Critique Summary

**Paper:** Facilitating Efficient Synchronization of Asymmetric Threads on Hyper-Threaded Processors  
**Author:** Nikos Anastopoulos, Nectarios Koziris  
**Summarized by:** Fei Hong

**Overview:**
This Paper presents a new way for synchronizing application threads that execute on hyper-threaded processors, and characterized by workload asymmetry. That is, use the privileged instructions MONITOR and MWAIT, which are mostly used in operating systems code for inter-thread synchronization.

The author propose a framework through which one can use the privileged MONITOR and MWAIT instructions to build condition-wait and notification primitives, with the least possible kernel involvement. Also, the author use the primitives presented in the paper to build synchronization barriers. Experiments show that, compared to Pthreads and spin-loops-based barriers, the implementation for fine-grained inter-thread synchronization and the synchronization barriers have better performance.

**Summary of strengths:**
1. This paper presents previous works on implementation of synchronization. The author pointed out their deficiencies in system performance.
2. The framework presented in the paper is both responsive and less processor intensive (i.e. with spin locks).
3. The scheme prevents resource waste on a hyper-threaded processor and the notification of the waiting thread does not require operating system intervention but a single memory value update.
4. The MONITOR and MWAIT based synchronization provides an excellent balance between resource waste, wake up latencies, and call overhead.
5. The speedup enhancements for improving the thread synchronization performance and throughput is going to be much more important especially in multi-core chip processor architectures.

**Summary of weaknesses:**
1. The framework is processor specified. The MONITOR and MWAIT instructions may not be available on the other processors. So the processors that can use this scheme are limited.
2. The implementation of the scheme requires the modification of OS. And those modifications are specified to a portion of the processors. So, if a hardware platform changes, the original scheme may not work.
3. The experiment should do more to other thread cases and different combinations of thread characteristics and numbers. Other than the single case that a busy thread executes with a more idle thread.
4. The instructions require special considerations in the OS, as well as the architecture on which the OS is running. So, when it comes to virtualized systems, other issues should be considered for those guest OS instances running on this scheme.
5. The time that the main thread needs to invoke a synchronization primitive should be as little as possible. It increases drastically when frequent synchronizations happen between threads. So, this technique must incur low resource consumption, high responsiveness and low call overhead.
6. In order to avoid false wake-ups, extra padding of the data structure to the largest monitor line size is required which can be an unnecessary overhead.