental scenarios: evaluation of the impact of frequency asymmetry on the Exynos 5 Octa SoC and evaluation of cluster asymmetry in gem5/McPAT simulation frameworks.

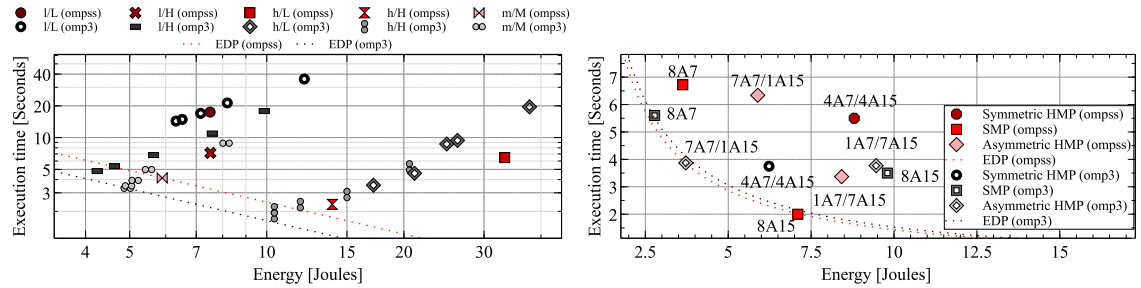
Cluster Frequency. Figure 3 (left side) shows the trade-offs between execution time and energy for five operation modes: l/L, l/H, h/L, h/H and m/M. each graph also includes two iso Energy Delay Product (EDP) curves. On each of these curves, the EDP is constant. We chose to represent the best achieved value when using OpenMP (omp3) and OmpSs (ompss) implementations.



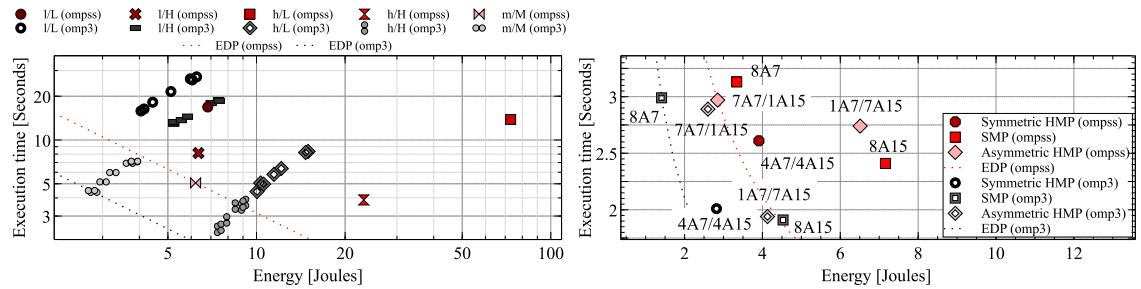
(a) *blackscholes (Exynos 5 Octa board)* (b) *blackscholes (gem5/McPAT simulation)*



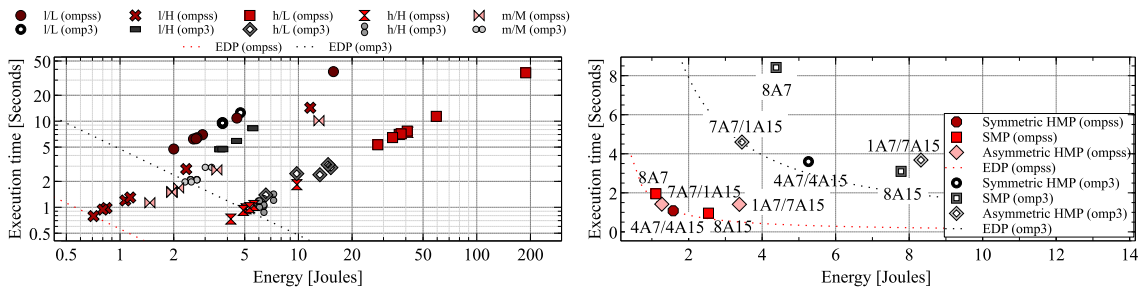
(c) *cholesky (Exynos 5 Octa board)* (d) *cholesky (gem5/McPAT simulation)*



(e) *fluidanimate (Exynos 5 Octa board)* (f) *fluidanimate (gem5/McPAT simulation)*



(g) *freqmine (Exynos 5 Octa board)* (h) *freqmine (gem5/McPAT simulation)*



(i) *lud (Exynos 5 Octa board)* (j) *lud (gem5/McPAT simulation)*

Fig. 3: Frequency Asymmetry Exploration on Exynos 5 Octa SoC: varying block size or chunk size parameters.

The comparison between OpenMP and OmpSs implementations confirms the observation described in Section 2 as a motivational example: carefully selected runtime parameter, i.e. OmpSs block size, results in significantly better performance and energy trade-off when using OmpSs, i.e. *blackscholes*, *cholesky* and *LU decomposition*). In average, OmpSs outperforms OpenMP 27x times in EDP. At the same time, inability to vary workload granularity in *fluidanimate* and *freqmine* makes OpenMP implementation more efficient as shown in Figure 3. In this case, OpenMP implementation outperforms OmpSs 13x times in EDP.

Considering the impact of frequency asymmetry, we observe that almost all workloads provide better results being executed in m/M or l/H modes. This trend is true for both, OmpSs and OpenMP implementations. Moreover, the worst results are usually produced when h/L operation mode is used. Thus, we conclude that operating frequency asymmetry is ineffective for both, performance and energy metrics. While cluster frequency balance, such as in m/M mode, or cluster frequency approximation, i.e. l/H mode, result in significantly better results.

In conclusion, we emphasize the crucial role of both considered features related to programming model parameters and operating frequency per cluster. At some point, the difference in execution time and consumed energy under single workload reaches two orders of magnitude. And although OmpSs implementations are usually more sensitive to block granularity, chunk size parameter for OpenMP also shows high impact. Also, variation of workload granularity within single architecture configuration results in important observation. Similar to demonstrated motivation example in Figure 2, each experimental workload shows the optimal block/chunk size scenario where the best EDP is achieved. However, this trend changes providing optimal result with low-granularity for one workload and high-granularity for another. And in case of *Cholesky Factorization*, the optimal result is achieved when the block/chunk size is balanced between the considered minimum and maximum values. Another observation is that these trends do not change with different frequency-asymmetric configurations. These effects require detailed workloads analysis and runtime execution profiling.

Cluster Architecture. Figure 3 (right side) shows the trade-offs between execution time and energy for three architecture configuration groups: (i) symmetric HMP (4A7/4A15), SMP (8A7 and 8A15) and asymmetric HMP (1A7/7A15 and 7A7/1A15). It also includes the iso-EDP curves that represent the best achieved value when using OpenMP and OmpSs implementations.

Per each workload we select two best points that represent one OpenMP chunk size and one OmpSs block size. Over these experiments, the m/M cluster frequency mode is used. This configuration excludes the impact of cluster frequency asymmetry and also provides best results as shown in Section 4.

Most of the workloads provide two separated groups of points, which correspond to OpenMP and OmpSs executions. For *fluidanimate*, the OpenMP and OmpSs points overlap. OpenMP implementation outperforms OmpSs for *freqmine* workload. The 8A15 SMP configuration running OmpSs implementation provides better EDP than OpenMP for *fluidanimate*. All other workloads provide better EDP using OmpSs programming model.

The combination of Cortex-A7 and Cortex-A15 cores into symmetric HMP architecture can provide suitable balance between the performance of A15 and energy consumption of A7. We observe it on several workloads such as *lud* and *freqmine*. However, in some cases asymmetric HMP configuration, i.e. 7A7/1A15, shows better balance between the performance and energy resulting in better EDP (see *blackscholes*, *freqmine* OpenMP, *lud*). Several workloads benefit more from SMP configurations, such as *fluidanimate* or *cholesky*.

Authors in [7] show that changing the number of blocks and sub-blocks, we also transform the task dependency graph. It becomes wider as the number of blocks increase. Hence, the ratio between critical and non-critical tasks changes. Consequently, the required number of performance-efficient cores in HMP configuration depends on the number of critical tasks. Thus, to identify the best pair of cluster configuration and block size, it is required to consider a set of block sizes. Due to the important simulation time provided by our accurate models, we aim to study these configurations in future.

We distinguish three architecture configurations that provide best EDP results over considered workloads: symmetric HMP (4A7/4A15), SMP (8A7) and asymmetric HMP (7A7/1A15). In most cases, there are more than one points located in the iso EDP curve, which correspond to several best EDP configurations. For example, *blackscholes* OmpSs implementation benefits from being executed in 8A7 and 7A7/1A15 configurations, or *lud* OmpSs implementation benefits from being executed in 7A7/1A15 and 4A7/4A15 configurations, etc. In such cases, the design choice is based on the target metric, i.e. performance-efficiency or energy-efficiency.

5 Conclusion

In this paper, we evaluated the performance and energy trade-offs of heterogeneous big.LITTLE architectures running OpenMP and OmpSs software implementations. Experiments have been conducted using real Samsung

Exynos 5 Octa SoC and calibrated performance and power models in gem5/McPAT simulation environment. We studied the impact of runtime and architectural parameters such as scheduling (i.e. *dynamic* or *CATS*), granularity (i.e. chunk size or block size), cluster frequency asymmetry and cluster architecture. We considered seven parallel workloads implemented in both programming models, i.e. OpenMP and OmpSs, that belong to different benchmarks such as PARSEC, Rodinia and SPLASH-2. We observed that results strongly depend on the application nature. Over seven considered workloads, four of them show important enhancement running OmpSs with CATS implementation. Three OmpSs implementations do not allow vary granularity, i.e. block size, thereby conceding OpenMP implementations, which benefit chunk size parameterization. Mostly, OmpSs implementations show higher sensitivity to application granularity than OpenMP. Concerning architecture configurations, we observe that frequency balance and frequency approximation result in significantly better trade-offs compared to strong frequency asymmetry. We also distinguished asymmetric HMP, which contains single performance-efficient core and multiple low-power cores as one that provides better performance and energy trade-off in most cases. For future work, we aim to evaluate recently released OpenMP 4.5 standard that provides a substantial improvement for heterogeneous multicore architectures. Also, we plan to evaluate the impact of critical task number running OmpSs workloads on different asymmetric HMP configurations.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2015] under the Mont-blanc 2 Project (www.montblanc-project.eu), grant agreement n^{o} 610402.

References

1. Bienia, C.: Benchmarking Modern Multiprocessors. Ph.D. thesis, Princeton University (January 2011)
2. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M.D., Wood, D.A.: The gem5 simulator. SIGARCH Comput. Archit. News
3. Board, O.A.R.: The openmp api specification for parallel programming. http://openmp.org/wp/ (November 2015), `http://openmp.org/wp/`
4. Brodowski, D., Goldeg, N.: Rodinia:accelerating compute-intensive applications with accelerators. http://lava.cs.virginia.edu/Rodinia/ (2014)
5. BSC: Bsc application repository. https://pm.bsc.es (2016)
6. Butko, A., Bruguier, F., Gamatié, A., Sassatelli, G., Novo, D., Torres, L., Robert, M.: Full-system simulation of big.little multicore architecture for perforamance and energy exploration. In: MCSoC 2016. pp. 1–8 (September)
7. Chronaki, K., Rico, A., Badia, R.M., Ayguad, E., Labarta, J., Valero, M.: Criticality-aware dynamic task scheduling for heterogeneous architectures. In: Bhuyan, L.N., Chong, F., Sarkar, V. (eds.) ICS. pp. 329–338. ACM (2015)
8. Dominik Brodowski, N.G.: CPU frequency and voltage scaling code in the Linux(TM) kernel. https://www.kernel.org/doc/Documentation/cpu-freq/ (2016)
9. Duran, A., Ayguad, E., Badia, R.M., Labarta, J., Martinell, L., Martorell, X., Planas, J.: Ompss: a proposal for programming heterogeneous multi-core architectures. Parallel Processing Letters 21(2), 173–193 (2011)
10. Hewlett-Packard: Mcpat. http://www.hpl.hp.com/research/mcpat/ (2008)
11. Kumar, R., Tullsen, D.M., Ranganathan, P., Jouppi, N.P., Farkas, K.I.: Single-isa heterogeneous multi-core architectures for multithreaded workload performance. ISCA '04, IEEE Computer Society, Washington, DC, USA
12. Muthukaruppan, T., Pricopi, M., Venkataramani, V., Mitra, T., Vishin, S.: Hierarchical power management for asymmetric multi-core in dark silicon era. In: DAC 2013 (May 2013)
13. Rajovic, N., Carpenter, P.M., Gelado, I., Puzovic, N., Ramirez, A., Valero, M.: Supercomputing with commodity cpus: Are mobile socs ready for hpc? SC '13, ACM
14. Tan, C., Muthukaruppan, T., Mitra, T., Ju, L.: Approximation-aware scheduling on heterogeneous multi-core architectures. In: Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific. pp. 618–623 (Jan 2015)
15. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems 13(3), 260–274 (Mar 2002)
16. Woo, S.C., Ohara, M., Torrie, E., Singh, J.P., Gupta, A.: The splash-2 programs: characterization and methodological considerations. In: Computer Architecture, 1995. Proceedings., 22nd Annual International Symposium on (June 1995)
17. Yu, K., Han, D., Youn, C., Hwang, S., Lee, J.: Power-aware task scheduling for big.little mobile processor. In: SoC Design Conference (ISOCC), 2013 International. pp. 208–212 (Nov 2013)