Evaluating Scientific Workflow Execution on an Asymmetric Multicore Processor

Ilia Pietri[1(✉)], Sicong Zhuang[2,3], Marc Casas[2,3], Miquel Moretó[2,3], and Rizos Sakellariou[4]

[1] Department of Informatics and Telecommunications, University of Athens, Athens, Greece
ipietri@di.uoa.gr
[2] Barcelona Supercomputing Center (BSC), Barcelona, Spain
[3] Universitat Politecnica de Catalunya (UPC), Barcelona, Spain
[4] School of Computer Science, University of Manchester, Manchester, UK

Abstract. Asymmetric multicore architectures that integrate different types of cores are emerging as a potential solution for good performance and power efficiency. Although scheduling can be improved by utilizing an appropriate set of cores for the execution of the different jobs, determining frequency configurations is also crucial to achieve both good performance and energy efficiency. This challenge may be more profound with scientific workflow applications that consist of jobs with data dependency constraints. The paper focuses on deploying and evaluating the Montage scientific workflow on an asymmetric multicore platform with the aim to explore CPU frequency configurations with different tradeoffs between execution time and energy efficiency. The proposed approach provides good estimates of workflow execution time and energy consumption for different frequency configurations with an average error of less than 8.63% for time and less than 9.69% for energy compared to actual values.

1 Introduction

Complex computational problems in many scientific fields, such as astronomy and physics, may consist of multiple computational steps (jobs) with data dependencies between them. For example, the output data of a program can be used as input from other programs. Scientific workflows [4] are commonly used to describe the computational jobs (tasks) and dependencies between them, separating the application development and execution. In this way, scientists can orchestrate the application components and provide a high level representation of the application independently of the particulars of the execution environment [5]. High performance computing (HPC) systems, including clusters, grids and clouds, have been widely used for the execution of workflow applications in order to improve application performance, by allocating a large number of resources to execute independent tasks (i.e., tasks without data dependencies

© Springer International Publishing AG, part of Springer Nature 2018
D. B. Heras and L. Bougé (Eds.): Euro-Par 2017 Workshops, LNCS 10659, pp. 439–451, 2018.
https://doi.org/10.1007/978-3-319-75178-8_36

between them) in parallel. However, optimizing the execution schedule of scientific workflows can be challenging, as the execution of a task can only start after the execution of its predecessors and data transfer have finished. This may result in idle slots between the execution of workflow tasks and wastage of resources.

When resources are heterogeneous, scheduling a workflow becomes particularly challenging, as many different combinations of the heterogeneous resources may be chosen. For example, the ARM big.LITTLE architecture is composed of fast and slow cores, which can additionally operate at different CPU frequencies. Although power consumption decreases for resources running at a lower computational speed (i.e. operating CPU frequency), overall energy consumption may increase. This is because computational speed may affect task execution time differently, depending on a task's characteristics. For example, the execution time of I/O bound tasks is not greatly affected from the reduction in CPU frequency, while the execution of CPU bound tasks may be greatly affected. As a result, gaps in the schedule due to idle time between the execution of the workflow's tasks may increase when resources operate at a lower computational speed. This may lead to significant idle energy, the energy spent while resources remain idle. Minimizing idle time, while balancing execution time and energy consumption, requires adjusted configurations combining fast and slow cores running at appropriate frequencies.

This paper carries out a set of real experiments to investigate the performance of a widely used scientific workflow application, Montage [9], using different CPU frequencies and different types of cores of an asymmetric multicore processor architecture. Energy consumption and task runtime models for the platform and for each type of core are proposed and validated using real measurements. Using these models, estimations of overall workflow execution time and energy consumption are obtained and compared with real measurements. To the best of our knowledge, this is the first paper evaluating the performance of a scientific workflow application on an ARM big.LITTLE platform.

In the rest of the paper, Sect. 2 gives background information on the architecture, the application model and the problem. Section 3 describes the models used to estimate execution time and energy required for the workflow execution under different frequency configurations. Section 4 evaluates the models on a real system. Section 5 concludes the paper.

2 Background

Architecture: The ARM big.LITTLE architecture is a system-on-chip technology for heterogeneous processing that uses two types of processors with different power and performance characteristics; ARM Cortex-A15 processors (Big out-of-order processors) for high performance processing and ARM Cortex-A7 processors (Little in-order processors) for power efficient processing. In the architecture used in this paper, each processor type contains four cores. The processors are coherently connected so that they can transfer information to each other. Also, the system provides frequency scaling capabilities, which allow to set CPU frequency individually for each core.

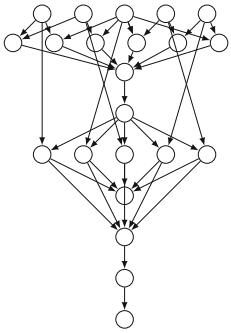


Fig. 1. Structure of a Montage workflow with 22 tasks.

Application Model: The paper assumes that a scientific workflow is modelled as a Directed Acyclic Graph (DAG) with the nodes being the tasks and the edges the data dependencies between them. The Pegasus Project [5] provides tools to generate abstract workflows: these are described in a form that includes information about the arguments required to run each task, their input and output files and data dependencies between them. These high-level abstract workflow descriptions are provided in DAX (directed acyclic graph in XML) files, which make use of a specific XML syntax for expressing the tasks, their arguments, files and dependencies between them. This information can be used to deploy and execute workflow applications on HPC systems in a way that data dependency constraints are respected. The scientific workflow application used in the paper is Montage, which is a real astronomy application that generates image mosaics of the sky [9]. A Montage workflow consists of collections of tasks (job classes) with different characteristics, such as task execution time and CPU utilization. Montage can be characterized as an I/O intensive application where most of the tasks have low CPU utilization and short runtime (in the order of seconds) as they mainly spend their execution time on I/O operations to read and write files. The tasks can be grouped into nine levels, each level corresponding to a different class of tasks. Figure 1 shows an example of a small Montage workflow with 22 tasks. In larger versions of Montage, the number of tasks of the first, second and fifth levels (counting from top to bottom) would increase further.

Problem Description: The problem of task scheduling onto heterogeneous HPC systems has been extensively studied [2,10–12,14,15], with several works focusing on multicore processors [1,3,8,10,16]. Some of the algorithms focus on optimizing application performance and execution time [10,12], while other works also consider energy and power optimization [1,8]. As heterogeneous multicore processing platforms integrating different types of processing cores are now used as a promising solution towards achieving different performance and power goals, there has been research on policies which aim to determine which types of cores are more appropriate for the scheduling of the applications or their parts [3,10,13]. For example, power-efficient cores may be used for the execution

of memory-bound or non-critical jobs while fast cores may be more suitable for CPU-bound or critical jobs [3,13].

This scheduling problem becomes more complex, as one has to select an appropriate CPU frequency for each core. Clearly, there is a trade-off between energy and performance for different configurations. For example, using fast cores may result in small execution time but increased energy consumption. By lowering the CPU frequency of the cores, lower energy but longer execution time may be achieved. Solutions with even lower energy but significantly increased execution time may also be achieved using slow but power-efficient cores. If we consider all different configurations it is expected that some of them will result in sub-optimal solutions; these are solutions which are dominated by other solutions lying on a Pareto-front of the energy-performance trade-off space. Hence, the aim of the paper is to suggest approaches to explore the space of the CPU frequency combinations for the heterogeneous types of cores in order to find solutions with good performance and energy efficiency close to the pareto front. In contrast to heuristics-based related work [6], this paper proposes energy and execution time models to obtain estimations for a wide range of configurations, which are based on metrics monitored through real measurements from a small set of configurations. For example, runtime and power characteristics may be available from historical data and can be sufficient to provide estimations for frequencies between the extreme cases.

3 Modelling Execution Time and Energy Consumption

The idea in the paper is that given a predefined assignment policy for the mapping of the tasks to the cores, the execution of the workflow under different configurations can be modelled using task execution time and energy consumption estimations. The assignment policy and data dependency constraints between tasks specify the execution order of the tasks on the cores. Hence, task runtime estimations for each configuration can be used to specify the time slot required for the execution of each task. Also, power models can be used to estimate under different configurations the energy consumed when cores are idle or busy executing tasks. Based on such estimations, overall execution time and energy required to run the workflow is estimated.

3.1 Estimation of Execution Time

While frequency scaling may impact job performance, the increase in execution time is not proportional to the decrease in frequency as non-CPU activity, such as memory access, is not sensitive to frequency changes. Hence, different jobs may exhibit different performance slowdown depending on their CPU-boundedness. Assuming that we know the execution time of each task when the core it is assigned to operates at the maximum CPU frequency, task runtime estimations for different frequencies can be computed by:

$$taskRuntime_{t,f} = (\beta_t \cdot (\frac{f_{max}}{f} - 1) + 1) \cdot taskRuntime_{t,f_{max}}, \quad (1)$$

where β_t, the CPU boundedness of task t, shows the performance sensitivity to frequency reduction [7]. The parameter β_t can be computed for each task based on measurements at the maximum and minimum operating frequency. In that way, task runtime estimations can be provided for different CPU frequencies (and for each type of core) by measuring task performance at the two extreme cases of operating the core, its maximum and minimum frequency. Then, the start time and finish time of each task in the schedule for a given assignment can be estimated recursively (Eqs. 2 and 3) based on the task runtime estimations. The start time of a task t is estimated as:

$$startTime_{t,f_t} = \begin{cases} \max_{\forall p \in pred_t} (finishTime_{p,f_p} + comCost_{p \to t}), & \text{if } pred_t \neq \emptyset \\ 0, & otherwise \end{cases} \tag{2}$$

where $pred_t$ includes the predecessors of the task at both the DAG and the core assigned. When the task has no predecessors at the DAG or the core, the start time of the task is zero. The communication cost from task p to task t, $comCost_{p \to t}$, is assumed to be zero, as tasks granularities are significantly larger than data transfer costs and there is a good interconnect. The finish time of a task t is estimated as:

$$finishTime_{t,f_t} = startTime_{t,f_t} + taskRuntime_{t,f_t}. \tag{3}$$

Overall workflow execution time for a given schedule is the finish time of the execution of the latest task:

$$makespan = \max_{\forall t \in w}(finishTime_{t,f_t}). \tag{4}$$

3.2 Estimation of Energy Consumption

The energy consumed during the execution of the workflow may vary between different execution schedules depending on the operating frequency of each core. The energy model used in this paper estimates the energy required under different frequency configurations taking into account the dynamic energy required for the execution of the tasks and the static energy of the system. Energy is the product of power and time. As frequency scaling does not affect non-CPU activity, power consumption at CPU (A7/A15 cluster) is modelled for the different frequency configurations, while power in the other system components, such as memory and GPU, is considered to be fixed, as it does not vary significantly between different frequency configurations. Note that CPU power consumption can be measured at the level of cluster and not individually per core. Power consumption at the A7/A15 cluster when running the task at a single core is modelled using a linear power model:

$$P_{f_t,t} = P_{base,t} + P_{dif,t} \cdot (\frac{f_t}{f_{min}} - 1), \tag{5}$$

where $P_{base,t}$ and $P_{dif,t}$ are parameters linearly fitted for each task t based on power measurements for the extreme cases of operating the cluster at minimum and maximum frequency supported. All the cores of the cluster operate at the same frequency level f_t. Power consumption while the cluster is not utilized (idle power) is also modelled for the different frequency configurations using a similar model:

$$P_{idle_{f_t}} = P_{base_{idle}} + P_{dif_{idle}} \cdot (\frac{f_t}{f_{min}} - 1), \tag{6}$$

where $P_{base_{idle}}$ and $P_{dif_{idle}}$ are parameters fitted based on the power measurements at the minimum and maximum frequency supported for each core type (cluster).

Based on the power models above, the dynamic power required to run each task t on a core can be estimated by:

$$P_{dyn_{t,f_t}} = P_{f_t,t} - P_{idle_{f_t}}. \tag{7}$$

Then, overall energy can be computed as:

$$E = \sum_{t \in w} (taskRuntime_{t,f} \cdot P_{dyn_{t,f}}) + P_{fixed} \cdot makespan, \tag{8}$$

where P_{fixed} includes the idle power of the system (A7 and A15 cluster, memory and GPU) when the minimum frequency is set to the cores and the memory power required for the execution of the workflow.

The models described above are validated next and used to estimate overall workflow execution time and energy consumption.

4 Results

4.1 Methodology and Experimental Setup

Experiments are conducted on an ODROID-XU3 board that contains an eight-core Samsung Exynos 5422 processor of ARM big.LITTLE architecture with 2 Gbyte LPDDR3 RAM. The processor chip consists of a Cortex-A15 1.6 GHz quad core CPU and a Cortex-A7 1.4 GHz quad core CPU with a shared 2 MB and 512 KB L2 cache, respectively. Both CPUs, the Cortex-A15 cluster of four fast (or Big) cores and the Cortex-A7 cluster of four slow (or Little) cores, can be used in order to run simultaneously independent tasks of an application. Each core can be set to operate on a different number of CPU frequencies using the `cpufreq` driver. In the experiments, we varied the CPU frequency in the range of 0.8–1.6 GHz with a frequency step of 0.2 GHz for the fast cores and 0.8–1.3 GHz with a frequency step of 0.1 GHz for the slow cores, resulting in a total of five and six operating CPU frequency configurations, respectively.

For performance counter events we used `perf`, a performance monitoring tool for Linux, to collect various information about workflow execution, such as

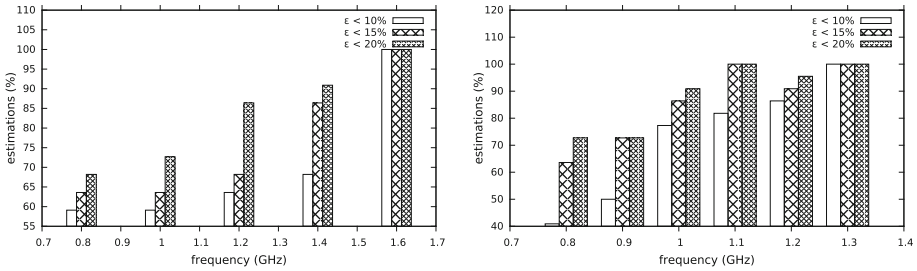
time duration and CPU utilization, and profile the application. Also, in order to estimate the energy consumed by the application at each configuration, power measurements are monitored using an energy daemon that reads power measurements at A7 cluster, A15 cluster, memory and GPU separately provided by INA-231 power sensors on the Hardkernel ODROID system every 0.27 sec. To do so, the sensors are enabled in advance. Regardless of the number of cores utilized, CPU power consumption is measured at the cluster level and per-core power monitoring is not supported. Overall energy consumption incurred during workflow execution (we refer to this as actual energy) is computed as the product of the average power consumption and workflow execution time, each one measured as above.

Two versions of the Montage workflow are used. The first version consists of 22 tasks and corresponds to the DAG shown in Fig. 1; this will be denoted by M_{22}. A second version of Montage with 65 tasks has more tasks at levels 1, 2, and 5 (counting from top to bottom), namely 15, 29, 15; we denote this version by M_{65}. In order to run a workflow on the platform, a script was written to manage the execution of the tasks using a statically predefined mapping policy. The tasks of each level are assigned statically to the CPU cores in a round-robin fashion, so that independent tasks can be executed in parallel; the mapping also specifies at what frequency each core runs. When a core is empty (no task has been assigned to it), the CPU frequency of the core is set to its minimum operating CPU frequency (0.8 GHz). Otherwise, the CPU frequency is set based on the predefined assignment policy. In each experiment, unless otherwise stated, four cores (two Big and two Little cores) from a total of eight available ones (four Big and four Little cores) are used to run the workflow. The script checks every t seconds if the execution of any running tasks finishes in order to start the execution of ready tasks (the successors of the tasks on the cores assuming that dependency constraints are met), adjusting the CPU frequency of the cores accordingly. The value of t is set close to the minimum task execution time so that performance overhead (in terms of both execution time and energy) but also the delay on triggering ready tasks is minimized. To do so, t is set equal to 0.2 and 0.5 secs for the small (22 tasks) and large (65 tasks) workflow, respectively. Also, the script runs at an idle core (a core that is not used for the execution of the workflow tasks) at minimum frequency so that the overhead is minimized.

Finally, we note that, unless otherwise stated, each experiment is repeated ten times and the average value of each metric is used to express the final results.

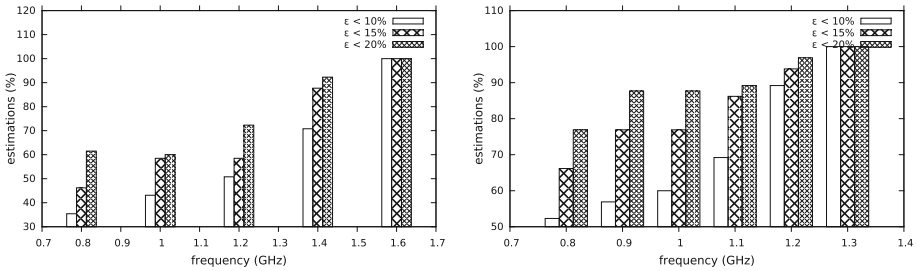
4.2 Validation of Task Runtime and Power Models

In this section, the power and task runtime models from Sect. 3 are validated. Task runtime estimations for homogeneous cores (cores of a single type operating at the same frequency) are computed using Eq. 1. To do so, each Montage workflow is executed using four homogeneous cores, alternatively four Big or four Little cores, for each available CPU frequency (five for Big and six for Little, as mentioned in Sect. 4.1) in order to collect information about the CPU utilization and runtime of the tasks. Parameter β_t is modelled as the average



(a) Big cores (b) Little cores

Fig. 2. Accuracy of task runtime estimation for each of the 22 tasks of M_{22}.



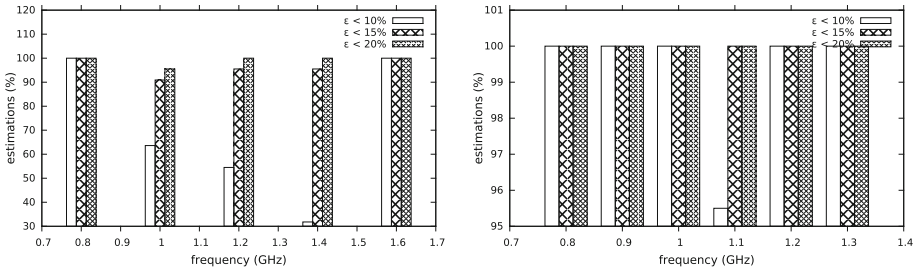
(a) Big cores (b) Little cores

Fig. 3. Accuracy of task runtime estimation for each of the 65 tasks of M_{65}.

CPU utilization for each task at maximum frequency. Also, from the ten different runs for each experiment, the average runtime of each task is computed using the five smallest values so that any outliers (runs with poor performance) are not taken into account. Task runtime estimations are compared with the actual task runtimes at the different frequencies used. Figures 2 and 3 show the percentage of estimations with an error, ϵ, of less than 10, 15 and 20% for Big and Little cores for the small (M_{22}) and large (M_{65}) workflow, respectively. Large % errors are mostly related to small duration tasks, with a small impact on the overall workflow.

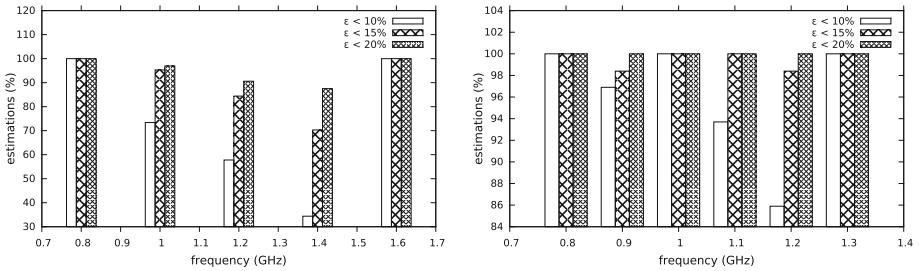
Experiments are also conducted to measure the power consumption of the A7/A15 cluster for different frequency configurations when running each workflow sequentially on a single core, as power consumption cannot be measured independently for each core. The parameters P_{base} and P_{dif} of the power model in Eq. 5 are fitted based on the power measurements at minimum and maximum frequency. Figures 4 and 5 show the percentage of power estimations with an error of less than 10, 15 and 20% at Big/Little cores (A7/A15 cluster) for the small and large workflow, respectively.

Finally, experiments are performed to profile idle power consumption and fit the parameters of the model in Eq. 6. For each experiment the CPU frequency of the Big/Little cores is set to the desired level and the `sleep` function is used for



(a) Big cores (b) Little cores

Fig. 4. Accuracy of power estimation for the M_{22} workflow.



(a) Big cores (b) Little cores

Fig. 5. Accuracy of power estimation for the M_{65} workflow.

10 sec to compute the average idle power consumption at the selected frequency. To do so, the energy daemon is triggered before and after the `sleep(10)` function to monitor the power measurements at the A7/A15 clusters (Big/Little cores). The parameters of the model, $P_{base_{idle}}$ and $P_{dif_{idle}}$, are fitted based on the power measurements at the minimum and maximum frequency for each core type. The results are compared with the actual measurements and the estimation error in idle power is less than 10% for all the frequency configurations used.

4.3 Workflow Energy and Execution Time Estimation

In this set of experiments we use as input parameters task runtime and power estimations when executing the workflow on homogeneous resources, Big or Little cores, in order to estimate overall workflow execution time and energy consumption when executing the workflow onto heterogeneous resources (combining two Big and two Little cores with different frequencies), based on the models in Sect. 4.2. These estimations are then compared with actual measurements. Delays that often happen in real environments, such as system overhead and job submission delays, are also incorporated. As mentioned in Sect. 4.1, the main script checks every t sec if the execution of any running tasks finishes in order to start the execution of ready tasks. This may cause delays in the assignment

Table 1. Overall execution time and energy estimations for the M_{22} workflow.

Configuration	Actual execution time (s)	Estimated execution time (s)	% error in execution time	Actual energy (J)	Estimated energy	% error in energy
L1.3B1.6	28.8	27.48	4.57	37.95	35.47	6.54
L1.3B1.4	29.5	28.56	3.19	35.11	33.01	6.00
L1.3B1.2	31.1	30.18	2.95	34.37	31.89	7.21
L1.3B1.0	33.6	33.97	1.11	32.33	30.88	4.48
L1.3B0.8	39.5	39.66	0.40	33.8	29.99	11.27
L1.2B1.6	29.2	26.63	8.81	38.44	34.27	10.84
L1.2B1.4	30	28.35	5.50	35.16	33.28	5.35
L1.2B1.2	31.9	30.50	4.38	33.98	31.46	7.40
L1.2B1.0	34.2	33.96	0.70	32.54	30.78	5.41
L1.2B0.8	39.7	38.46	3.13	32.16	29.11	9.49
L1.1B1.6	30.1	26.36	12.43	37.93	33.64	11.32
L1.1B1.4	31	28.76	7.23	35.22	32.79	6.89
L1.1B1.2	32.3	30.69	4.99	33.5	31.24	6.77
L1.1B1.0	34.7	34.95	0.72	31.48	30.77	2.25
L1.1B0.8	40.3	39.74	1.38	31.88	29.45	7.62
L1.0B1.6	31.1	27.38	11.96	37.86	33.96	10.29
L1.0B1.4	31.8	28.40	10.68	34.69	31.92	7.98
L1.0B1.2	32.9	31.23	5.07	33.53	31.29	6.69
L1.0B1.0	34.8	34.26	1.56	31.32	29.86	4.67
L1.0B0.8	39.9	39.74	0.40	31.09	29.01	6.68
L0.9B1.6	32.7	27.73	15.19	39.36	33.42	15.09
L0.9B1.4	33.3	28.54	14.29	35.35	31.49	10.92
L0.9B1.2	34.3	31.85	7.14	33.64	31.23	7.19
L0.9B1.0	36	35.79	0.60	31.63	30.54	3.44
L0.9B0.8	39.8	39.87	0.18	30.59	28.67	6.28
L0.8B1.6	34.1	30.07	11.81	38.73	34.46	11.04
L0.8B1.4	34.9	30.48	12.66	35.47	32.26	9.05
L0.8B1.2	36.1	31.98	11.42	33.83	30.81	8.91
L0.8B1.0	36.8	35.64	3.16	31.29	29.50	5.73
L0.8B0.8	40.7	40.93	0.57	30.43	28.75	5.51

of ready tasks. In order to account for such submission delays and the overhead imposed by the script, a random delay between [0,t] is considered at the runtime estimated for each task. Also, an additional delay of about 1 sec is observed for the large workflow between the execution of subsequent tasks which appears to be due to the time required for the script to check the data dependency constraints between the tasks at the end of each time interval. Thus, a delay of

Table 2. Overall execution time and energy estimations for the M_{65} workflow.

Configuration	Actual execution time (s)	Estimated execution time (s)	% error in execution time	Actual energy (J)	Estimated energy	% error in energy
L1.3B1.6	116	100.13	13.68	128.55	112.11	12.79
L1.3B1.4	118.9	100.61	15.38	121.22	100.13	17.40
L1.3B1.2	122.5	104.71	14.52	114.08	95.62	16.19
L1.3B1.0	127.9	108.00	15.56	108.38	91.36	15.71
L1.3B0.8	135.2	113.07	16.37	103.36	89.04	13.86
L1.2B1.6	121.3	101.96	15.95	130.02	110.56	14.96
L1.2B1.4	124	103.14	16.83	118.62	99.50	16.12
L1.2B1.2	128.9	106.34	17.50	116.95	94.87	18.88
L1.2B1.0	132.2	110.93	16.09	107.65	91.14	15.34
L1.2B0.8	137.7	116.00	15.76	103.81	88.75	14.51
L1.1B1.6	108.1	103.77	4.01	110.12	108.56	1.42
L1.1B1.4	110.3	106.34	3.59	102.84	99.12	3.62
L1.1B1.2	113.4	107.45	5.25	96.91	93.47	3.55
L1.1B1.0	118.6	112.14	5.45	98.3	89.50	8.95
L1.1B0.8	123.9	118.31	4.51	93.94	88.22	6.09
L1.0B1.6	111.8	105.36	5.76	115.37	108.49	5.96
L1.0B1.4	115.2	107.34	6.82	107.53	98.31	8.58
L1.0B1.2	117.5	109.74	6.60	104.46	93.33	10.65
L1.0B1.0	121.8	114.33	6.13	98.94	88.90	10.15
L1.0B0.8	129.1	118.80	7.98	94.1	86.98	7.57
L0.9B1.6	116.3	111.81	3.86	116.45	109.73	5.77
L0.9B1.4	119.9	114.48	4.52	109.47	100.44	8.25
L0.9B1.2	123	118.19	3.91	103.04	95.93	6.90
L0.9B1.0	125.6	121.78	3.04	100.14	91.25	8.88
L0.9B0.8	133.6	125.85	5.80	95.14	88.90	6.56
L0.8B1.6	122.1	115.25	5.61	120.05	110.71	7.78
L0.8B1.4	118.26	117.93	0.28	98.35	100.43	2.12
L0.8B1.2	128.4	120.53	6.13	105.06	95.37	9.22
L0.8B1.0	132.6	124.63	6.01	99.18	91.53	7.72
L0.8B0.8	138.7	130.50	5.91	94.55	89.66	5.17

19 sec is added to the estimated workflow execution time for the large workflow. Finally, an extra amount of 0.036 W and 0.05 W for memory power, $P_{mem_{dyn}}$ in Eq. 8, is considered for each configuration, for the small and large workflow respectively, as average memory power did not vary significantly between the different configuration runs.

Tables 1 and 2 compare actual measurements with estimated values for each of the two workflows, M_{22} and M_{65}, for configurations of two Little (L) and two Big (B) cores at different frequencies (indicated by the number following L or B in the first column of the tables). In 38 of the 60 cases, the error for both execution time and energy consumption is less than 10%, with larger errors mostly related to extreme frequency choices. Across all 60 cases the average error is less than 8.63% for time and less than 9.69% for energy. This validates the main hypothesis: power and performance measurements at the minimum and maximum available CPU frequencies of the cores are sufficient to model energy and execution time with a reasonable precision for a wide range of configurations. Our models can be used for the appropriate selection of cores for heterogeneous platforms and the evaluation of different scheduling policies.

5 Conclusion

This work considered the problem of modelling and evaluating execution time and power/energy consumption of asymmetric multicore systems, using as a case study the execution of the Montage scientific workflow on an asymmetric multicore processor of ARM big.LITTLE architecture. The approach described provides energy and execution time estimations for a wide range of CPU frequency configurations based on metrics monitored at a smaller set. Our approach allows users to select core and frequency configurations that achieve different trade-offs between execution time and energy consumption. Future work could investigate more elaborate modelling of system overheads to improve the accuracy of the estimations and can use such models to assess different scheduling policies.

Acknowledgment. This work was supported through a collaboration grant from HiPEAC (www.hipeac.net), the RoMoL ERC Advanced Grant (GA 321253), by the Spanish Ministry of Science and Innovation (contract TIN2015-65316-P), and by Generalitat de Catalunya (contracts 2014-SGR-1051 and 2014-SGR-1272).

References

1. Ahmad, I., Ranka, S., Khan, S.U.: Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy. In: Proceedings of the IPDPS, pp. 1–6. IEEE (2008)
2. Canon, L.C., Jeannot, E., Sakellariou, R., Zheng, W.: Comparative evaluation of the robustness of DAG scheduling heuristics. In: Gorlatch, S., Fragopoulou, P., Priol, T. (eds.) Grid Computing: Achievements and Prospects, pp. 73–84. Springer, Boston (2008). https://doi.org/10.1007/978-0-387-09457-1_7
3. Chronaki, K., Rico, A., Casas, M., Moreto, M., Badia, R., Ayguade, E., Labarta, J., Valero, M.: Task scheduling techniques for asymmetric multi-core systems. IEEE Trans. Parallel Distrib. Syst. **28**(7), 2074–2087 (2017)
4. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-Science: an overview of workflow system features and capabilities. Future Gener. Comput. Syst. **25**(5), 528–540 (2009)

5. Deelman, E., Singh, G., Su, M.H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J.: Pegasus: a framework for mapping complex scientific workflows onto distributed systems. Sci. Program. **13**(3), 219–237 (2005)
6. Durillo, J.J., Nae, V., Prodan, R.: Multi-objective energy-efficient workflow scheduling using list-based heuristics. Future Gener. Comput. Syst. **36**, 221–236 (2014)
7. Etinski, M., Corbalan, J., Labarta, J., Valero, M.: Understanding the future of energy-performance trade-off via DVFS in HPC environments. J. Parallel Distrib. Comput. **72**(4), 579–590 (2012)
8. Ge, R., Feng, X., Cameron, K.W.: Modeling and evaluating energy-performance efficiency of parallel processing on multicore based power aware systems. In: Proceedings of the IPDPS, pp. 1–8. IEEE (2009)
9. Katz, D.S., Jacob, J.C., Deelman, E., Kesselman, C., Singh, G., Su, M.H., Berriman, G., Good, J., Laity, A., Prince, T.A.: A comparison of two methods for building astronomical image mosaics on a grid. In: Proceedings of the IEEE International Conference on Parallel Processing Workshops, pp. 85–94. IEEE (2005)
10. Kumar, R., Tullsen, D.M., Ranganathan, P., Jouppi, N.P., Farkas, K.I.: Single-ISA heterogeneous multi-core architectures for multithreaded workload performance. ACM SIGARCH Comput. Archit. News **32**(2), 64–75 (2004)
11. Li, K., Tang, X., Veeravalli, B., Li, K.: Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems. IEEE TC **64**(1), 191–204 (2015)
12. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE TPDS **13**(3), 260–274 (2002)
13. Van Craeynest, K., Jaleel, A., Eeckhout, L., Narvaez, P., Emer, J.: Scheduling heterogeneous multi-cores through performance impact estimation (PIE). ACM SIGARCH Comput. Archit. News **40**(3), 213–224 (2012)
14. Von Laszewski, G., Wang, L., Younge, A.J., He, X.: Power-aware scheduling of virtual machines in DVFS-enabled clusters. In: Proceedings of the IEEE International Conference on Cluster Computing and Workshops, pp. 1–10. IEEE (2009)
15. Young, B.D., Pasricha, S., Maciejewski, A.A., Siegel, H.J., Smith, J.T.: Heterogeneous makespan and energy-constrained DAG scheduling. In: Proceedings of the Workshop on Energy Efficient High Performance Parallel and Distributed Computing, pp. 3–12. ACM (2013)
16. Zheng, W., Bao, W., Xu, C., Zhang, D.: Evaluation of the DAG ready tasks maximization algorithms in multi-core computing platforms. In: Proceedings of the CBD, pp. 110–115 (2016)