----------------------- REVIEW 1 ---------------------

PAPER: 46

TITLE: On the Maturity of Parallel Applications for Asymmetric Multi-Core Processors

AUTHORS: Kallia Chronaki, Miquel Moreto, Marc Casas, Alejandro Rico, Rosa M. Badia, Eduard Ayguade, Jesus Labarta and Mateo Valero

Tool Paper: no

----------- Summary -----------

The paper presents an experimental campaign using the PARSEC benchmarks to evaluate scheduling options on a big.little platform (the Odroid XU3), ranging between static threading, OS-based task migration policies and user-level tasking available from the runtime system.

----------- Questions/Comments to Address in Rebuttal -----------

None.

----------- Detailed Review Comments -----------

I do not see a contribution in this paper. The authors present a range of experiments with a popular parallel benchmarking suite on Big.Little. The experiments use schedulers that have already been presented in the literature. The authors offer some critical comparison of the schedulers but this is narrow and more importantly, repeating known results and trade-off's. It is obvious that a user-level task-based runtime while balance workload better and achieve higher performance than static threading or an OS-based approach on this instance.

What is even more concerning here is that the authors miss the point of an architecture like big.Little in the first place, which is to run truly heterogeneous and multi-tasking workloads, with tasks serving different purposes (e.g. processing simple events vs. running compute-intensive kernels). Taking standalone parallel applications and splitting them between big and little cores for a 15% performance boost is really little return for the trouble. Things like the big.little thrive with realistic mobile computing applications.

Beyond the above the paper offers too little to the reader. There are no interesting or new metrics reported in the experiments, particularly metrics that capture throughput with multi-tasking workloads on heterogeneous multi processors (see e.g. recent papers by Lieven Eeckhout's or Mike O'Boyle's group). The experimental analysis is not rigorous, sticking to high level conclusions based on time measurements and power measurements. For example, in no place the authors measure idle time in their applications, or core utilization, or other metrics that prove how different schedulers affect load balancing, locality or other key properties in the codes. There is also no new scheduling algorithm, or even some sort of tuning of the algorithms presented in the paper, despite the fact that the authors have a fully functioning experimental environment to explore parameters of the schedulers and tune them accordingly.

----------------------- REVIEW 2 ---------------------

PAPER: 46

TITLE: On the Maturity of Parallel Applications for Asymmetric Multi-Core Processors

AUTHORS: Kallia Chronaki, Miquel Moreto, Marc Casas, Alejandro Rico, Rosa M. Badia, Eduard Ayguade, Jesus Labarta and Mateo Valero

Tool Paper: no

----------- Summary -----------

This paper presents evaluation of multiple scheduling mechanisms (e.g., OS and task-based application design) on running the PARSEC multi-threaded benchmark applications on an ARM platform with asymmetric cores.

----------- Questions/Comments to Address in Rebuttal -----------

Is ARM a good choice for parallel HPC or data analytics applications? Please give evidence (existing publication or empirical results) showing that the choice is overall appealing in terms of performance, energy, and usability.

----------- Detailed Review Comments -----------

The paper needs to first convince readers that current ARM or similar AMC architecture is suitable for HPC or PARSEC-like parallel applications. The TaihuLight or earlier Xeon Phi based machines or more recent KNL based machines do adopt AMC, but not ARM-like AMC. Though ARM processors are energy-efficient, the overall energy consumption may not be lower, considering longer computation time and larger node footprint. For example, PARSEC itself is a benchmark suite containing data-intensive applications. What is the point of evaluating it on a system where the 672MB input (plus intermediate data structures) of "dedup" cannot fit into main memory?

Along this vein, the work has the resources for performing an interesting study, assessing the potential energy efficiency for using ARM-like architecture for HPC/data-analytics workloads. Unfortunately, the authors seem to be pursuing more detailed solutions without validating the aforementioned major assumptions. Not every application needs to be ready for every new architecture developed.

The asymmetric architectures adopted by supercomputers today, for example, mostly have one powerful and general purpose "big" processor leading many worker-style, simple-minded "little" processors, with large per-node DRAM. This fits the more homogeneous and data-parallel nature of HPC applications. The half-big-half-little hybrid architecture appears to fit the heterogeneous nature of personal computing (e.g., the concurrent apps running in foreground and background on a phone or tablet). Therefore, rather than reporting the effectiveness of different scheduling schemes, the authors have to convince people first why the "2+2" configuration is sensible for HPC. More specifically, energy wise, it's hard to see whether for any application, the 2+2 mode is more energy-efficient than "4+0" or "0+4" (which is against the ideal speedup calculation given in Sec IV). Taking one step back, it's still interesting to assess what kind of applications would benefit, energy wise, from u sing little rather than big cores, and if so, is the performance degradation (not compared to the big cores, but typical supercomputer or data center processors) tolerable for their users?

----------------------- REVIEW 3 ---------------------

PAPER: 46

TITLE: On the Maturity of Parallel Applications for Asymmetric Multi-Core Processors

AUTHORS: Kallia Chronaki, Miquel Moreto, Marc Casas, Alejandro Rico, Rosa M. Badia, Eduard Ayguade, Jesus Labarta and Mateo Valero

Tool Paper: no

----------- Summary -----------

The paper investigates the PARSEC benchmarks on an ARM big.LITTLE system investigating different scheduling and parallelization strategies. They look at the performance, power and EDT of the workload with the different strategies along with different big.LITTLE configurations.

----------- Questions/Comments to Address in Rebuttal -----------

In a lot of the figures the baseline is relative to 1 little core yet for the investigation the base system is different. For example figure 6 where the starting system is 4 large cores and the object of the study is to determine if adding little cores is beneficial. The graph would be easier to interpret if either a line of the real baseline (e.g. 4 big cores) was drawn across so that one can see relative improvement or not OR just use the 4 big core case as the baseline to normalize too.

----------- Detailed Review Comments -----------

The paper covers a series of scheduling and parallelization strategies for the PARSEC benchmarks for the ARM big.LITTLE node. They show results for power, performance, and EDP. Improving the motivation for the work would make the paper stronger by showing the application/workload that is the target of this AMC architecture. For example there is mention of supercomputing and mobile market but the PARSEC benchmark suite is really neither. The analysis in the paper covers what the different scheduling strategies provide for the workload but the presentation of the results could be improved. For example the comparison or baseline seemed to always be 1 little core even though for some comparisons that is not the ideal baseline...see above comment on figure 6. The strengths of the paper it is well written and that it covers multiple strategies for both scheduling and parallelization and also looks at different configurations of varying the number of cores both big and small.

----------------------- REVIEW 4 ---------------------

PAPER: 46

TITLE: On the Maturity of Parallel Applications for Asymmetric Multi-Core Processors

AUTHORS: Kallia Chronaki, Miquel Moreto, Marc Casas, Alejandro Rico, Rosa M. Badia, Eduard Ayguade, Jesus Labarta and Mateo Valero

Tool Paper: no

----------- Summary -----------

The paper presents an evaluation of different schedulers (OS-level and user-level/runtime, static and dynamic) and programming models (pthreads, openmp loops with static and dynamic scheduling, task-basked) on an actual asymmetric multi-core (AMC) using the Parsec benchmark suite. Experimental results show that the runtime system is the best entity for making scheduling decisions. While a heterogeneity-aware OS scheduler improves performance relative to a scheduler that is heterogeneity-unaware, relying on the runtime to dynamically schedule tasks on a AMC provides better performance and energy consumption.

----------- Questions/Comments to Address in Rebuttal -----------

i. What is the baseline system for Figure 4? The columns corresponding to a system with 0 big cores and 4 small cores and static threading are not equal to 1 (In Figure 5, all columns for the same system are 1).

ii. It is not clear why exactly the task-based approach is better for smaller sized work chunks and dynamic scheduling is better for coarse-grained applications. How do the two compare when the task sizes for the task-based approach and the chunk sizes used for dynamic scheduling are the same?

iii. In figure 8, since the task-based approach keeps all available cores (both big and small) busy, shouldn't the average power consumption increase as the number of small cores are increased? What is the power ratio for the big and small cores for the evaluated applications? Updating Table 1 with this information will be helpful for readers.

iv. Why were only 9 of the 13 Parsec benchmarks used in the evaulation?

----------- Detailed Review Comments -----------

Major comments:

i. The paper provides an evaulation of different schedulers and programming models for parallel applications on AMC using the Parsec benchmark suite. Though some of the findings/conclusions are known(for eg. heterogeneity-unaware vs heterogeneity-aware OS scheduler, static scheduling vs dynamic scheduling). The paper includes new results in the form of comparing the GTS scheduler (a heterogeneity-aware scheduler) with a runtime task-based scheduler.

ii. The paper provides results for only one combination of frequencies of the big and small cores (big cores at 1.6GHz, small cores at 800MHz). Other frequency combinations may show different behavior for some benchmarks, please include results for other combinations (at least for section V.A) as well.

Minor comments:

i. In Section V, it will be easier for readers if names such as All-Big (for example) as used instead of configuration 4+0. The equations in Section IV list the number of small cores first, while in Section V, the configurations list the number of big cores first, using the same order in all places will be helpful to readers.

ii. In abstract, heterogeneous-aware -> heteogeneity-aware.

iii. In Figure 2, facesim and fluidanimate have fewer data points than other benchmarks. Similarly, facesim and fluidanimate have fewer data points in figures 7 and 8. Please include the missing data points.

------------------------------------------------------