== Paper summary ==

In multi-core CPU-GPU systems, the memory requests from GPUs can overwhelm the requests from CPUs. The memory controller with centralized request buffer will become very complex and costly for such system.

The key idea in this paper is to decouple the memory controller's three major tasks into three significantly simpler structures. The three stages are: 1) group requests by locality, 2) prioritize inter-application requests, and 3) schedule low-level DRAM commands.

The evaluation compared the proposed staged memory scheduling against other three state-of-the-art memory scheduling mechanisms. The results showed better balancing between performance and fairness in CPU-GPU systems, with much simpler design.

== Strengths of paper and mechanisms (bullet points) ==

* Presented an important challenge to memory controller scheduling with very good data analysis.

* The discussion of SMS rationale was pretty thorough. It's the key to make the mechanism much simpler than previous schedulers.

* The experimental evaluation was pretty solid. It included a lot of metrics and also evaluated on the sensitivity of parameters of SMS.

* The proposed scheduler was simpler and more flexible than state-of-the-art ones. The rigorous evaluation also showed that SMS had better potential to fulfill different needs

== Weaknesses of paper and mechanisms (bullet points) ==

* The tunable parameter, p, provided flexibility to fulfill different needs (i.e., either fairness or performance). However that can be a problem as an incorrect parameter can cause severe performance or fairness degradation.

* This memory scheduler was very simple, but it gave up some important optimization opportunity. For example, it gave up cross-source row buffer locality. It didn't try to prevent row conflict either.

* The workload consisted of unrelated workloads between CPU and GPU for evaluation, which may not represent the system performance goal very well.

== Detailed comments (expand on the summary as necessary) ==

The author raised an important problem on CPU-GPU system. GPUs, with much higher capability to issue requests to memory controller, can make state-of-the-art memory scheduling mechanism very inefficient or costly due to buffer number. The data in performance, fairness and the sensitivity to buffer size in CPU-GPU system supported the problem statement very well.

To mechanism proposed in this paper decouples the task of memory controller into three stages. Each stage focuses on a relative simple task so it does not need a lot of buffers. The key insight of this paper is that it did a good tradeoff between simplicity and performance optimization. It gave up out-of-order batch formation, out-of-order batch scheduling, out-of-order DRAM scheduling and surprisingly delivered comparable throughput and fairness.

Another interesting design is an adjustable parameter, p, which controls the probability of applying shortest job first scheduling. It makes the design attractive if the weight of GPU performance can vary in different scenarios. However, this parameter can also be a problem if the value is incorrectly selected.

Overall, the evaluation was quite solid. It compares the proposed SMS mechanism with other three state-of-the-art mechanisms (FR-FCFS, ATLAS, TCM). It considers most of the important metrics like speedup, fairness, GPU frame rate, cost, and

power. The results showed that it can reach comparable or better tradeoff between speedup and fairness, with acceptable GPU frame rate and much simpler design.

One thing that can be improved is the workload. The truly interesting performance index should be the tightly-coupled workload, which really exploits the benefit of parallel computing on CPU-GPU system. It will be better if this paper can show the performance data without GPU.

== Ideas for improvement - Can you do better? ==

* Research on inter-dependent, state-of-the-art benchmark for CPU-GPU multicore system. Use it to validate state-of-the-art memory controllers and maybe come up with a more efficient scheduling mechanism.

* Research other mechanisms which can do better tradeoff between performance and fairness by considering more optimization opportunities. It shouldn't limit to CPU-GPU only and should take more system configurations into account.

== Lessons learned ==

* The best part of this paper was a simple yet effective memory scheduling mechanism. Most of the time we want to build a mechanism with optimal performance without the consideration of complexity. In fact, SMS gave up several optimization opportunities, but it still performs fairly well. That is a good demonstration of the beauty of simplicity.

* Another good lesson was that sometimes it would be better to look at problems differently. While there are a lot of good memory scheduling mechanisms for multi-core system, they may not work that well when taking GPU into account. By introducing an important factor, it can raise a new interesting and valuable problem to solve.