**Why Do We Care?**
- Achieving good performance on today’s platforms is really hard
  - Must be an expert in the architecture and the application
  - Many cases still require exhaustive search of optimization parameters
- Obtaining good performance is going to get increasingly difficult for manycore architectures
  - Greater diversity of platforms: cell phones, laptops, etc.
  - Complex interactions between threads
  - Unknown mix of applications space sharing the machine
- Current optimization techniques aren’t going to be enough
  - Search space is too large for purely exhaustive
  - Machine state varies from run to run
- We are exploring performance counters as an approach to get insight into an application’s performance and adapt during runtime
  - Hints to OS scheduling
  - Hints to Online Autotuning

**Application Overview**
- Parsec Fluidanimate (Intel)
  - Benchmark Fluid Dynamics Solver
  - Simulates the underlying physics of fluid motion for real-time animation purposes with the SPH algorithm (Smoothed Particle Hydrodynamics)
  - Algorithm similar to the one from the ‘Parallelize Particle Simulation’ assignment from class
  - Exhibits coarse-granular parallelism, static load balancing
  - Contains large working sets, some communication

- Chombo Finite Elements Solver
  - Used in the ParLAB Health Application to simulate bloodflow in arteries as an incompressible fluid
  - Uses finite differences to discretize partial differential equations on block-structured, adaptively refined grids using published algorithms
  - This specific application uses the Poisson Solver for Oct-Tree Adaptive Meshes introduced by Martin & Cartwright

**Interesting Results**

**Architecture Overview**
- Dual Socket – Quad Core Intel x86 Nehalem
  - CPU_L2MISS
  - CPU_L2MISSALL
  - CPU_L2REQ
  - Hyperthreaded- 2 Thread Context Per Core
  - Private L1 (32K) and L2 (256K) per Core
  - Inclusive Shared L3 (8 MB)
  - Up to 6 instructions issued per cycle
  - 10 outstanding data cache misses at a time
  - 635 Events available for Performance Counters

- Single Socket – Six Core x86 SiCortex
  - CPU_L2MISS
  - CPU_L2MISSALL
  - CPU_L2REQ
  - Single Threaded
  - Private L1 (32K) per Core
  - Inclusive Shared Partitioned L2 (256K/Core)
  - Up to 2 instructions issued per cycle
  - 1 outstanding data cache miss at a time
  - 3993 Events available for Performance Counters

**Parsec L2 Cache Behavior on SiCortex**
- Cache requests greatly increase with more cores
- Cache misses go from 65% to 97%
- Cache misses going to DRAM remains constant
- Data is just moving from L2 to L2 for 8 Cores

**Future Work**
- Short Term
  - Get the rest of the Chombo data gathered
  - Continue scaling analysis and perform better pipeline analysis of data
  - Analyze data to see if combinations of counters provide more useful insight

- Long Term
  - Propose a standard set of useful counters for profiling performance and energy usage at runtime
  - Apply machine learning algorithms to find trends

**Conclusions**
- Too Many Counters, not enough useful ones
  - Semantics of counters between different machines never quite exactly the same
  - Need a movement towards standardization
  - Keep counters useful for hardware debugging
  - Standardize on one’s most useful for predicting application performance and energy usage

- Most useful counters are:
  - Total Cycle Counts
  - Instructions Committed
  - Last level cache misses
  - Missing counters
  - Energy metering
  - Cache sharing statistics (present on SiCortex)