CS 194 Results

Nathan Beckmann
Advisor: Glenn Reinman

Agenda

- Introduction & Motivation
- Approach
- Results
- Conclusion

Introduction

- My project explores cache configurations for physics simulation
  - Chip Multiprocessors (CMPs) provide parallelism to run these applications efficiently
  - CMPs put additional stress on memory system in bandwidth and communication
  - Physics simulation divides work into small, independent working sets – unusual run-time behavior
  - How will different cache configurations affect application performance?

Introduction – Prior Work

- CMPs
  - IBM Cell
  - Intel Core
  - Intel Tulsa
  - Sun Niagara
  - GPU architecture
- Different application domains have led to many successful yet completely different designs for CMPs.

Introduction – Prior Work

- Caching
  - Static partitioning (simple)
    - Private caches – small caches with low latency, but high miss rate, high migration between caches
    - Shared caches – large cache shared between several (all) cores, high latency but low miss rate, low migration
    - Hybrids – share part of a local cache, keep part private
  - Dynamic partitioning (complex)
    - Cooperative caching – detect data that must be shared, keep the rest private
    - PDAS – allocate cache space based on performance thresholds

Introduction – Our Experiment

- Explore performance of the following static partitioning schemes:
  - Private L2
  - Shared L2
  - Shared-Pairwise L2
  - Shared-Pairwise L2 with backing L3
- What's new:
  - What are the memory properties of physics simulations on conventional CMPs?
- Hypothesis:
  - Due to the independent nature of each thread, the private and L3 schemes should perform well because of low L2 latency.

Approach

- Use SimpleESCalar simulator (SESC) to simulate applications
- Use PhysicsBench, a suite of physics simulations using the Open Dynamics Engine (ODE), developed by Thomas Yeh
  - Due to time constraints and simulator problems, two tests were used
    - Test A – simple, low-resource test
    - Test B – complex, high-resource test

Approach

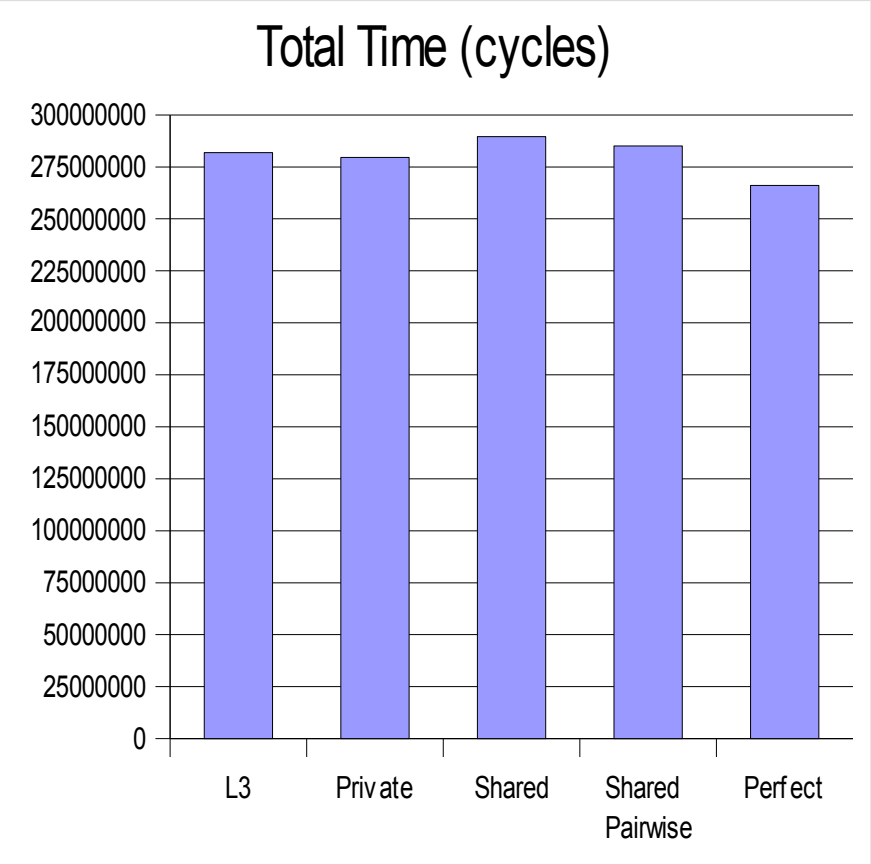
- All tests use an identical “Master core” that feeds jobs to 4 worker cores
- L2+ Cache for workers is kept to ~4MB
  - Private: Four 1MB caches, latency of 8 cycles
  - Shared: One 4MB cache, latency of 15 cycles
  - Shared-pairwise: Two 2MB caches, latency of 12 cycles
  - Shared-pairwise with L3: Two 512kB L2 caches, latency of 6 cycles; one 4MB L3 cache, latency of 20 cycles
  - Base-line “perfect” cache: Always hits with a latency of 1 cycle

Results - Overall

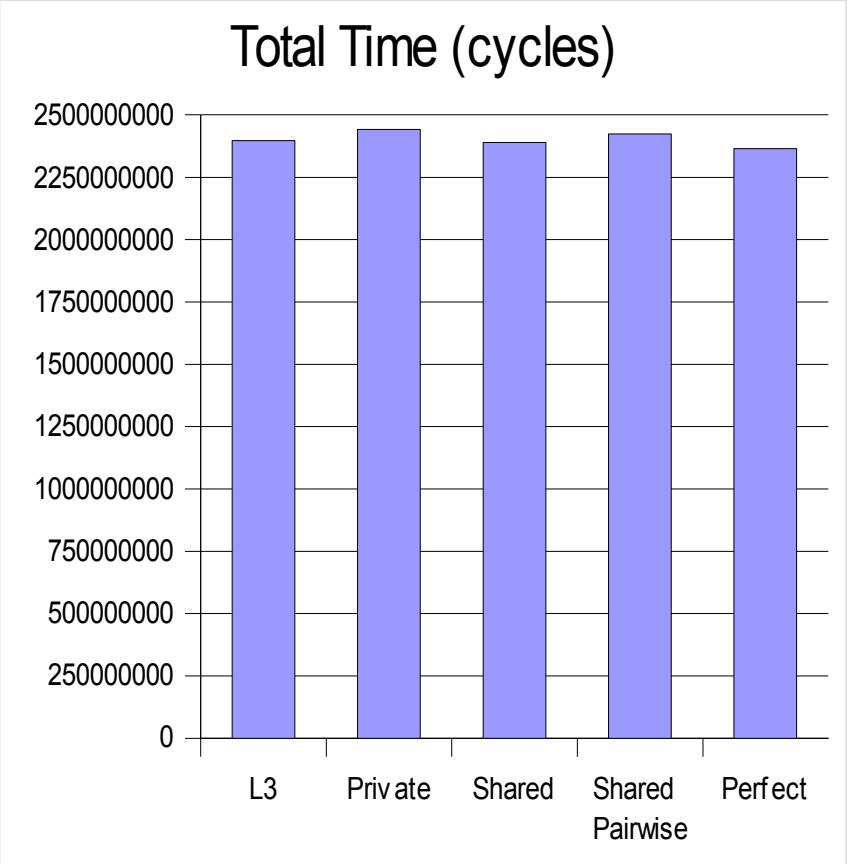
- Overall performance:
  - Measured in average no. of cycles executed by each worker thread
- Little variation among schemes
  - 2% deviation from mean
  - 6% improvement by “perfect” configuration
  - The memory system is not a bottleneck for these applications!

Results – Overall

- Test A



- Test B

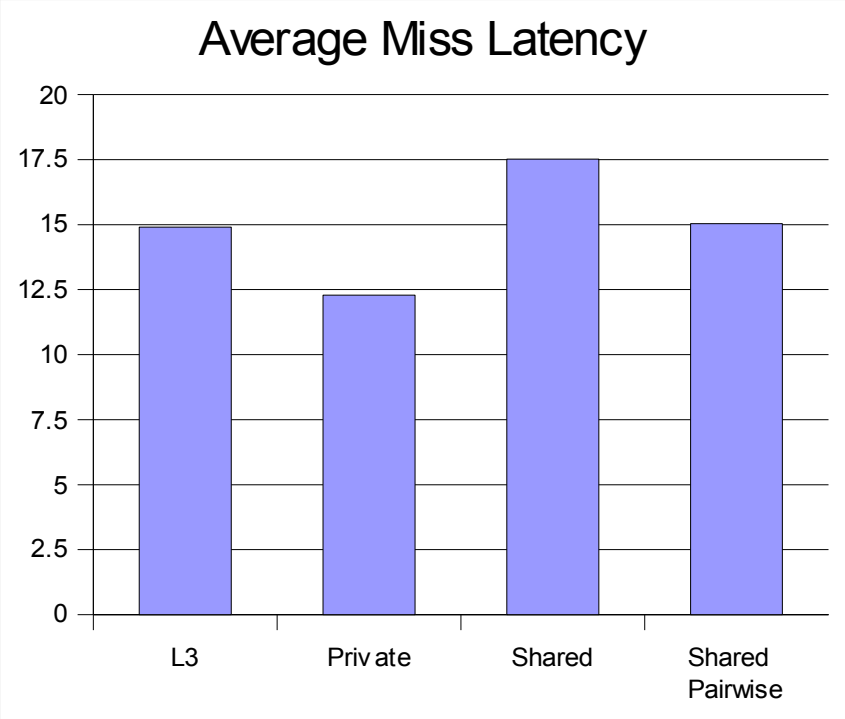


Results – Caches

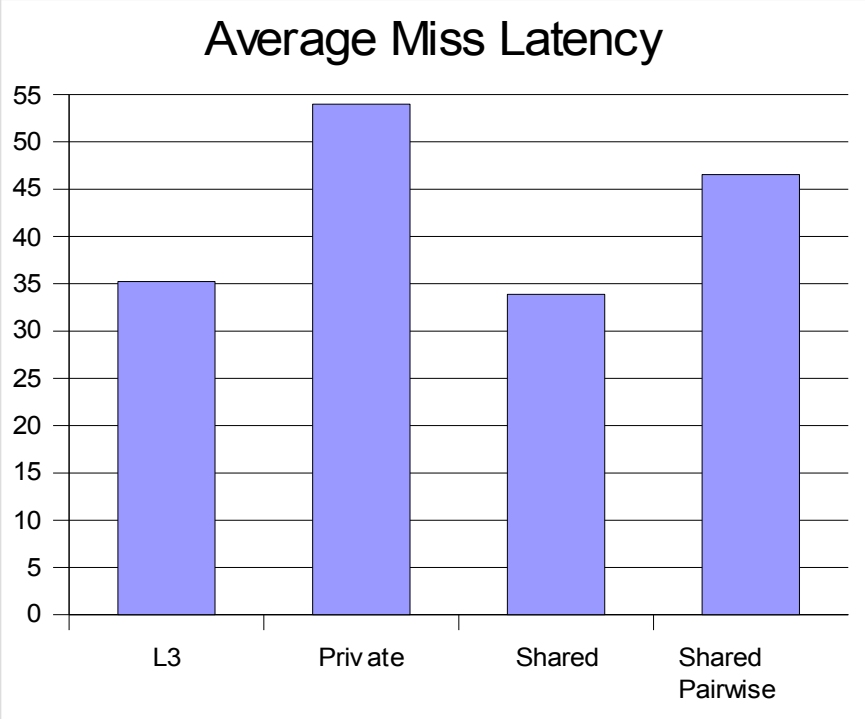
- Despite this, we can compare how each configuration fared
- Metrics:
  - Avg. Miss Latency – the access time to memory in the L2 caches; directly applicable to application performance
  - Miss rate – percentage of requests not able to be served by the L2

Results – Avg Miss Latency

- Test A

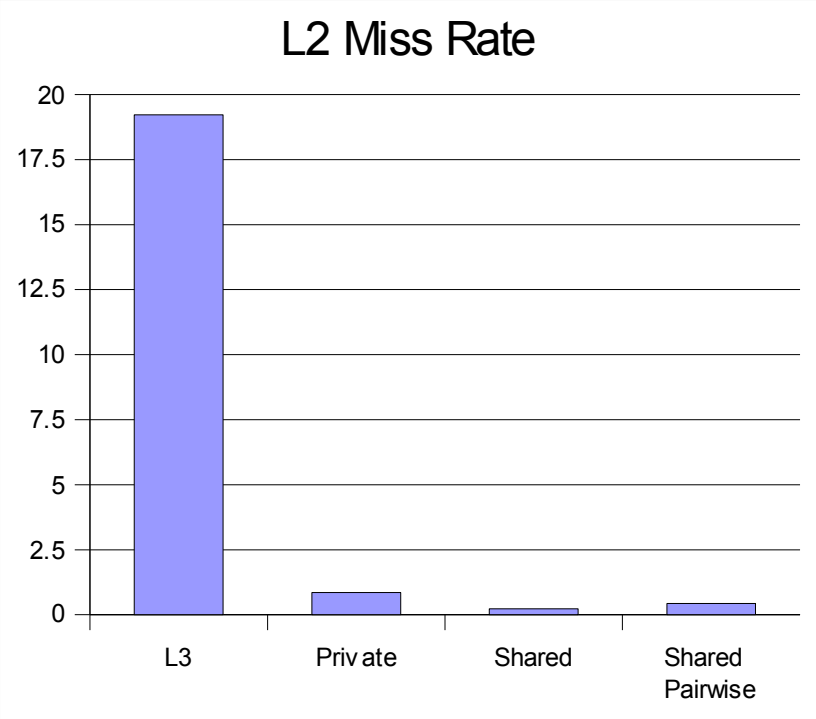


- Test B

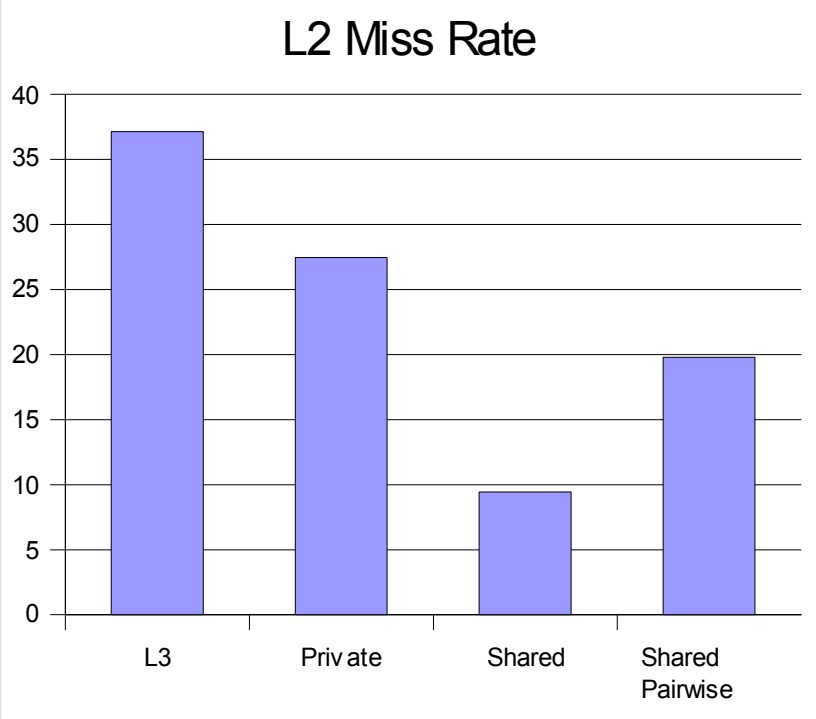


Results – Miss Rate

- Test A

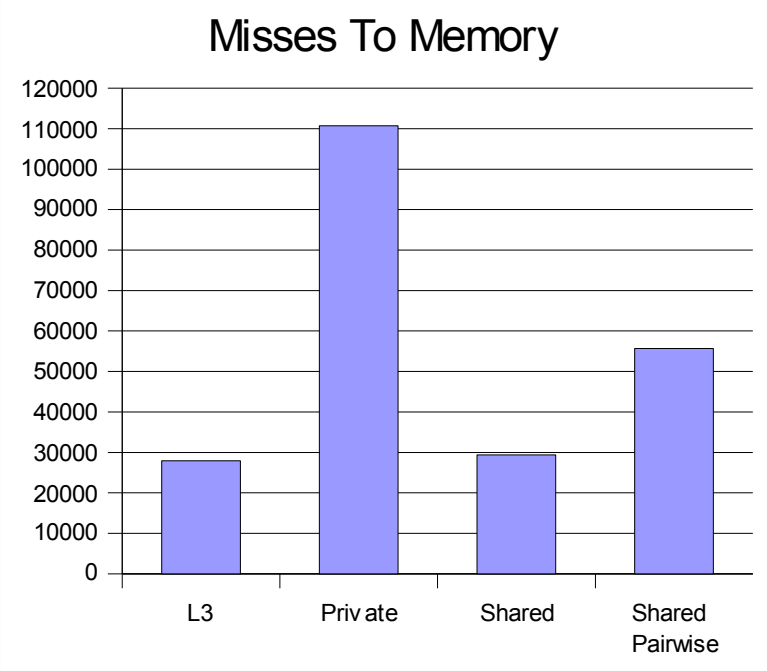


- Test B

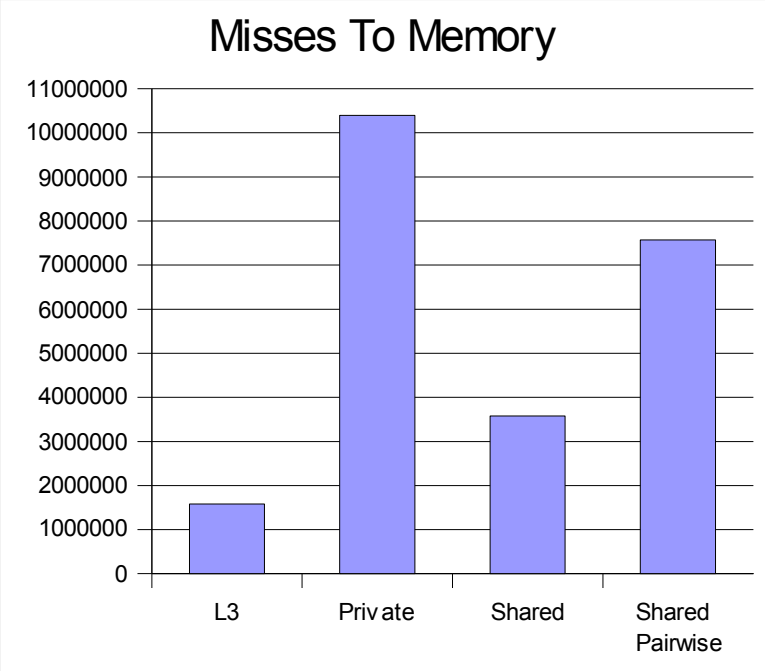


Results – Misses To Memory

- Test A



- Test B



Results – Miscellaneous

- Not shown:
  - Caches for the Master core performed nearly identically
  - L1 caches performed nearly identically
- L3 cache miss rate in the L3 scheme:
  - Test A: 2.22 %
  - Test B: 22.47 %

Results – Expected

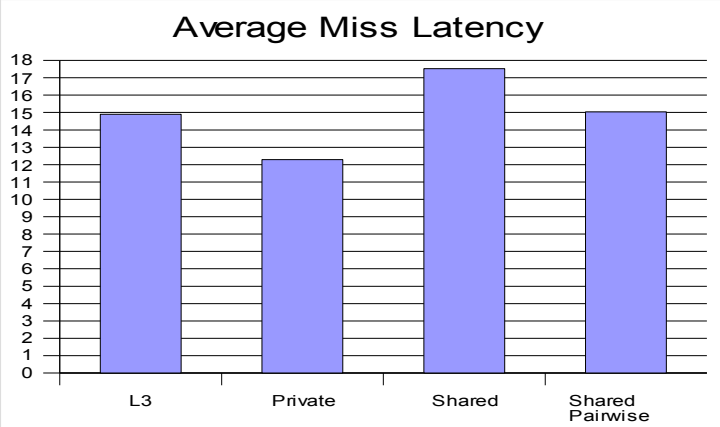
- Misses to memory highest for private, lowest for L3/shared, in between for shared-pairwise
- Shared-pairwise performs “decent” in both tests, but never best
- Private scheme fairs well in Test A, shared performs worst

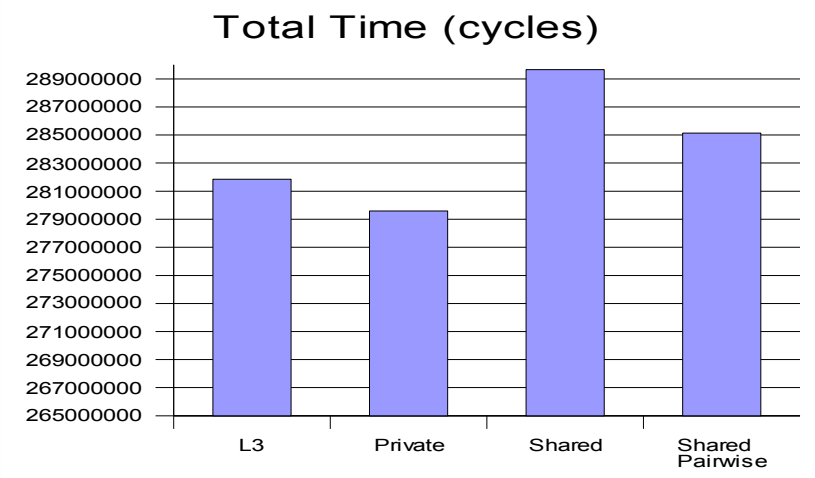
Results – Unexpected

- L3 scheme does poorly in Test A
  - L2 miss rate at 19%, private only 0.86%
    - 512 kB can't hold application memory footprint!
  - Private performs best, so application fits comfortably in 1 MB
- Private scheme does poorly in Test B
  - L2 miss rate at 27%, misses all the way to memory
    - 1 MB can't hold application memory footprint!
  - Shared scheme fares best; only 10% miss rate
  - L3 scheme performs nearly as well as shared
    - L3 compensates for nearly 40% L2 miss rate
    - Miss rates of other caches also quite bad

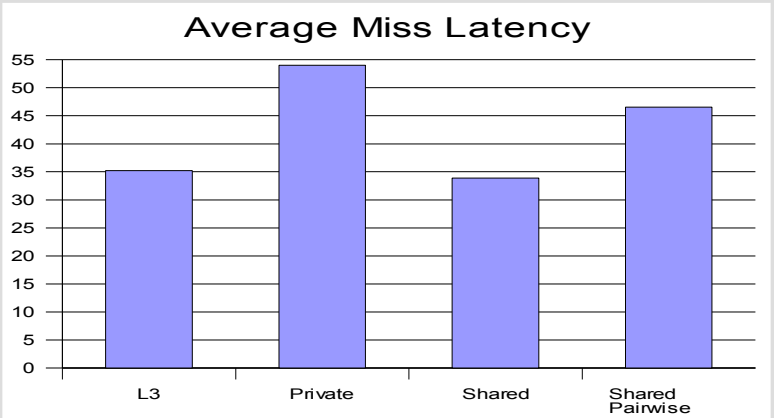
Results – Cache vs. Overall

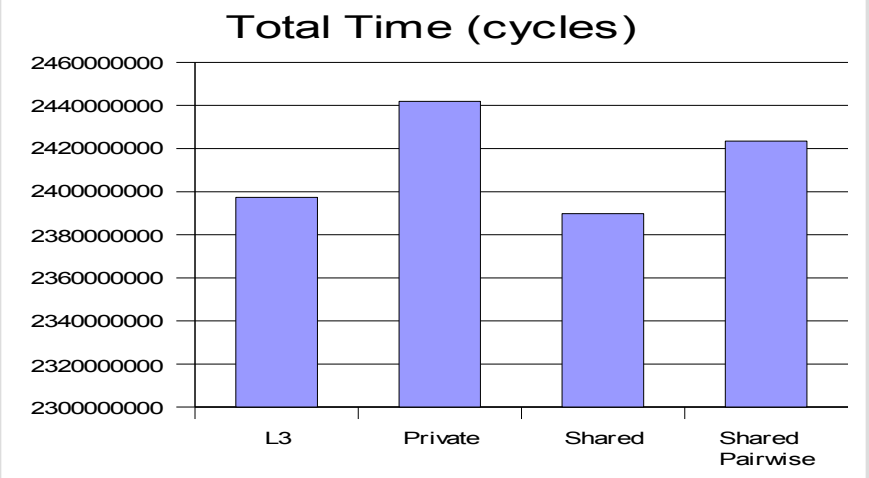
- Test A





- Test B





Results – Cache vs. Overall

- Clear correlation between cache access time and application performance
- But 18% variation in cache performance accounts for only 2% application performance

Conclusions

- Even under an idealized cache, performance gain is disappointing
  - Optimization efforts could be better spent improving on-core execution
- Memory properties of physics applications
  - Footprint size varies considerably
  - Private schemes fare well only if test uses small amounts of data
  - A shared-pairwise scheme provides stable performance

Conclusions

- L3 performance inconclusive
  - Provided decent performance even when L2 was much too small for application
  - Provided good performance when private scheme failed
  - Performance in Test A indicate of poor choice of parameters, not a flaw in the scheme

Future Work

- Increase aggregate cache size slightly and test performance of each scheme
  - Will L3 cache up to private in Test A and keep its performance in Test B?
- Investigate performance using dynamic partitioning
  - Different memory footprints for each application suggest adaptive partitioning

Questions

?