Friday Jan. 26 Paper Summary

1. For me the most interesting part of the paper was how much the execution time and off-chip memory accesses can vary based on the workload size, and especially the accelerator. This wasn't really apparent to me before and makes a strong case for why Cohmeleon may be necessary to improving SoC cache coherence.
2. The use of active learning to decide which coherence method is the most optimal for each accelerator is certainly an interesting decision as most hardware decisions are often made at design-time. Going down this route may help optimize the cache performance with older hardware even while the workloads or accelerator drivers adapt and are updated.
3. I was overall surprised at the performance that Cohmeleon can achieve, given the results show it is Pareto-optimal for off-chip memory accesses and execution time on all of the modeled SoC's.

For me the biggest downfall in this paper is the transparency of cache coherence to application programmers. It wasn't discussed in detail, but the chance of preventing legacy software from being run is certainly higher with this methodology. Also, the transparency of the cache can expose security vulnerabilities where some accelerator's memory accesses and cache coherence modes could be available to another workload.

The biggest takeaway from this paper for me, is that the cache coherency method on an SoC can be adapted at runtime for better performance. For designing future systems, this may provide inspiration to include hardware structures that alter the cache coherence method at runtime, even if it's not exposed to programmers.