Initial Results on the Performance Implications of Thread Migration on a Chip Multi-Core

Y. Sazeides, P. Michaud, L. He, D. Fetis,
C. Ioannou, P. Charalambous and A. Seznec

Department of Computer Science
University of Cyprus, Nicosia

IRISA/INRIA
Campus de Beaulieu 35042 Rennes
Cedex, France

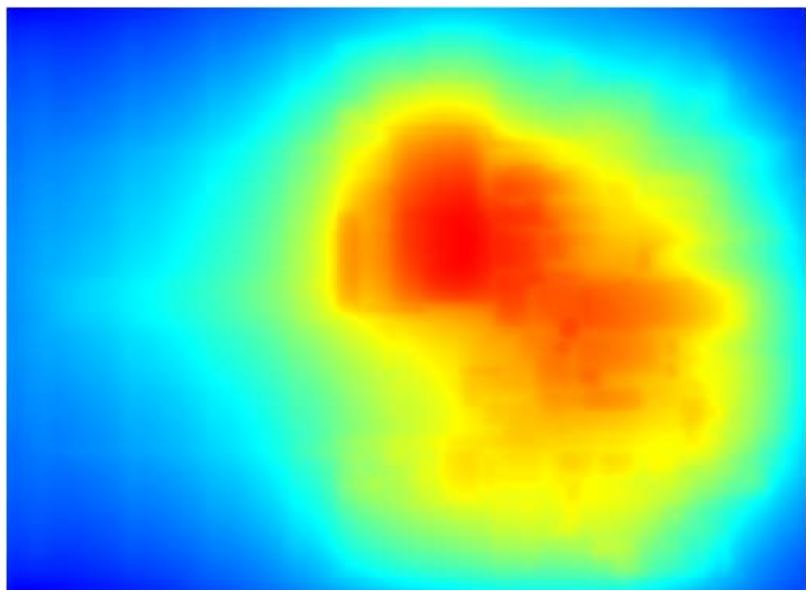
3rd HiPEAC Workshop IBM, Haifa 17-4-2007

Motivation

- Computer industry conflicting challenges:
  - Market calls for high performance cores
  - Need for low cost, low power and short time to market
- One of the industry responses is single chip multi-core processors
- Multi-cores can increase performance with thread and program level parallelism

Temperature Wall

- But parallel execution can result in
  - Overall higher power consumption
  - Non-uniform power-density map
  - Temperature increase



Produced by ATMI using a simulated power density map
http://www.irisa.fr/caps/projects/ATMI/

Harness Temperature (1)

- Improve packaging and cooling (exotic/expensive)

- Temperature is not a packaging only issue anymore: important for circuit, microarchitecture, OS design

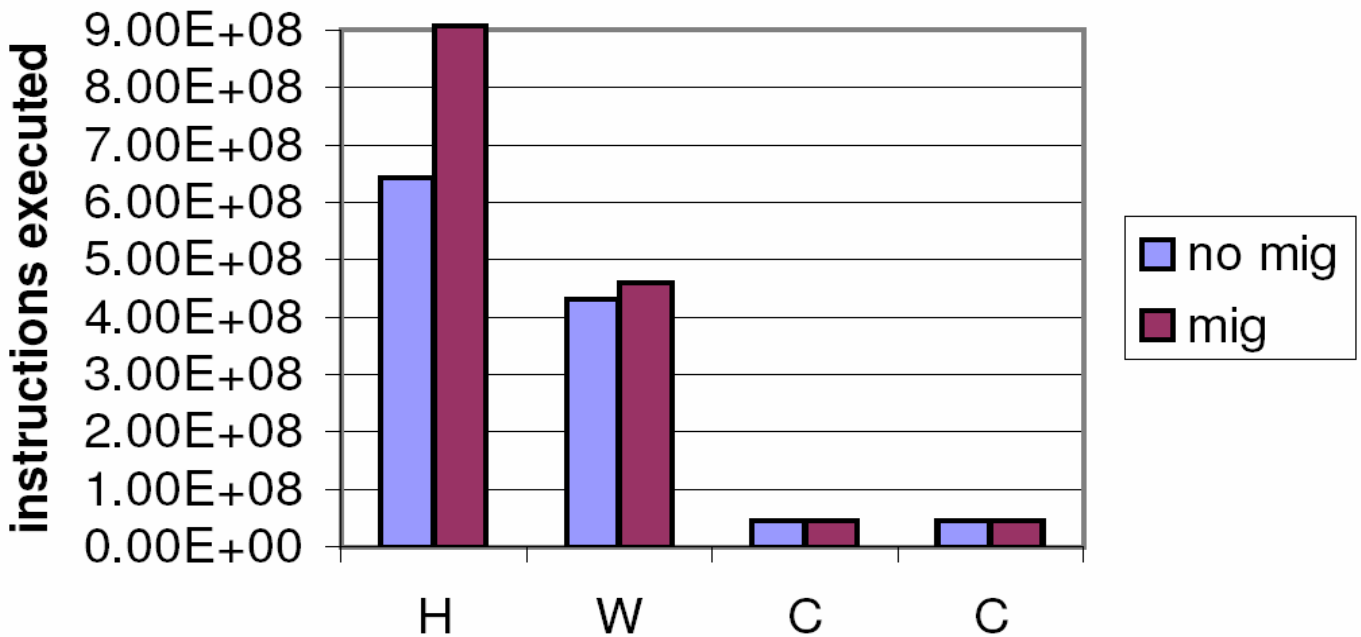
Harness Temperature (2)

- DTM techniques:
  - turn-off clock
  - operate cores at a lower voltage and frequency to ensure correct and/or efficient operation
  - Cost: lower performance

- This work aim: solve the temperature problem with minimal performance loss using Activity Migration (AM)

Activity Migration (1)

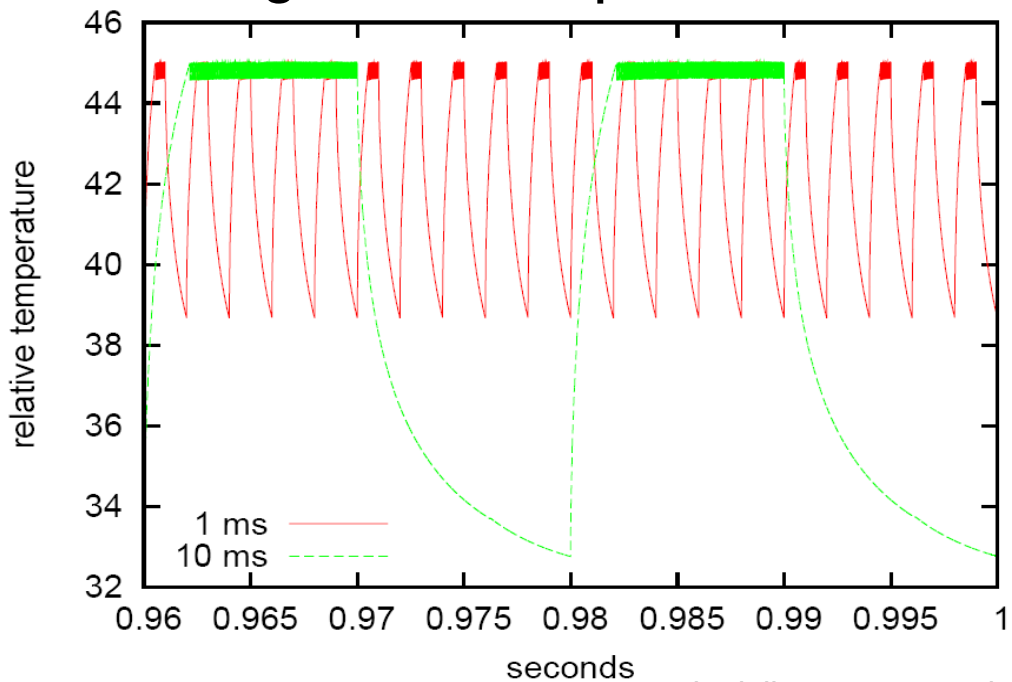
- Leverage multi-cores to distribute activity and temperature over the entire chip
- Basic Idea: transfer thread from a hot core to a cold core (swap threads). Ex. 4 cores and 4 threads*



*A study of Thread Migration in Temperature Constrained Multi-cores P. Michaud and Y. Sazeides, to appear in TACO 2007

Activity Migration (2)

- Prefer AM with minimum performance and design overheads
- OS can implement AM. BUT the shorter the migration period the larger the temperature reduction*



*Scheduling Issues on Thermally Constrained Processors, P. Michaud and Y. Sazeides, IRISA TR 6006, 2006

Activity Migration (3)

- At what layer to implement AM:
  - Shorter OS scheduling periods, or/and
  - Hardware support for faster activity migration

- This talk
  - Present initial results that quantify the performance implications of thread AM on a four core system.
  - Understand the bottlenecks, determine possible ways to minimize AM overhead

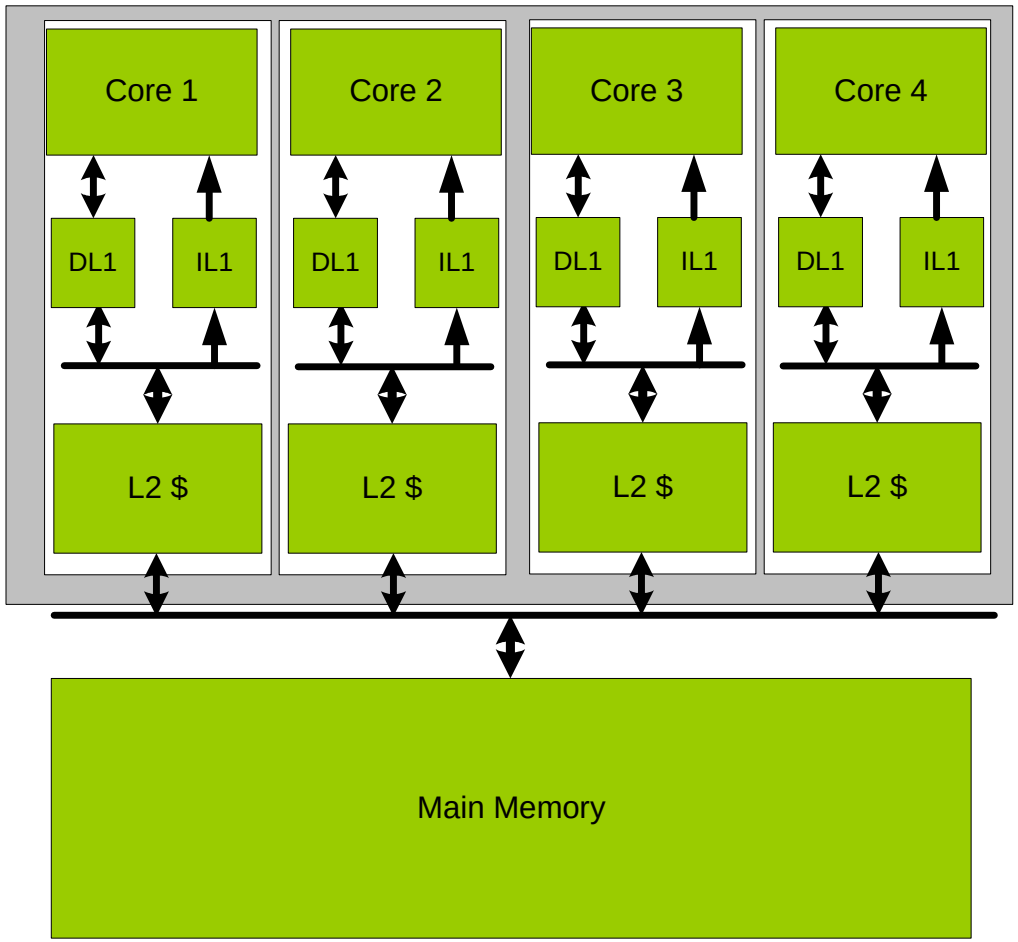
Others on Activity Migration

- Intel introduce term Core-Hopping in 2002
- Hao et al ISLPED2003 proposed the term AM
- Intel in 80-core TSTP uses core-hopping to deal with hotspots, EE Times Jan. 2007

Outline

- Multi-core Architecture and Activity Migration Model
- Experimental Framework
- Results
- Conclusions
- Future Work

Multi-core Architecture-private L1/L2



Activity Migration Model

- Assume fixed migration frequency for all cores
- Every migration for each running thread:
  - Assign a new core to the thread
  - Empty current core pipeline
  - Flush DL1 and L2 to maintain coherence
    - Simple and possibly low performing
    - When more than 1 thread running all cores will try to flush their L2 simultaneously
  - Predictors state not flushed
  - Transfer register to new core (PC, registers)
  - When new core ready resume execution

Activity Migration Time Overhead

- Direct:

  empty pipe + flush caches + transfer register + wait for new core to finish flushing

- Indirect Cold Effects:
  - cold caches
  - stale predictor state OR other thread(s) updating predictor

AM Design Space Explored

- Workload size: 1-3 threads
- Number of cores: 4
- Migration frequency: 1ms, 0.1ms, 0.01ms
- Effect of cold/warm L1/L2 caches and cold/warm branch predictor
  - Caches Warm — Predictor Warm — (Baseline no AM)
  - Caches Warm — Predictor Cold — (predictor implicat.)
  - Caches Cold — Predictor Warm — (cache implications)
  - Caches Cold — Predictor Drowsy — Activity Migration
- AM patterns:
  - Round-robin (RR) over all cores
  - Swaps (SWAP) between two cores (for two-thread workload)

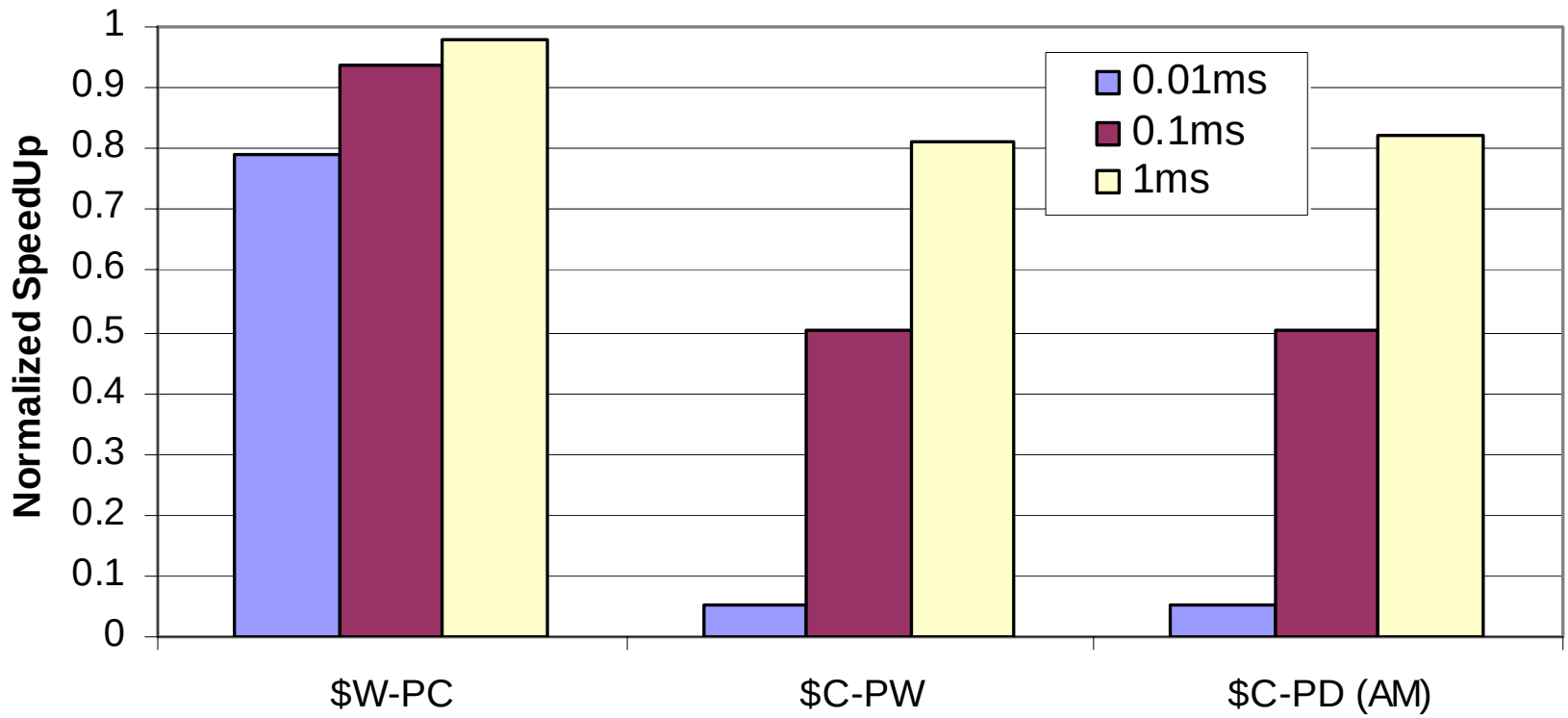
Experimental Framework

- Simulator
  - Core based on a modified version of sim-alpha
  - Multi-core simulator is a parallel program
    - Using pthreads (Donald and Martonosi CAL2006)
- Multi-core Configuration
  - OOO Core:
    - 4-way superscalar
    - 15 pipeline stages
    - 3GHz frequency
    - Icache/Dcache: 64B block, 64KB, 2way, 1/3 cycles
    - L2: 2MB, 4-way, 10 cycles access latency
    - Predictor: 16KB combining predictor (bimodal-gshare)
  - Shared memory bus
    - 8 Bytes wide, 500MHz
    - Split transaction bus between L2 and memory
    - arbitration based on OS thread scheduling order
    - 200 cycles minimum memory latency

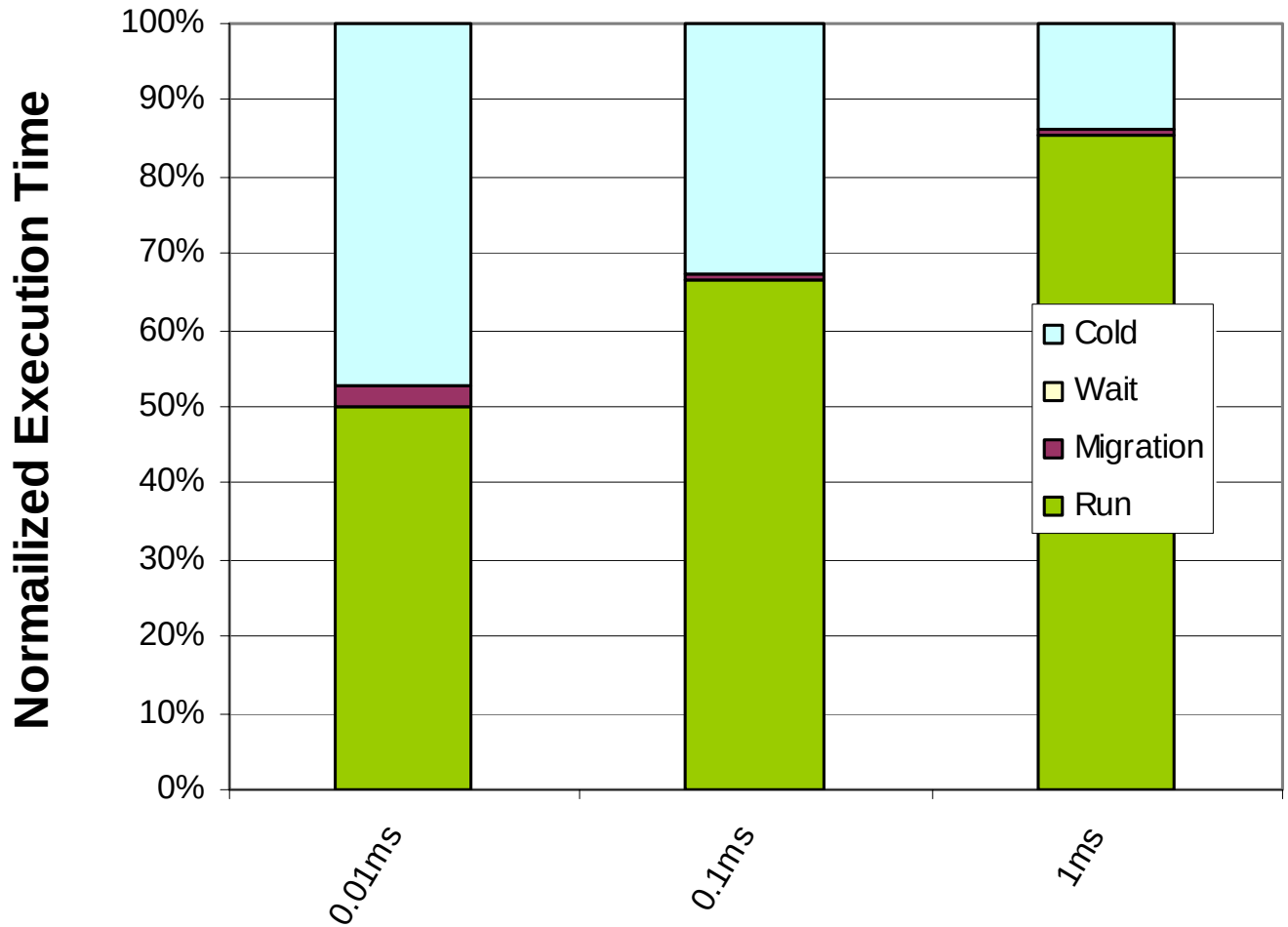
Benchmarks, Methodology and Metrics

- SPEC CPU2000
- Fast forward 1billion instructions, executing until the slowest thread commits at least 100million instructions
- Performance Metric:
  - How faster is multi-core run over the same workload when run sequentially
  - Normalized to the Speedup of CMP baseline with no AM
- Assumption: register transfer latency for AM has no latency and no side affects on memory

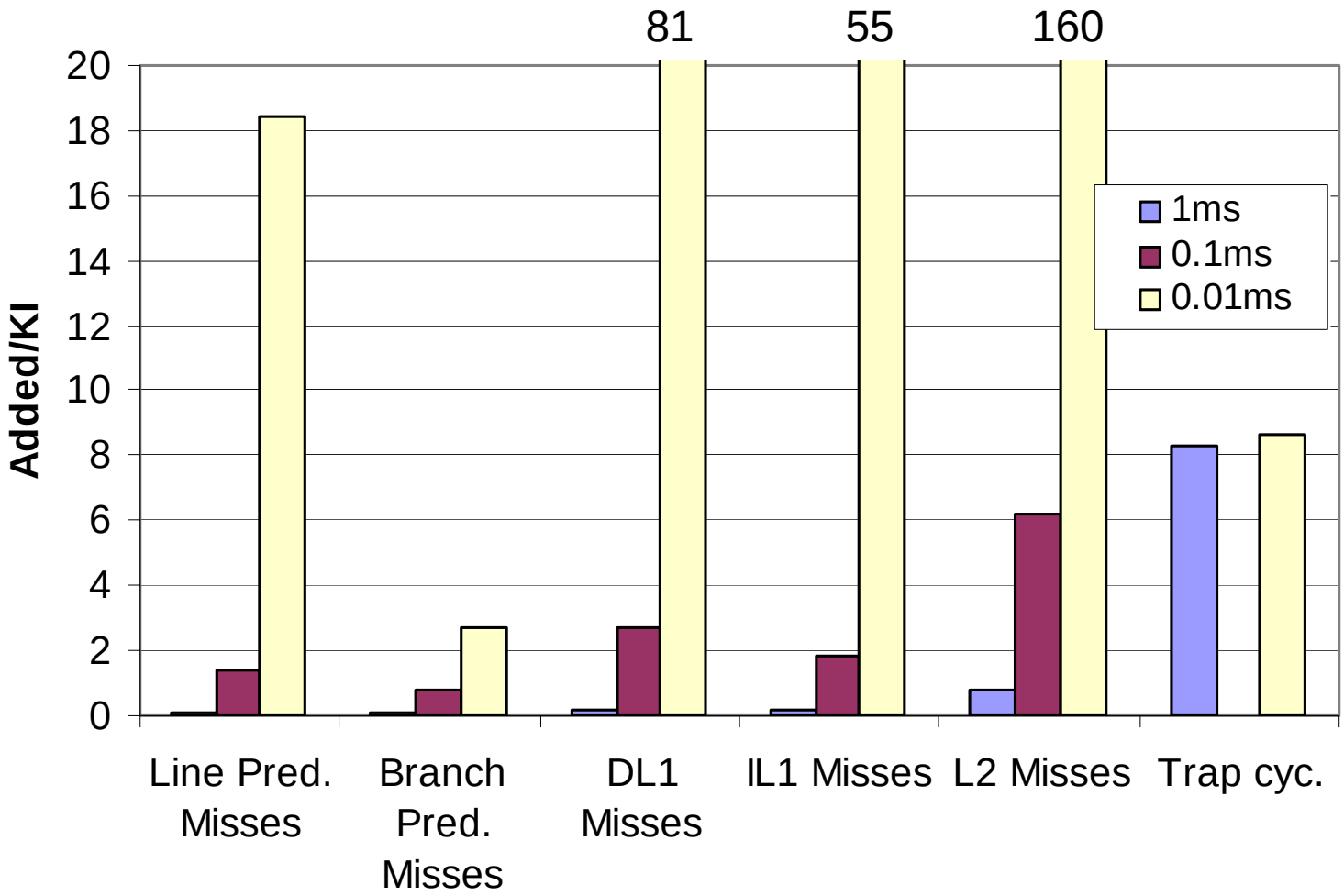
Single Thread crafty(AM:RR 4 cores)



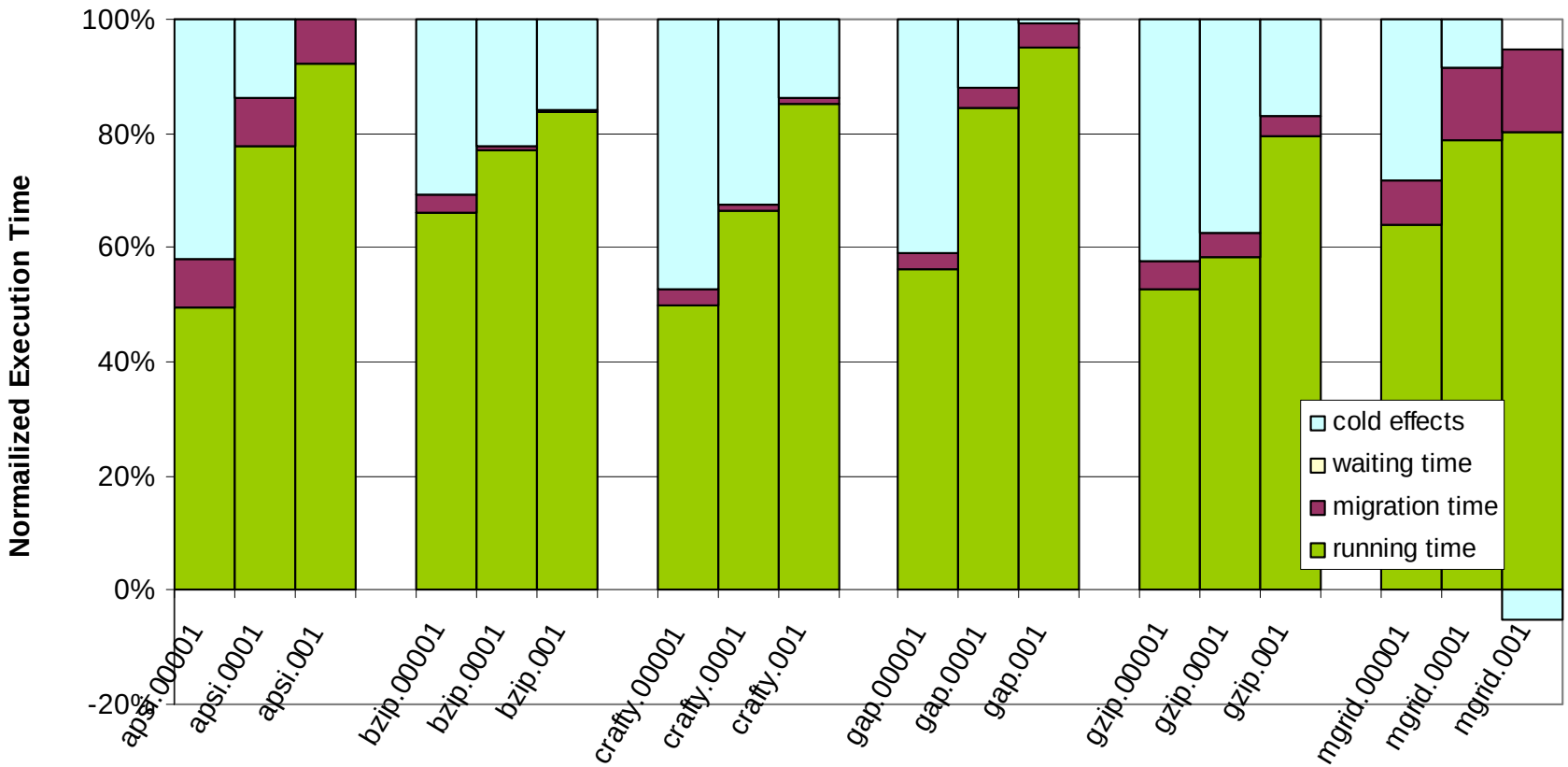
Single Thread crafty: $C-PD (AM)



Single Thread crafty: $C-PD (AM)



Various Single Threads: $C-PD (AM)



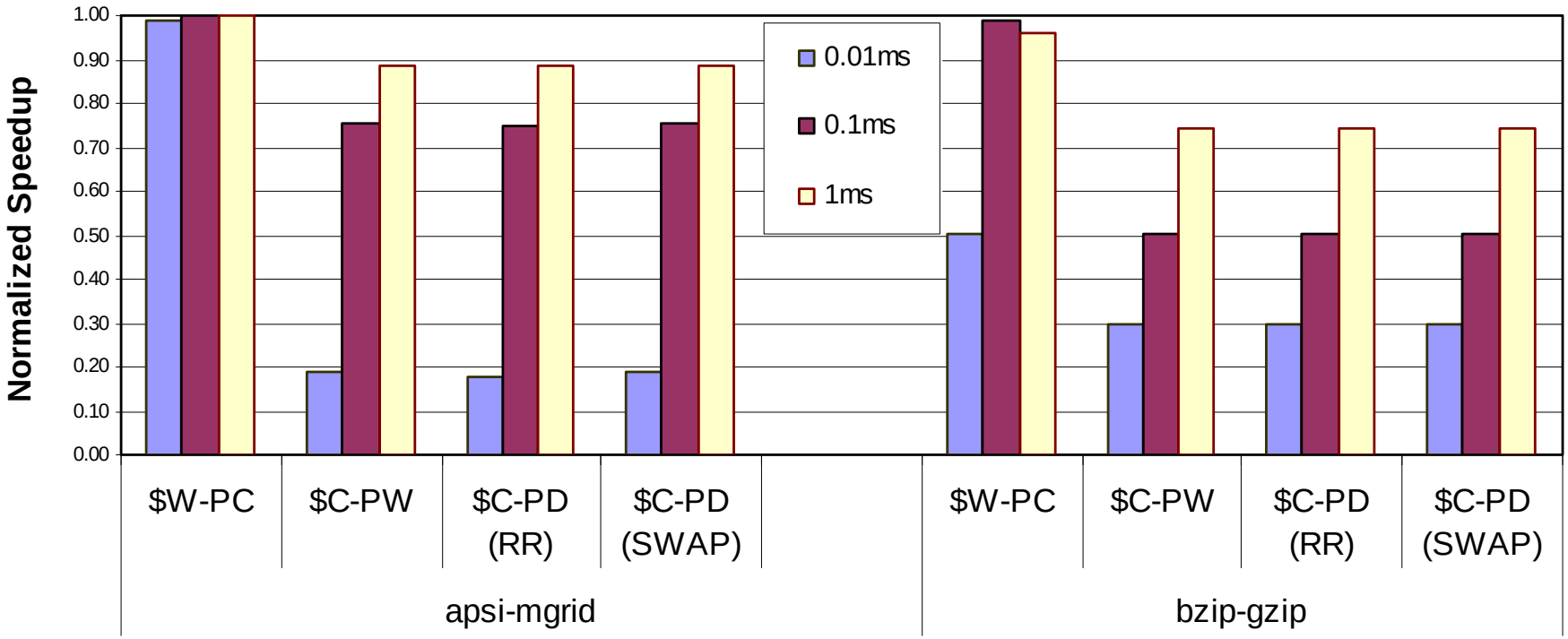
Observations for 1-thread AM

- Performance loss is correlated with migration period
- Cold effects usually dominate
  - Flushing is not slow
- Both L2 cache and branch predictor are critical resources to have warm
  - Drowsy predictor sufficient

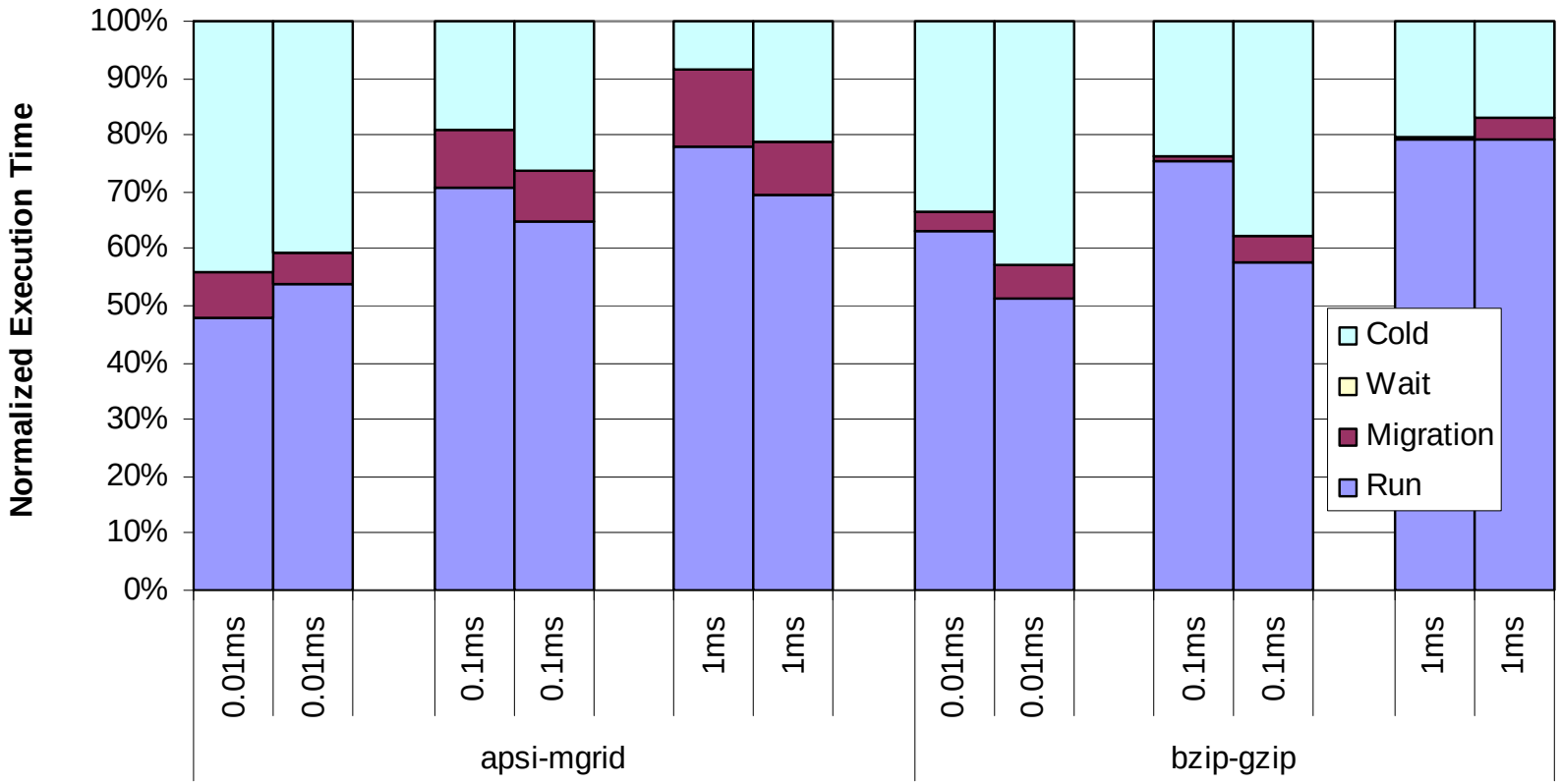
Size, type of resource and working set

- Cold effects more likely for programs with large footprint in predictor and L2
  - Size of resources in blocks
    - 64KB IL1 cache: 1024 blocks
    - 64KB DL1 cache: 1024 blocks
    - 16KB Branch Predictor
      - 64K blocks (Entries)
      - Hybrid bimodal/gshare
    - 2MB L2 Cache: 64K blocks
      - long latency

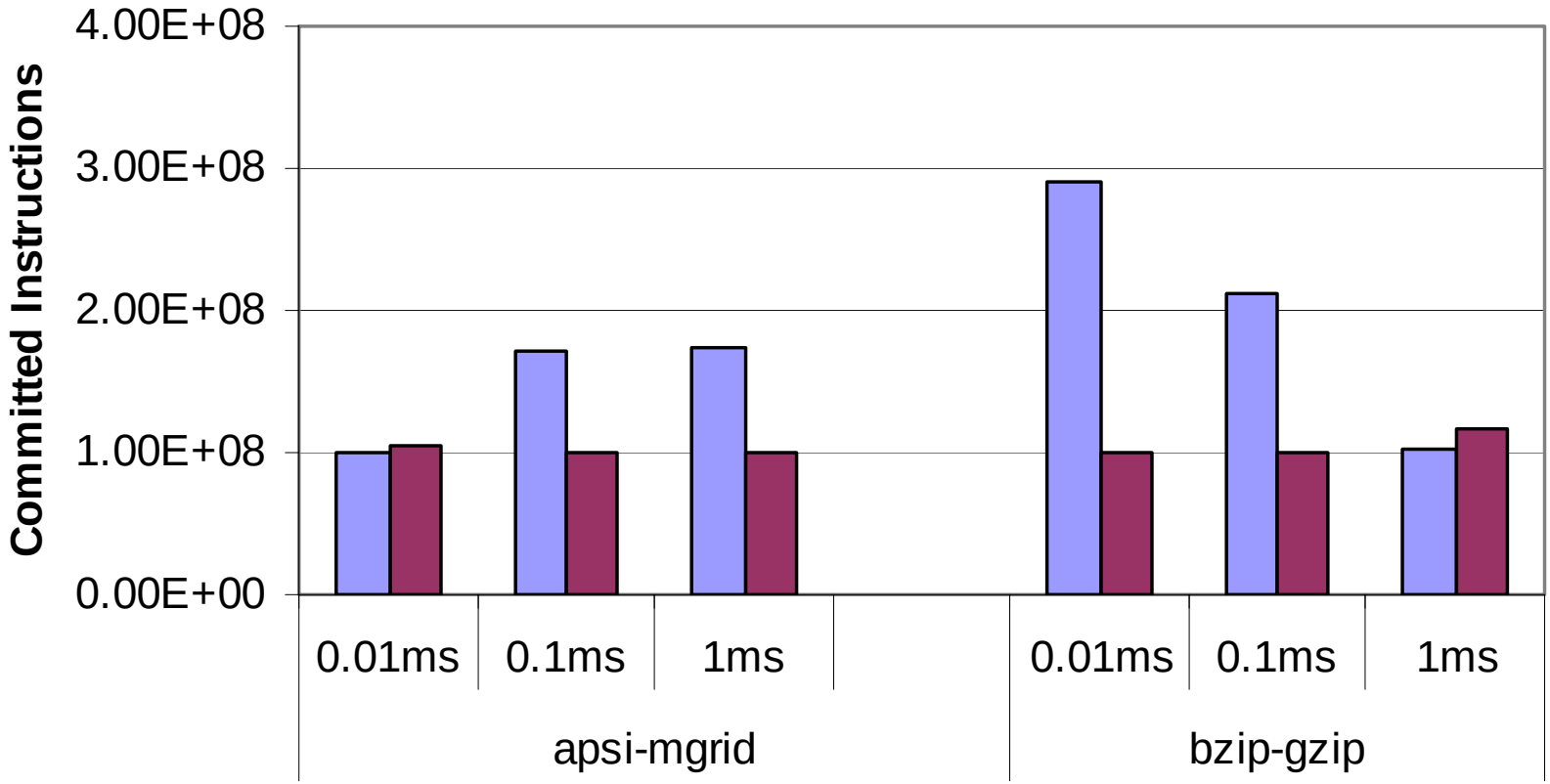
Two Thread(RR 4-cores / swap 2-cores)



Two Thread (AM RR 4 cores)



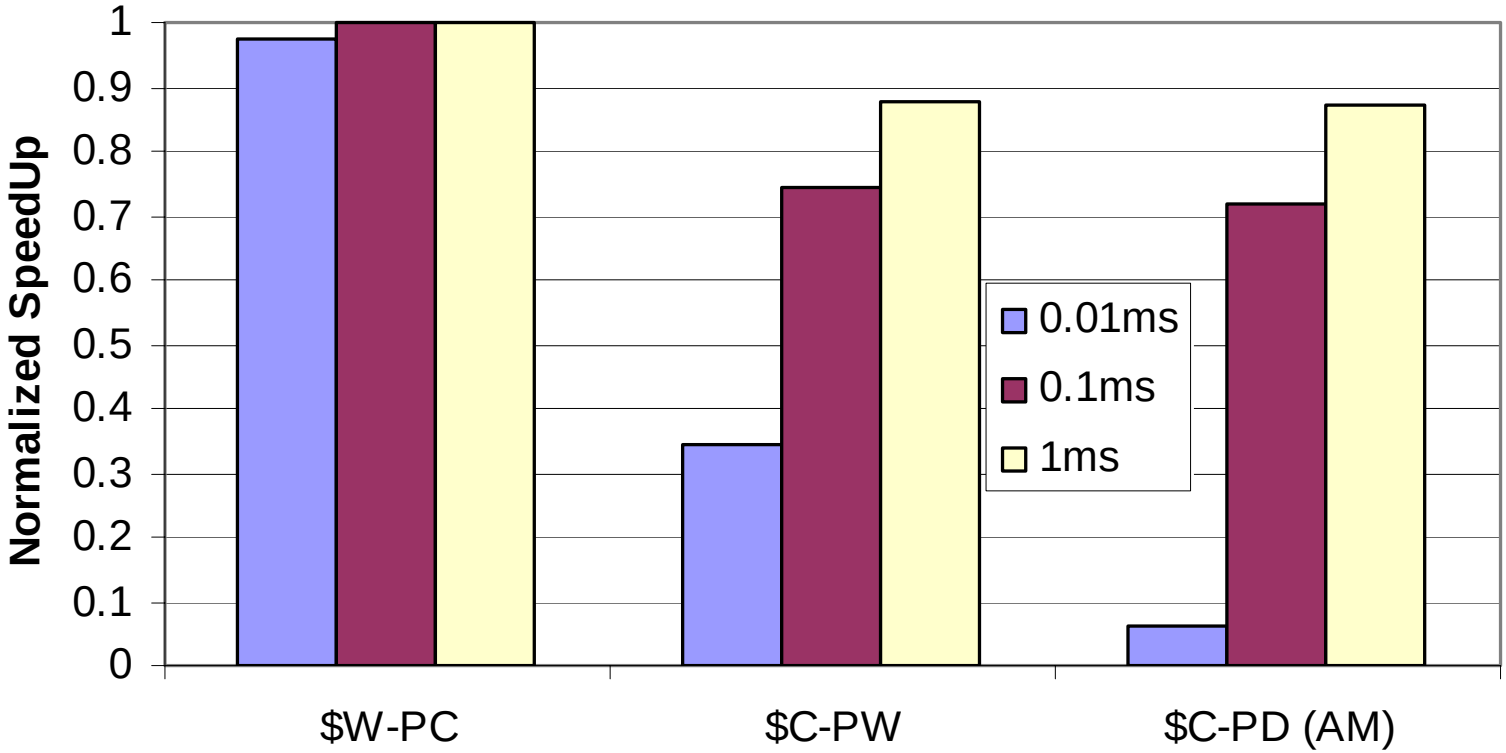
Possible Issue: non-uniform contribution of threads



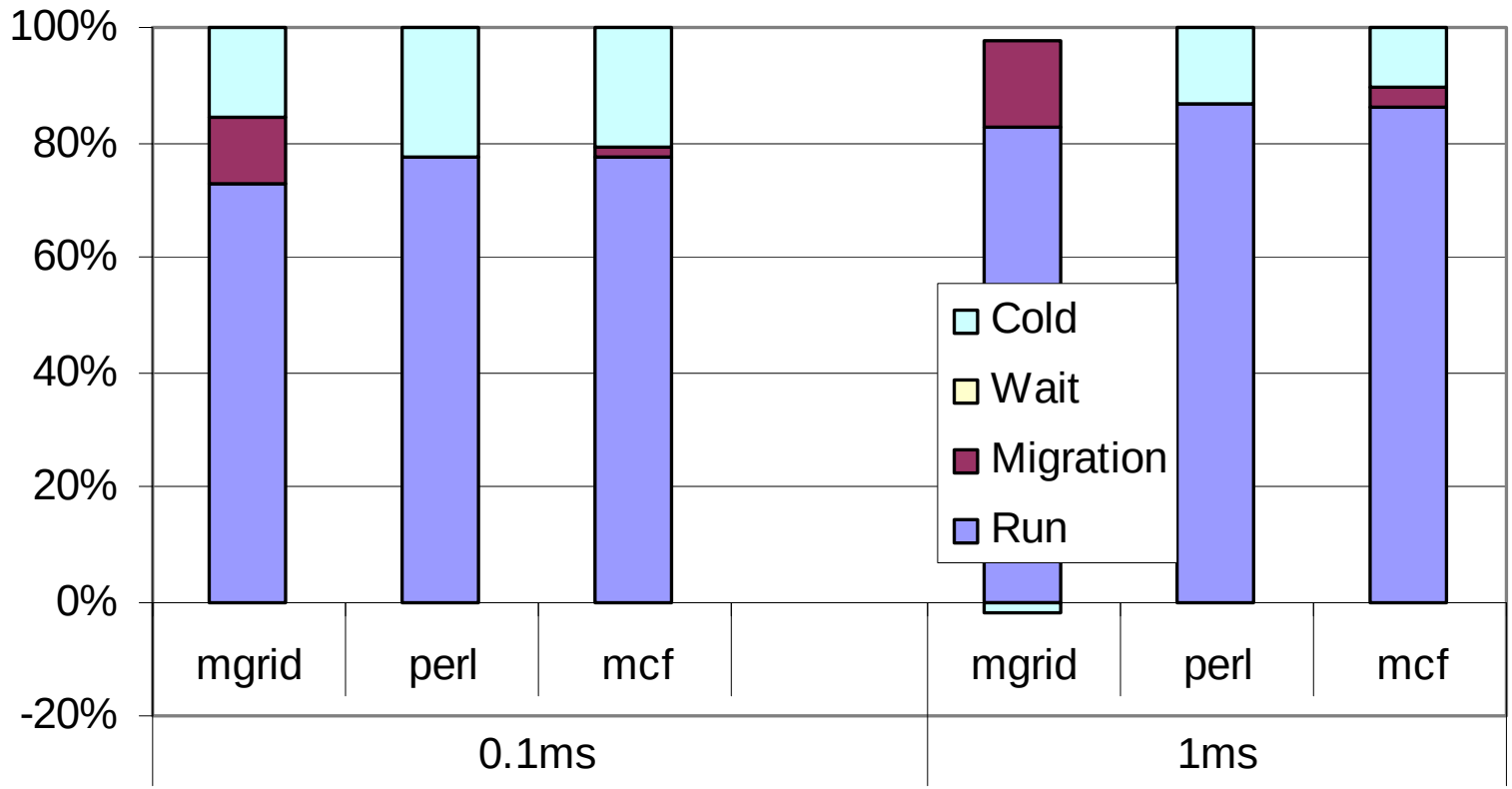
Observations for 2-thread AM

- Performance loss is correlated with migration period
- Round-robin not worse than swap
  - Good news for temperature
- Both cold effects and Migration important
  - More contention on bus due to flushing L2 caches at the same time from two threads
- L2 cache critical resource
- Drowsy predictor works
  - Minimal interference across threads
- Issue with non-uniform contribution across AM periods
  - Metric and Scheduling

Three Thread (AM RR 4 cores)
mgrid.perl.mcf



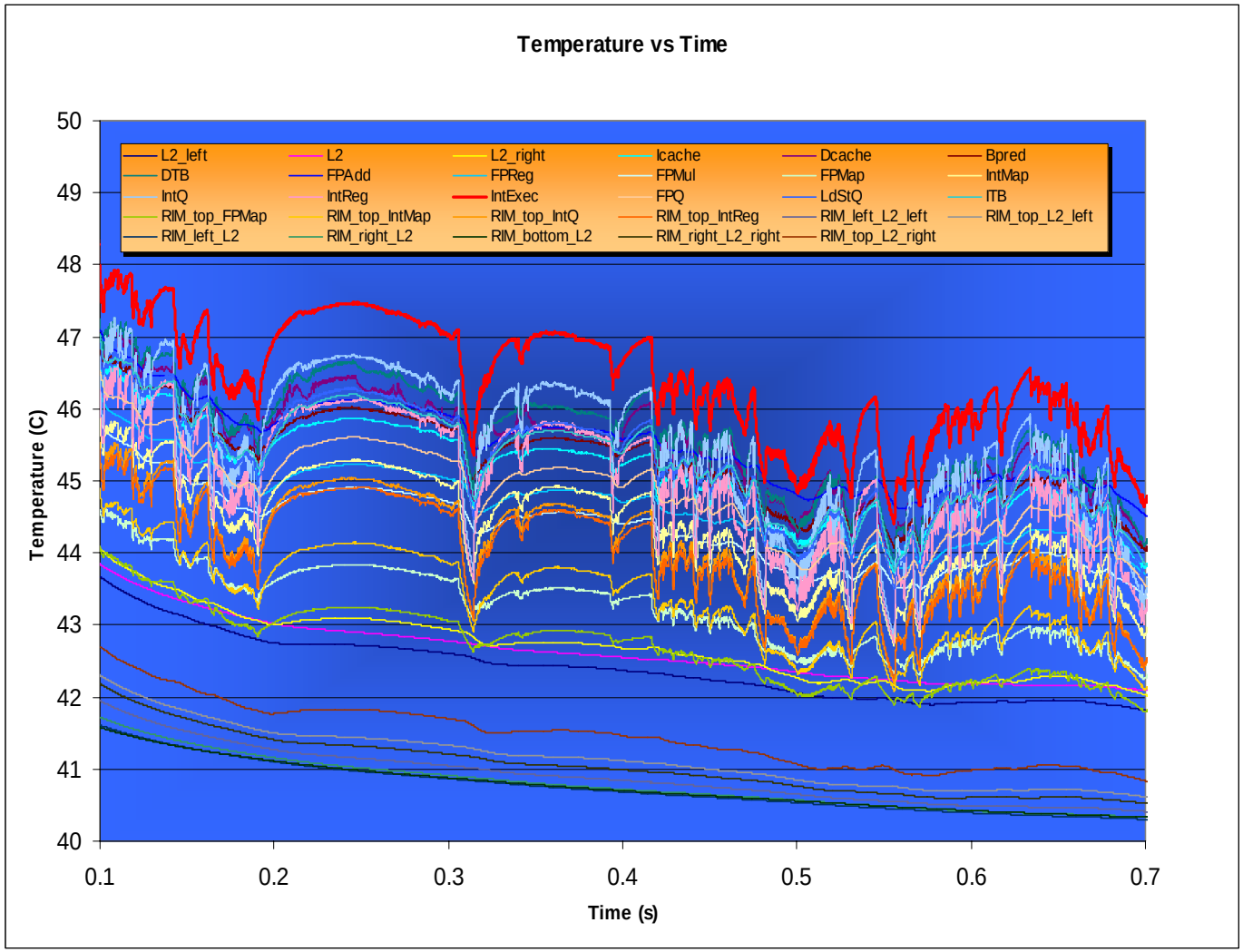
Three Thread Distr.(AM RR 4 cores)



Conclusions

- Simplistic implementation sufficient for some cases BUT definitely not for all
  - Need for low overhead coherence
  - Drowsy predictor same performance as warm
    - Little interference across benches
  - Swap and round-robin same behavior
  - More pronounced effects for benchmarks that use shared resources: bus

Ongoing Work: Power Modelling



Future Work

- Develop and evaluate low overhead migration techniques for cache coherence
- Integrate power/temperature model and evaluate sensor based migration
- Build synthetic simulator for exploring scheduling issues

Acknowledgements

- Funding for this work:
  - University of Cyprus
  - Intel
  - Cyprus Research Promotion Foundation
  - HiPEAC
- T. Constantinou for initial Single Thread Migration studies

Questions?