======================================================================

HPCA23 Review #265A

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

Paper #265: On the Maturity of Parallel Applications for Asymmetric

Multi-Core Processors

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

===== Paper summary =====

The paper evaluates different scheduling algorithms/different programming models for PARSEC benchmarks on ARM Big/Little asymmetric multi-core architecture. The evaluation reports both power and performance all measured in a real system.

Evaluated thread scheduling algorithms are static scheduling, GTS, task-based scheduling.

The paper also evaluates loop-static, loop-dynamic OpenMP scheduling policies. It also evaluates the performance/power efficiency by adding little cores.

The results show that in general task-based scheduling is better for performance and power.

Overall merit: 1. Reject

Reviewer expertise: 3. I know the material, but am not an expert

Writing quality: 3. Adequate

Experimental methodology: 4. Good

Novelty: 2. Incremental improvement

===== Strengths =====

Use a real asymmetric processor to evaluate the different scheduling/execution models.

It also measures power efficiency as well.

===== Weaknesses =====

No new idea is proposed.

The work is confirming the needs for a better scheduler that understand the workload characteristics.

===== Comments for author =====

The paper reads more about measuring performance/power with different scheduling algorithms. Although there has been no work that reports this much comprehensive results for ARM big/little cores, the findings are not so new.

If the evaluated workload is much wider, the results could be more interesting.

The paper will be more suitable for workload characterization conferences.

Although the paper reports all the real measured data on a real system with the real scheduler, the scheduling on asymmetric core has been studied for a long time as the authors summarized well in the related work section. Even though the real system can provide good data but the other hand, all the evaluations are all limited to the existing run-time system scheduling and loop scheduling algorithms, which give very little new insights from this paper.

======================================================================

HPCA23 Review #265B

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

Paper #265: On the Maturity of Parallel Applications for Asymmetric

Multi-Core Processors

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

===== Paper summary =====

This paper considers a big.LITTLE Samsung Exynos 5422 CPU system when running PARSEC. The Hardkernel Odroid-Xu3 development board is used. The goal is to evaluate when big.LITTLE is going to be a good target for server workloads. The paper considers 3 different scheduling schemes and compares performance speedup and power savings versus the out-of-the-box static scheduler. A range of results provided when varying the number of big and little cores.

Overall merit: 3. Weak accept

Reviewer expertise: 3. I know the material, but am not an expert

Writing quality: 4. Well-written

Experimental methodology: 4. Good

Novelty: 2. Incremental improvement

===== Strengths =====

+ big.LITTLE platform has been adopted by embedded systems, but it is not clear if this is a viable platform for servers

+ measurements are made on live systems

+ the results are interesting

===== Weaknesses =====

- The analysis is not particularly strong. The choice of configurations is not well motivated.

- The results are not well explained - there is little description of why a particular PARSEC benchmark performs better/worse with a particular scheduler, and for a specific configuration

- It would have been interesting to evaluate a wider class of workloads

===== Comments for author =====

I appreciate the evaluation in this paper. But I am not sure that HPCA is the right conference for this paper. While the paper considers a large number of core configurations and scheduling schemes, I did not see a lot of analysis that could have guided this exploration. Further, I would have appreciated a deeper dive into the results on a per application basis.

===== Questions for authors =====

1. Is this a reasonable platform for PARSEC-class applications? Are there better big.LITTLE targets?

2. In your power measurements, you mention 4 current sensors. Which ones are you capturing?? It sounds like you are including the GPU, but that is not clear.

2. Is OpenMP too heavy-weight to allow for a fair evaluation?

3. How does the best configuration compare to an X86 solution homogeneous?

4. PARSEC is just one class of workload (multithreaded) - did we learn anything about other classes of workloads out of this study?

5. The differences between GTS and task-based are not convincing that one is clearly better. Are there other reasons why I would pick one versus the other? You even tend to argue for one versus the other, and then reverse yourself.

======================================================================

HPCA23 Review #265C

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

Paper #265: On the Maturity of Parallel Applications for Asymmetric

Multi-Core Processors

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

===== Paper summary =====

This paper studies the performance affects of having both little and big cores on a multi-processor chip from a task scheduling perspective using PARSEC benchmark suite. Various coarse and finer grain scheduling methods are evaluated. Evaluation is performed on a development board featuring a big.LITTLE architecture. The paper concludes that in most cases having little cores hurt performance (both delay and power) unless a finer grain scheduling such as "task based" scheduling is used.

Overall merit: 3. Weak accept

Reviewer expertise: 3. I know the material, but am not an expert

Writing quality: 4. Well-written

Experimental methodology: 4. Good

Novelty: 2. Incremental improvement

===== Strengths =====

Real hardware is used in the evaluation. A somewhat surprising result is presented which shows asymmetric multi-core architectures may not be suitable for coarse grain scheduling. Hence, the paper provides a strong argument that for asymmeteic multi-core designs long executing threads may be harmful and a much finer task granularity is needed.

===== Weaknesses =====

Several of the scheduling techniques require modification of the programs; particularly task based scheduling. This introduces the user factor into the evaluation.

===== Comments for author =====

Overall, I find the paper easy to follow and well written, with the exception of broken sentences in several spots and some number of typos. Examples include: "oppositional benefits": perhaps relative benefits? "We can alternatively dynamic loop scheduling", "in oder to" ...

I am having a difficult time interpreting Figure 5. Perhaps it is me. What is the y-axis? Inverse of delay? It will be a good idea to split this figure and move EDP elsewhere.

My main concern with this paper is the modifications necessary on these programs to implement dataflow based task scheduling. The paper does not make it clear whether a different run-time system suitable for task scheduling has been employed, or, this is a simple modification on top of open MP.

This paper may also be more suitable to a conference such as Sigmetrics and not HPCA. This is primarily an evaluation paper.

===== Questions for authors =====

1. What is the effort that is required to modify given programs so that the desired scheduling is achieved? If a different user had adopted the benchmarks for a particular scheduling mechanism, particularly the task based scheduling, would the results change?

2. Does task based scheduling use a different run-time system than others?

======================================================================

HPCA23 Review #265D

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

Paper #265: On the Maturity of Parallel Applications for Asymmetric

Multi-Core Processors

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

===== Paper summary =====

The paper evaluates three different scheduling approaches on an ARM big.LITTLE asymmetric multicore platform using the PARSEC benchmark suite. The three scheduling approaches studied are (a) Static threading that makes scheduling decisions at the application level, (b) Global task scheduling where scheduling decisions are made by OS, and (c) task-based scheduling where scheduling decisions are made at the runtime level. The paper concludes that heterogeneous aware OS scheduler and runtime scheduler achieves better load balancing and performance than static threading approach that makes scheduling decisions at the application level.

Overall merit: 2. Weak reject

Reviewer expertise: 3. I know the material, but am not an expert

Writing quality: 3. Adequate

Experimental methodology: 4. Good

Novelty: 2. Incremental improvement

===== Strengths =====

1. The paper is touching an important topic in the HPC domain.

2. Strong evaluation section.

===== Weaknesses =====

The advantage of using task-based applications for load balancing is well known. Other than confirming this previous knowledge, the contribution of this paper is unclear.

===== Comments for author =====

The paper has a strong evaluation section that discusses the power and energy consumption in addition to the performance of various scheduling techniques on asymmetric multi-core processors. However, task-based runtime performing better load balancing because of the lack of global synchronization in task-based applications is well known. The related work section in the paper has listed several prior works that explore task scheduling for load balancing and better utilization of available compute resources. It looks to me that the paper is just confirming this load balancing capability of task-based applications.

Modifying the runtime to make scheduling decisions based on the compute capability of the asymmetric cores or modifying the existing available scheduling techniques to improve performance/power would have made this a stronger paper. Without any of the above stated explorations, I think this paper has little contribution.

===== Questions for authors =====

1. The task scheduling runtime resolves the dependencies of tasks and schedules them on available cores. Where is this task scheduling runtime running? It looks like the task scheduling runtime should be running on the big cores for fast resolution of dependencies and task scheduling but I am not sure if this is enforced in anyway.

2. Did you modify the task scheduling runtime so that it is aware of the asymmetric cores and make scheduling decisions based on the compute capability of these asymmetric cores?

======================================================================

HPCA23 Review #265E

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

Paper #265: On the Maturity of Parallel Applications for Asymmetric

Multi-Core Processors

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

===== Paper summary =====

This work evaluates the trade-offs of different versions of PARSEC (and PARSECSs) with different schedulers on a big.LITTLE processor (the Samsung Exynos 5422 on the Odroid XU3).

Overall merit: 2. Weak reject

Reviewer expertise: 3. I know the material, but am not an expert

Writing quality: 3. Adequate

Experimental methodology: 3. Average

Novelty: 2. Incremental improvement

===== Strengths =====

\* The authors provide an evaluation of PARSEC showing how fine-grained task-based parallelism can help improve performance given a larger energy budget.

===== Weaknesses =====

\* Lack of a power budget and a fixed core frequency limits the applicability of this work.

\* Missing evaluation of Linux’s global scheduler (GTS) along with the task-based version of PARSEC/Ss.

===== Comments for author =====

This is an interesting study that compares schedulers (OS and runtime) to determine the applicability of task-based systems for big.LITTLE-style processor cores. By showing that the task-based system (with the proper application, runtime and scheduler support) can achieve a 13% performance improvement on average, this work clearly shows that the PARSEC/Ss suite can be improved and can take advantage of the processor cores available.

Nevertheless, there are a few drawbacks with this work that prevent me from recommending acceptance. It is generally accepted that task-based schedulers can achieve higher performance given their ability to schedule tasks across a number of compute resources (and this is re-affirmed in Fig. 9). This work appears to provide a similar result, but through the use of a real system instead of simulation or emulation. One major issue I have is the use of fixed CPU frequencies and no power budgets. I feel that achieving higher performance is definitely possible given a higher power budget (which is seen in Fig. 10), but the challenge today, as described in the Introduction, is truly energy efficiency. If that is true, then task scheduling does not seem to win in these configurations (Fig. 6: higher power, energy (in most cases) and EDP) appears to favor static-scheduling in all but one case (4+1 in Fig. 6).

While PARSEC seems to align with the goal of general-purpose processing, your use of the Exynos 5422 processor appears to target mobile platforms.

Evaluating the Juno platform could be interesting, even though there are only 2 large cores. The power envelope of that design could be closer to the general-purpose processing target that you detail in this work.

Even though EAS isn’t part of the current kernel versions, couldn’t early / beta versions be used to evaluate the ideas of energy-aware scheduling?

===== Questions for authors =====

\* Why do you use 1.6GHz/0.8GHz for your big/little cores (instead of the maximum 2.0GHz/1.4GHz values?)

\* Why evaluate the Global (GTS) linux scheduler with the original PARSEC software? How do the results change when you evaluate the task-based version allowing Linux to schedule the threads (versus the runtime doing the scheduling) and compare those to the task-based results?

\* Should there be a power limit for this work? Performance increases as the power utilization increases (higher budget).