A Comprehensive Analysis of OpenMP Applications on Dual-Core Intel Xeon SMPs

Ryan E. Grant and Ahmad Afsahi

Department of Electrical and Computer Engineering

Queen's University

Kingston, ON, Canada

2007 Workshop on Multi-Threaded Architectures and Applications

March 30, 2007

Presentation Outline

- Introduction
- Experimental Framework
- Multithreaded, Single-Program Results
- Multithreaded, Multi-Program Results
- Work in Progress
- Conclusions and Future Work

Performance and Power

- High-performance computing has always been concerned with performance, and occasionally price/performance. Such systems
  - Tackle high-end applications known as Grand Challenge problems
  - Provide services such as search engines, data mining, eCommerce, web hosting, digital libraries

- However, the push toward maximum performance has resulted in significant power consumption and excessive heat generation.
  - This has led the industry to develop aggressive Chip Multithreading (CMT) processors for general-purpose applications such as Intel Itanium 2, Intel Xeon, IBM Power5, Sun UltraSPARC T1 etc.

Chip-Multithreading

- CMT: combine Chip Multiprocessing (CMT) and Simultaneous Multithreading (SMT) to achieve performance through TLP.
  - CMPs: contain multiple cores allowing more than one thread to be executed at a time. Each core has its own resources as well as shared resources.
  - SMT: is a technique that allows multiple independent threads to execute different instructions each cycle.

- Hyper-Threading (HT): is an implementation of SMT found in Intel processors such as the Xeon EM64T.
  - An HT-enabled processor appears as two logical processors to the operating system, where each processor maintains a separate run queue.
  - HT-enabled logical processors share many hardware resources such as cache, execution units, TLBs, branch prediction unit, and load/store buffers.

Motivation

- Hybrid CMT-based SMPs (having up to three levels of memory hierarchy) present new challenges as well as new opportunities to maximize performance provided the resources available could be used efficiently. In reality, the increased thread load may result in:
  - heavy demand on cache subsystem, growing number of bus transactions, more stall cycles etc.

- Objectives:
  - Discover optimal configurations of such commercial systems for scientific, multithreaded applications written in OpenMP (OpenMP was originally designed for flat systems).
  - Identify the resources that might become a bottleneck to performance under different hardware configurations.

➔This work is the first step towards devising optimal schedulers to improve the performance of multithreaded applications running on emerging multi-core systems.

Presentation Outline

- Introduction
- Experimental Framework
- Multithreaded, Single-Program Results
- Multithreaded, Multi-Program Results
- Work in Progress
- Conclusions and Future Work

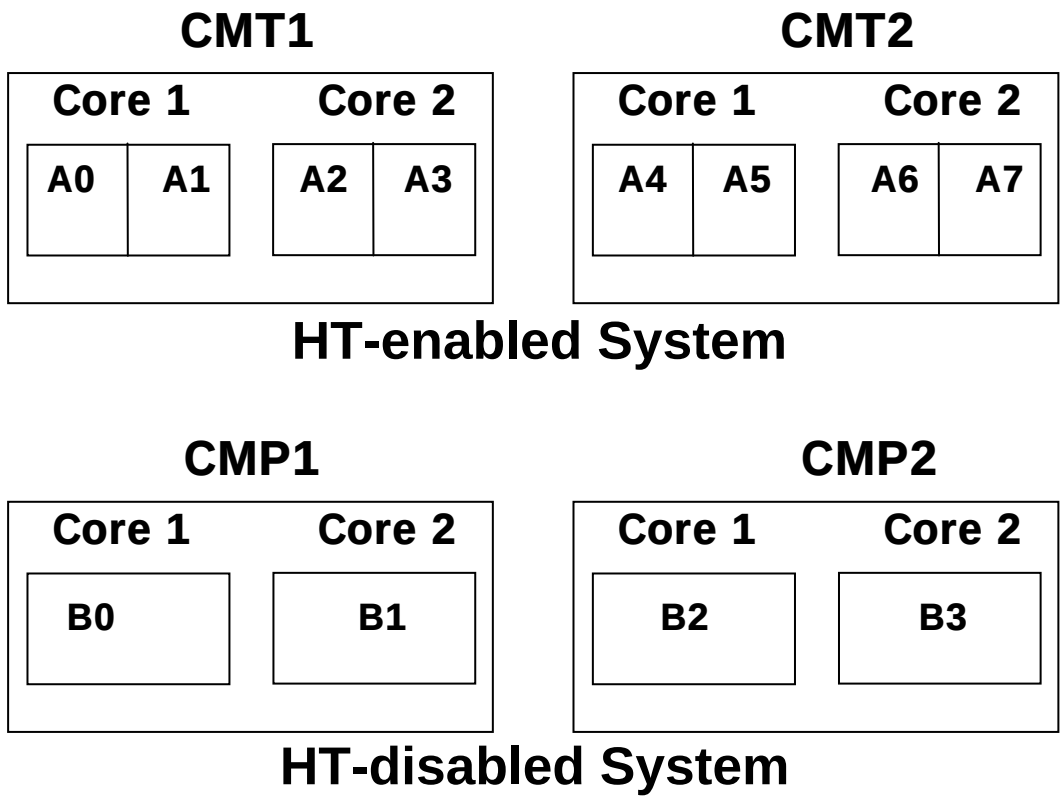
Platform

	Dell PowerEdge 2850
Processors	Two dual-core 2.8GHz Intel Xeon EM64T • 12KB shared execution trace cache and 16KB shared data cache on each core. • 2MB L2 cache per core
Main Memory (MM)	4GB of DDR-2 SDRAM on an 800MHz Front Side Bus.
Memory latency	L1: 1.43ns; L2: 10.61ns; MM: 136.85ns
Memory bandwidth	Threads on a single physical chip: • Read: 3.57GB/s; Write: 1.77GB/s Threads spread out to both physical chips: • Read: 4.43GB/s; Write: 1.61GB/s
Operating system	Red Hat Enterprise WS 4.1 with kernel 2.6.9-11. Used *maxcpus = X* boot option to only initialize and use *X* logical processors

Terminology

Configuration information

Terminology	H/W Contexts	Architecture
Serial	B0	Serial
HT_{on} -2-1	A0, A1	SMT
HT_{off} -2-1	B0, B1	CMP
HT_{on} -4-1	A0, A1, A2, A3	CMT
HT_{off} -2-2	B0, B2	SMP
HT_{on} -4-2	A0, A1, A4, A5	SMT-based SMP
HT_{off} -4-2	B0, B1, B2, B3	CMP-based SMP
HT_{on} -8-2	A0, A1, A2, A3, A4, A5, A6, A7	CMT-based SMP



System configuration labeling

Application Benchmarks

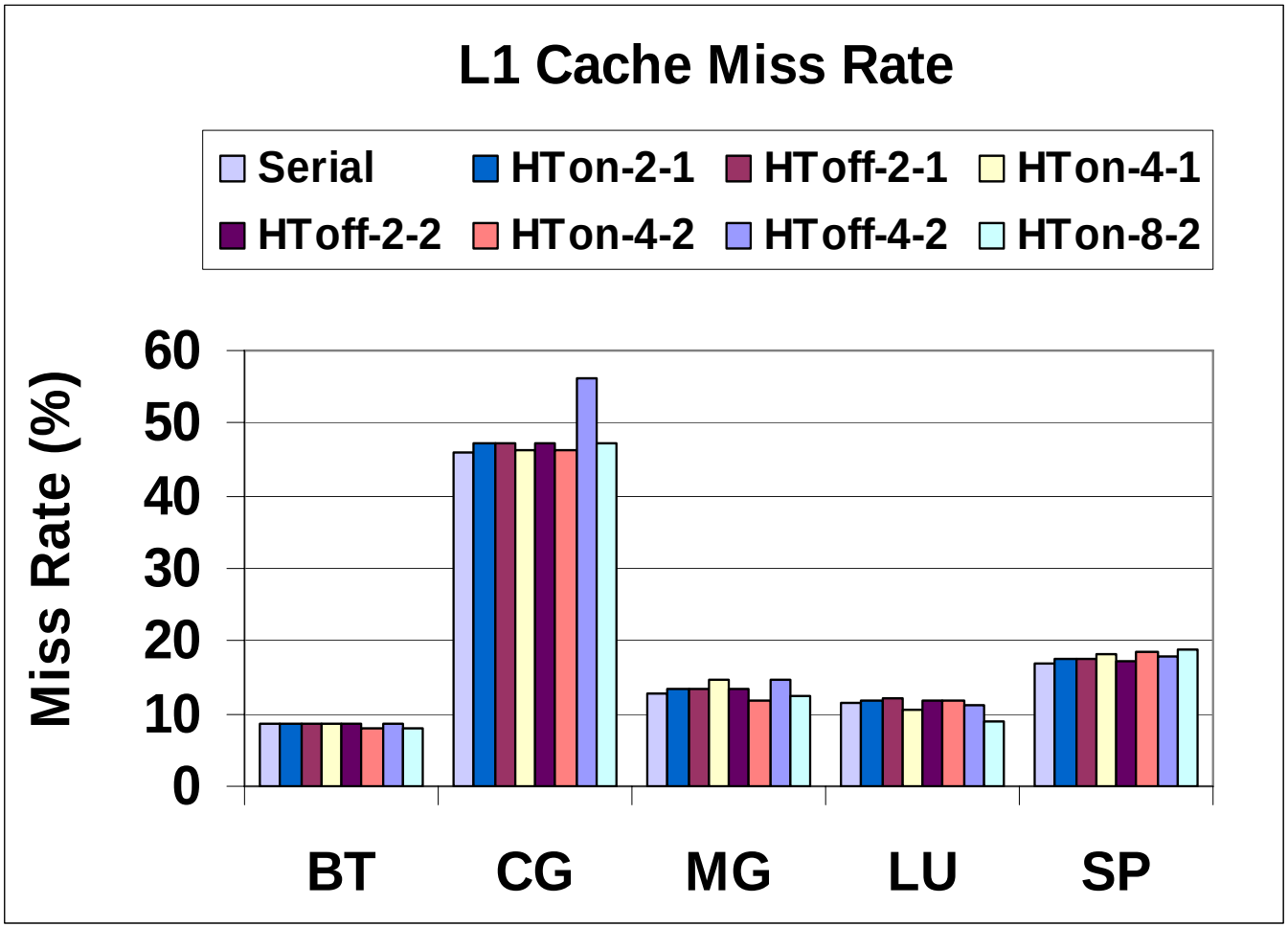
- NAS OpenMP 3.2 parallel benchmark suite:
  - Class B (large enough to provide realistic results, while ensuring the working set fits in memory)
  - Three kernel applications: CG, MG, FT
  - Three CFD applications: BT, SP, LU
- Intel Vtune Performance Analyzer 7.2: is used to gather
  - Cache performance (L1, L2, Trace cache)
  - TLB miss rates
  - Stalled CPU cycles
  - Branch prediction rate
  - Bus transactions
  - Overall CPI
- Intel compilers 8.1 (with default compiler flags) were used in building the applications.

Presentation Outline

- Introduction
- Experimental Framework
- Multithreaded, Single-Program Results
- Multithreaded, Multi-Program Results
- Work in Progress
- Conclusions and Future Work

Multithreaded, Single-Program Results

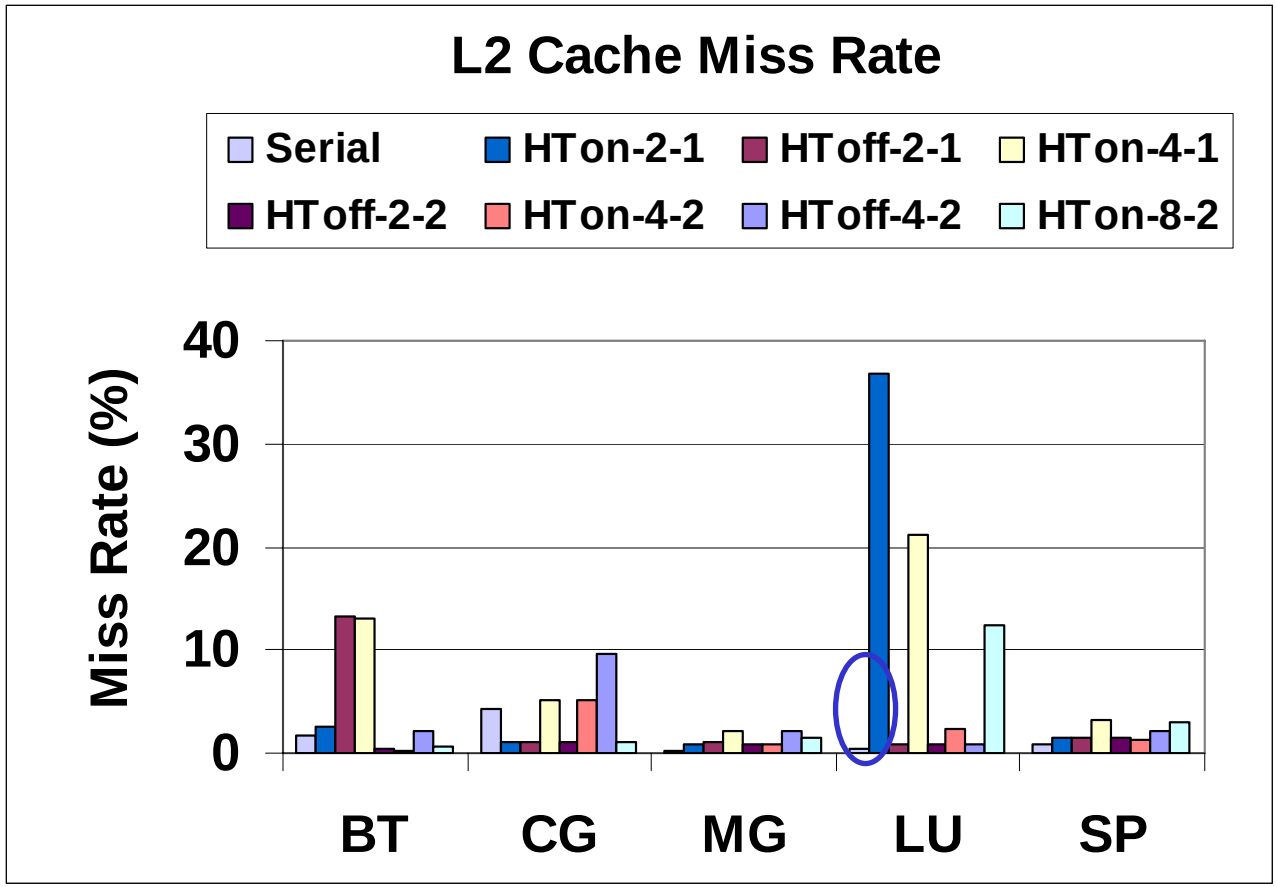
- **L1 Cache Performance**



Application benchmarks mostly use a large number of infrequently changing variables, and only a small number of new variables in each loop.

Multithreaded, Single-Program Results

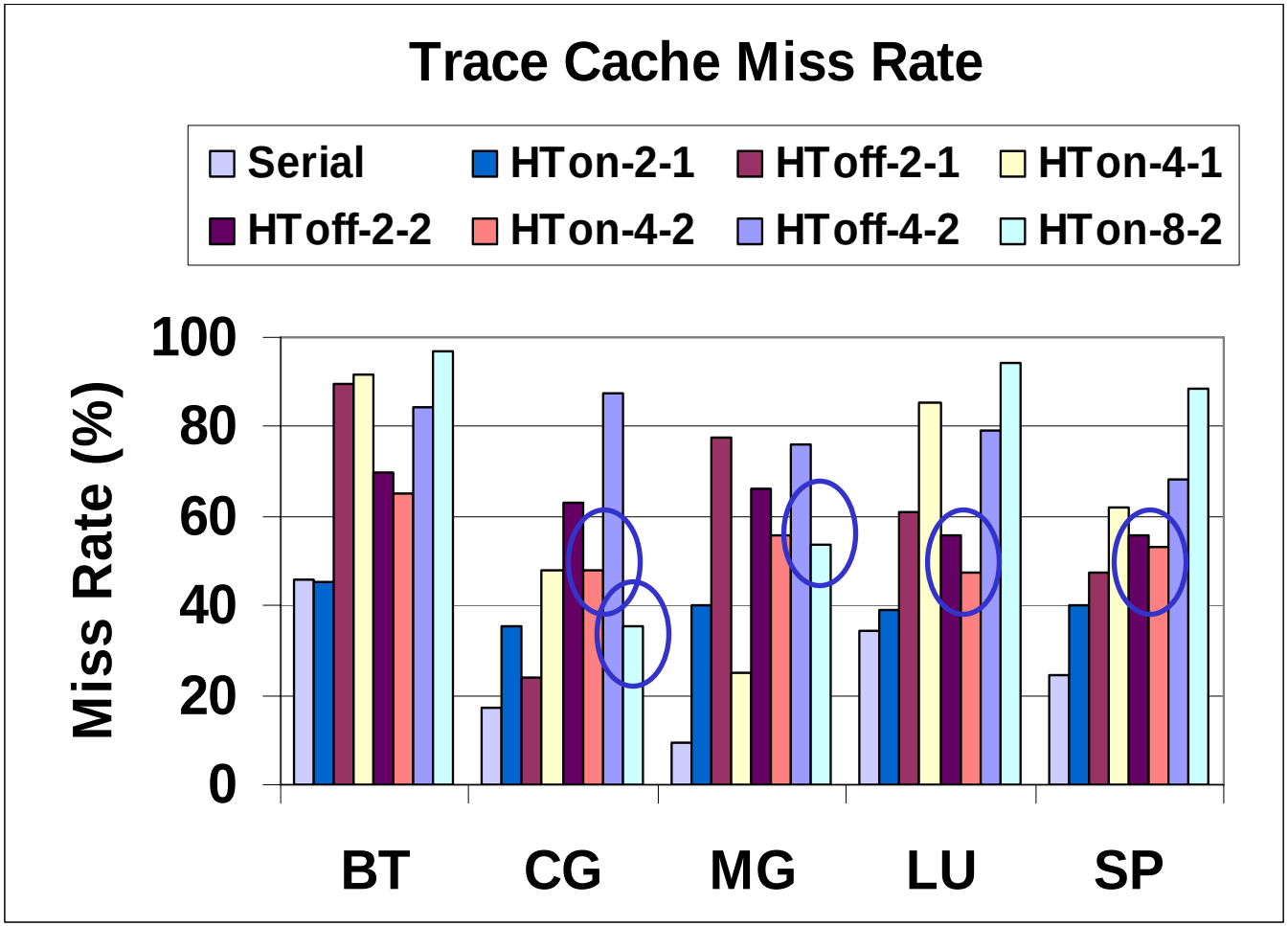
- **L2 Cache Performance**



The second group is similar to the third group in trends in L2 performance, with the HT_{on} configurations mostly having a higher miss rate than the HT_{off} configurations. This is not unexpected as the HT_{on} configurations have less memory available to the system per execution thread.

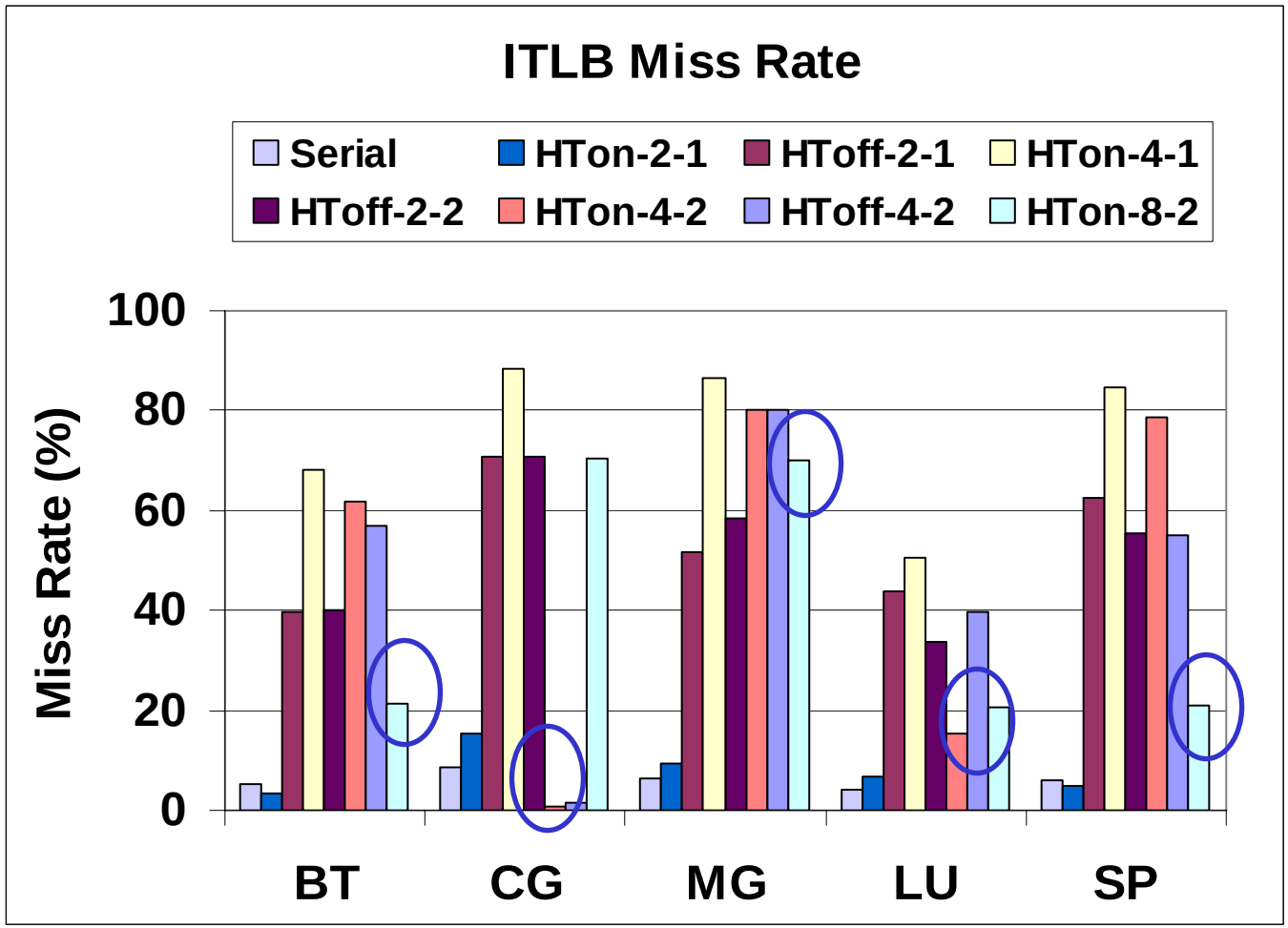
Multithreaded, Single-Program Results

- **Trace Cache Performance**



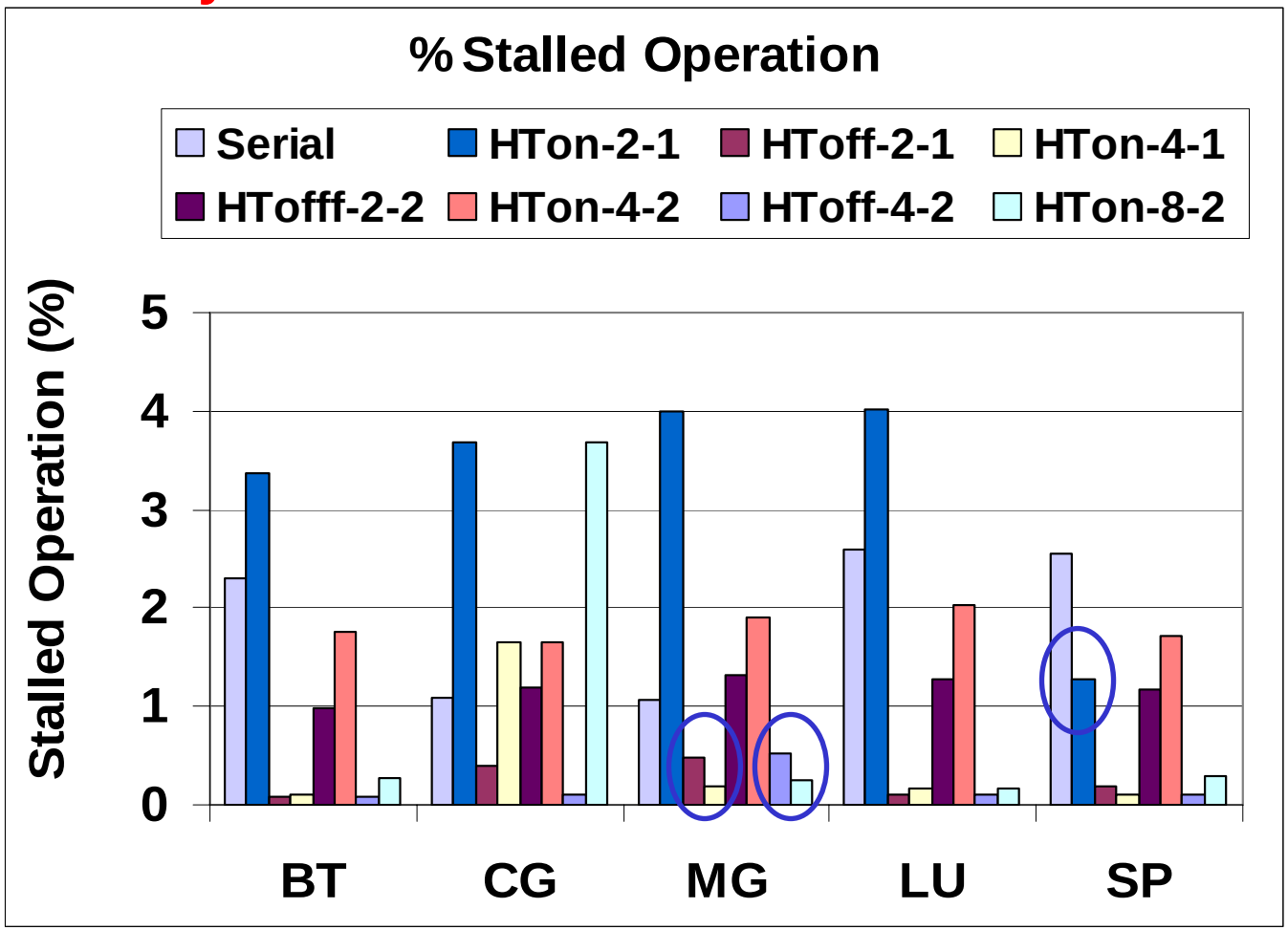
Multithreaded, Single-Program Results

- **ITLB Performance**



Multithreaded, Single-Program Results

- **Stall Cycles**

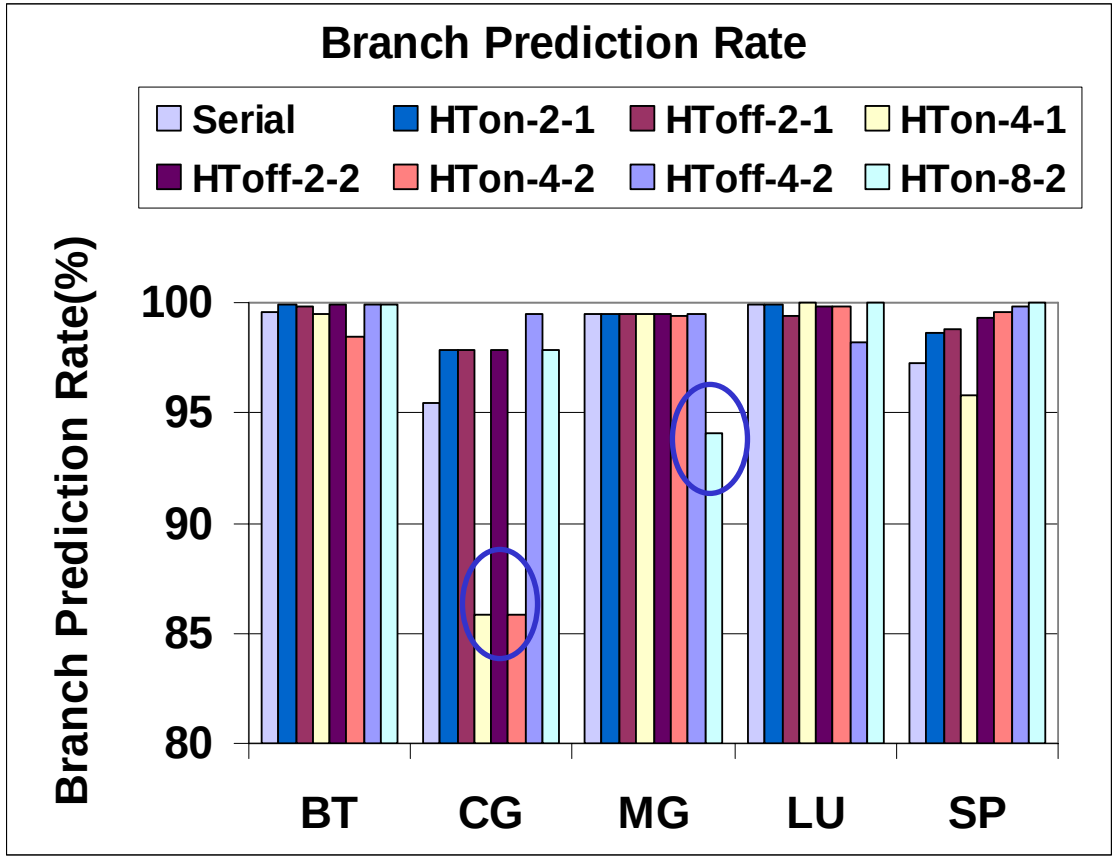


Groups 2, 3 and 4 show similar patterns with the HT_{on} configurations having more stalled cycles than the HT_{off} configurations. This is an indication of thread contention for shared resources in the cores.

The number of cycles spent in a stalled state for the HT_{on}-2-1 configuration is poor relative to the other configuration groups.

Multithreaded, Single-Program Results

- **Branch Prediction**

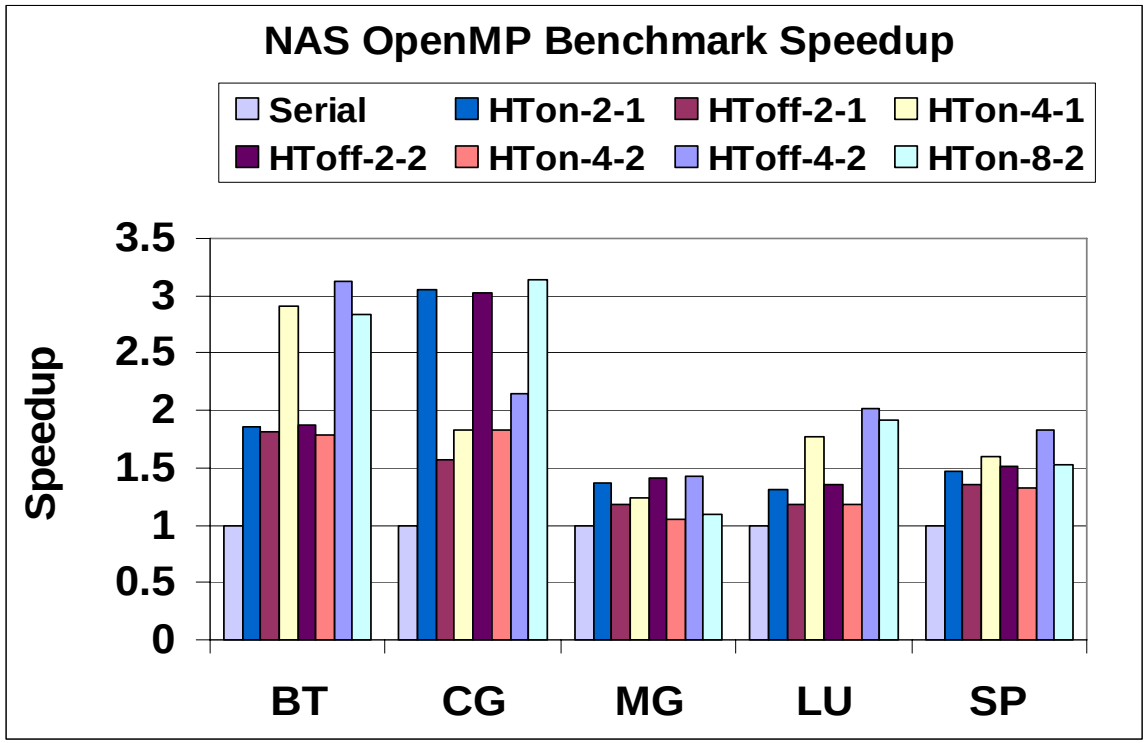


Branch prediction rates are excellent with the exception of the HT_{on} configurations from groups 2 and 3 for CG (and HT_{on}-8-2 for MG). This helps to explain the number of stalled cycles and consequently the high CPI that the HT_{on} configurations in groups 2 and 3 have for the CG benchmark.

Multithreaded, Single-Program Results

- **Wall Clock Performance**

Speedup for NAS OpenMP applications



Average speedup for architectures across all application benchmarks

SMT HTon-2-1	CMP HToff-2-1	CMT HTon-4-1	SMP HToff-2-2	SMT based SMP HTon-4-2	CMP based SMP HToff-4-2	CMT based SMP HToff-8-2
1.81	1.42	1.87	1.83	1.43	2.11	2.10

Multithreaded, Single-Program Results

- **Observations**

1. The use of SMT on a CMT chip can be beneficial to the performance of a system.
   - In this case, the overall performance of a single CMT chip is comparable (13.6% slowdown) to the performance of two dual-core processors operating with HT_{off} (CMP-based SMP) despite having half as many available computational resources.

2. When overall processor resources are increased to utilize both dual-core processors with HT enabled, the overall effect reduces computational speed and results in a slowdown of approximately 6.7% versus HT disabled.
   - We can conclude that HT is of benefit when enabled for smaller numbers of logical processors (<4). However, the efficiency of HT with fewer physical processors has increased from previous observations most likely due to the improvements in memory bus speed.

- Introduction
- Experimental Framework
- Multithreaded, Single-Program Results
- Multithreaded, Multi-Program Results
- Work in Progress
- Conclusions and Future Work

Multithreaded, Multi-Program Results

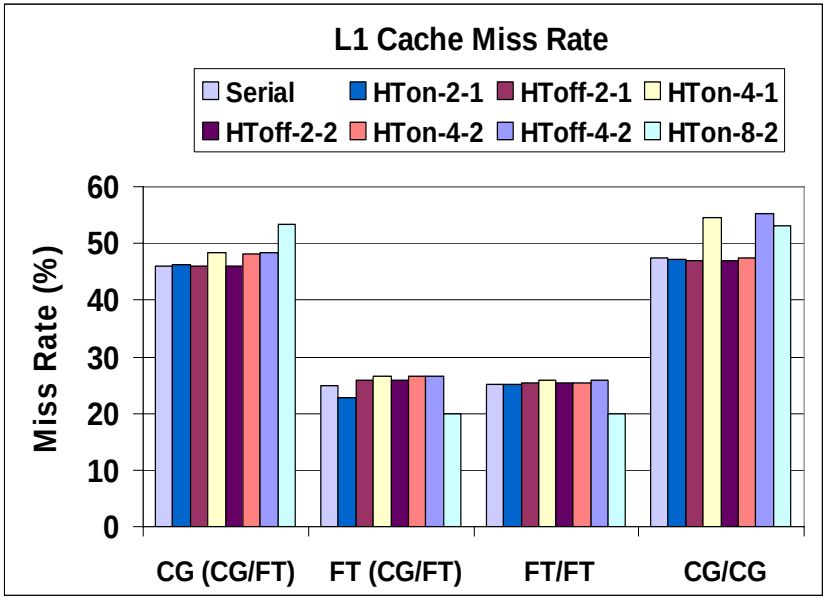
- In this section, we utilize more than one concurrent program execution at a time to examine the ability of the architectures to handle complimentary and uncomplimentary workloads of multiple programs.
  - FT benchmark: is a Fourier transform benchmark requiring mostly computational resources and limited memory resources
  - CG benchmark: requires significant memory resources

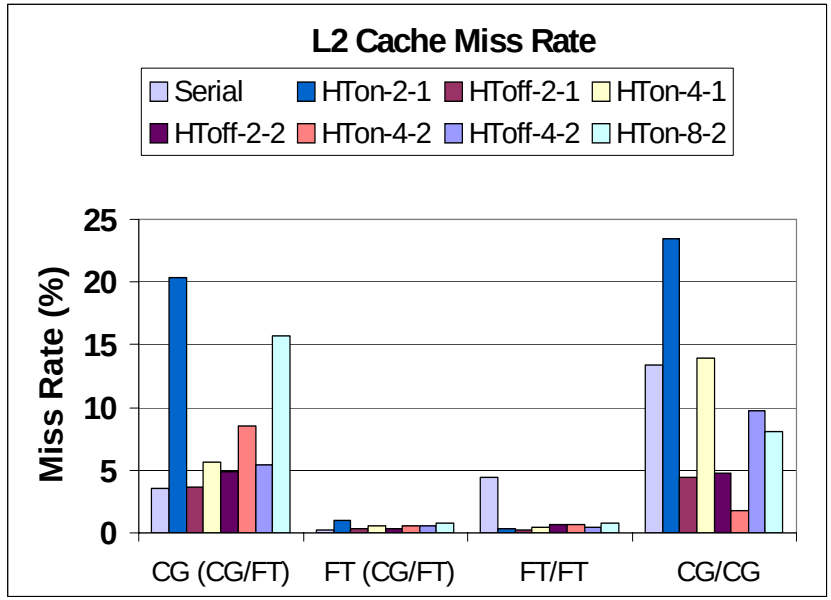
- Three separate tests were conducted:
  - One with two copies of CG,
  - One with two copies of FT, and
  - One with one copy of CG and one copy of FT

- The maximum number of execution threads available to each system configuration was used, with the threads being distributed evenly between the executing programs.

Multithreaded, Multi-Program Results

- **L1 and L2 Cache Performance**



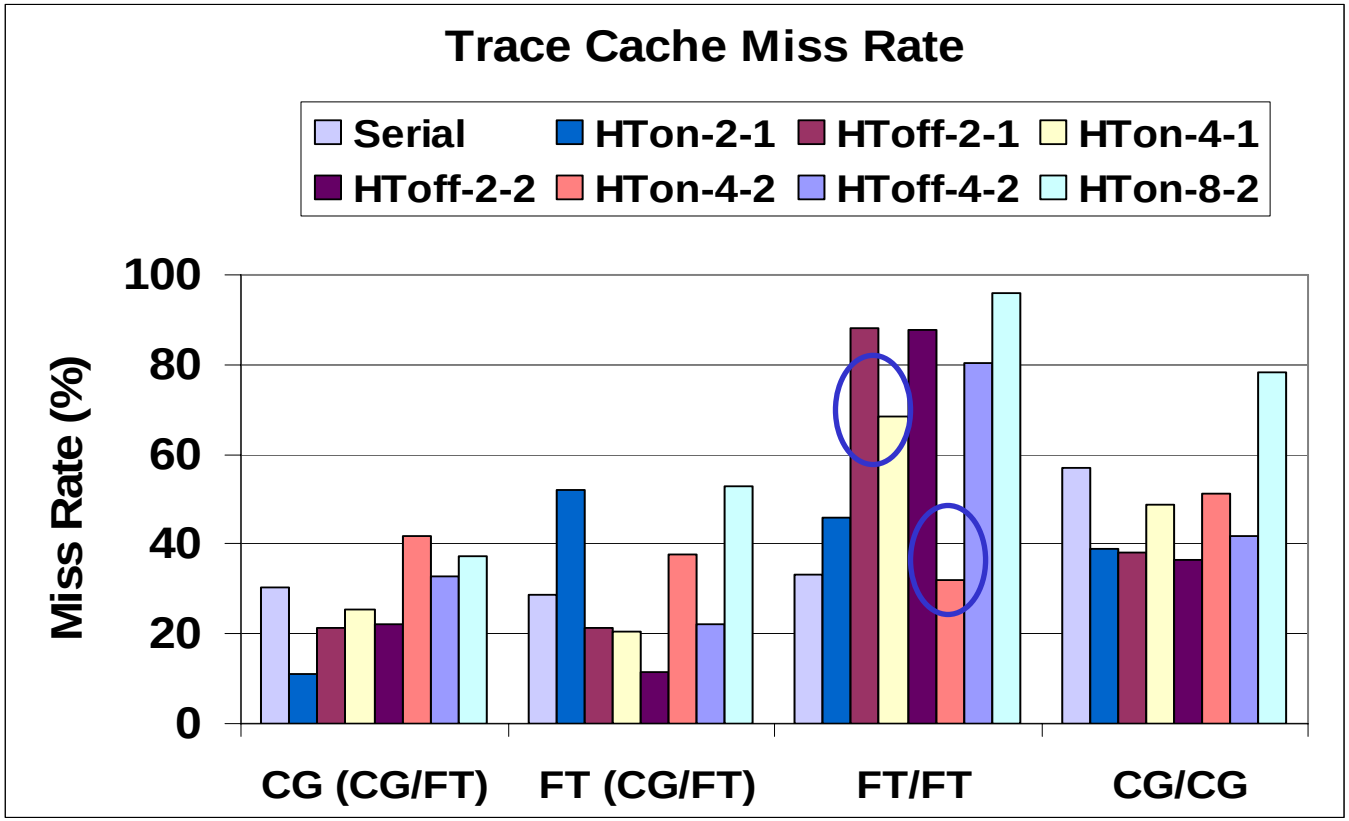


The L1 cache miss rates are relatively stable across the different configurations.

The HT_{on}-2-1 configuration has difficulty achieving a high hit rate for the CG, as does the HT_{on}-8-2 configuration.
In general, all of the HT_{on} configurations have a worse L2 miss rate than their HT_{off} equivalents, except for a couple of cases with the CG/CG workload.

Multithreaded, Multi-Program Results

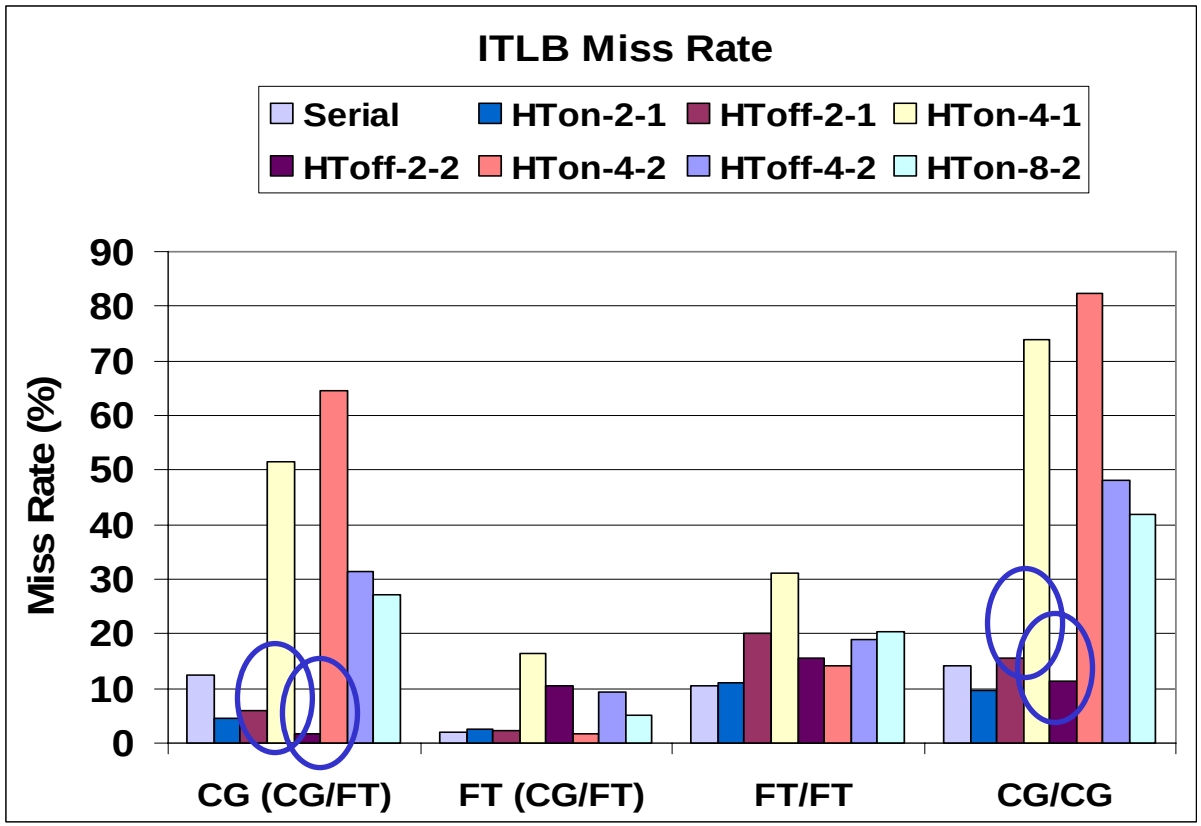
- **Trace Cache Performance**



HT_{off} configurations for both groups 2 and 3 are better than the HT_{on} configurations for both the CG/FT and CG/CG workloads, with the HT_{on} configurations having an advantage in the FT/FT workload. Also, there is no advantage to the HT_{on}-8-2 configuration .

Multithreaded, Multi-Program Results

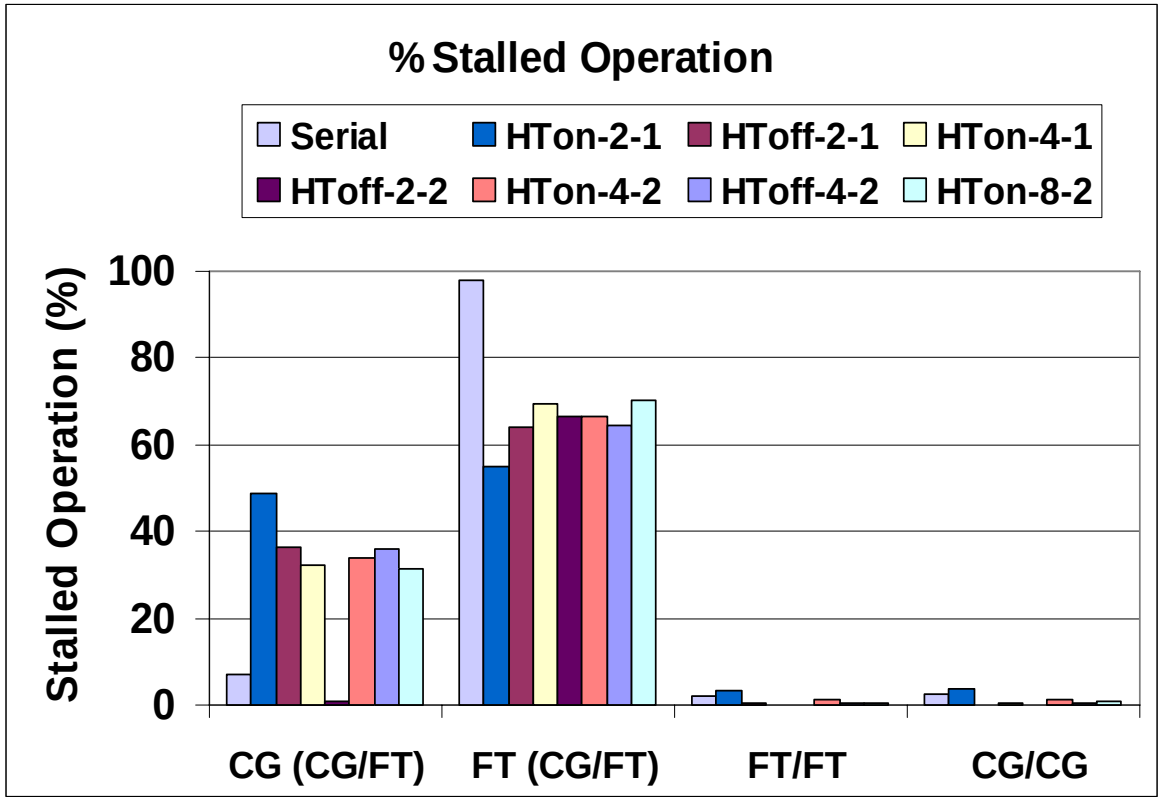
- **ITLB Performance**



The HT_{on} configurations suffer from excessive ITLB misses in groups 2 and 3 when running CG.

Multithreaded, Multi-Program Results

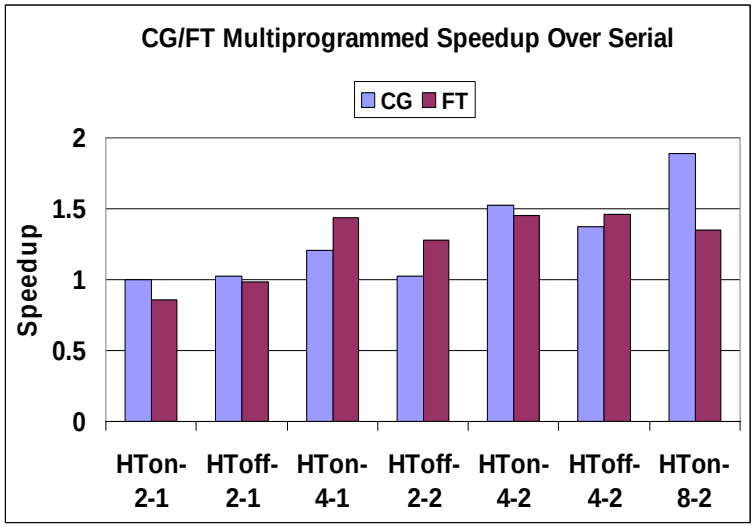
- **Stall Cycles**

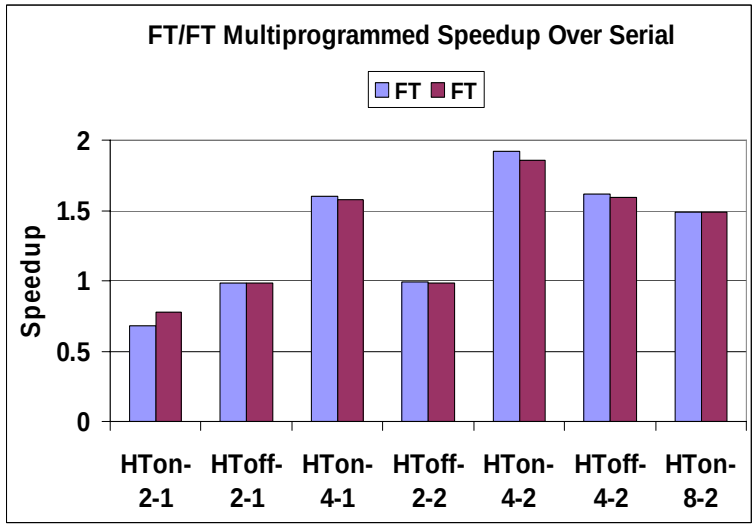


When running a complimentary workload (CG/FT), we can see that a significant amount of time is spent in a stalled state. From this, one can infer that the system is having a very difficult time providing the programs with the required resources, possibly switching the processors on which the programs are running frequently.

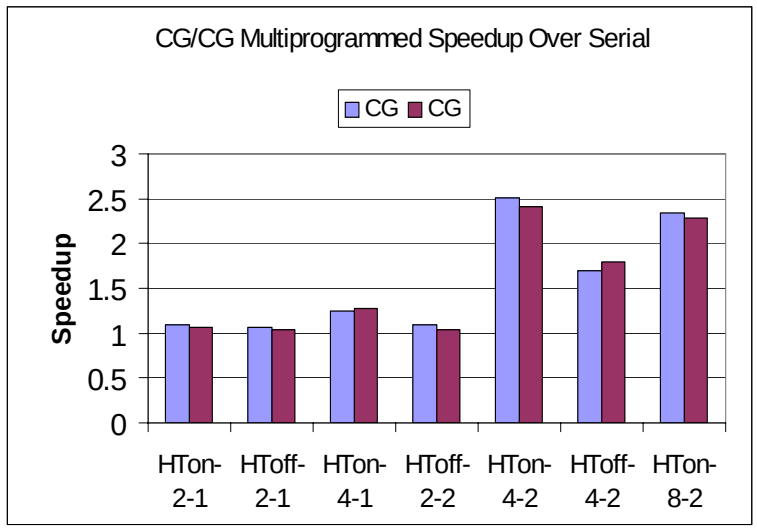
Multithreaded, Multi-Program Results

- **Wall Clock Performance**





The results clearly indicate that there is a tangible performance benefit to running compute bound and memory bound applications separately as the performance of both applications is better in such a configuration for most architectures.

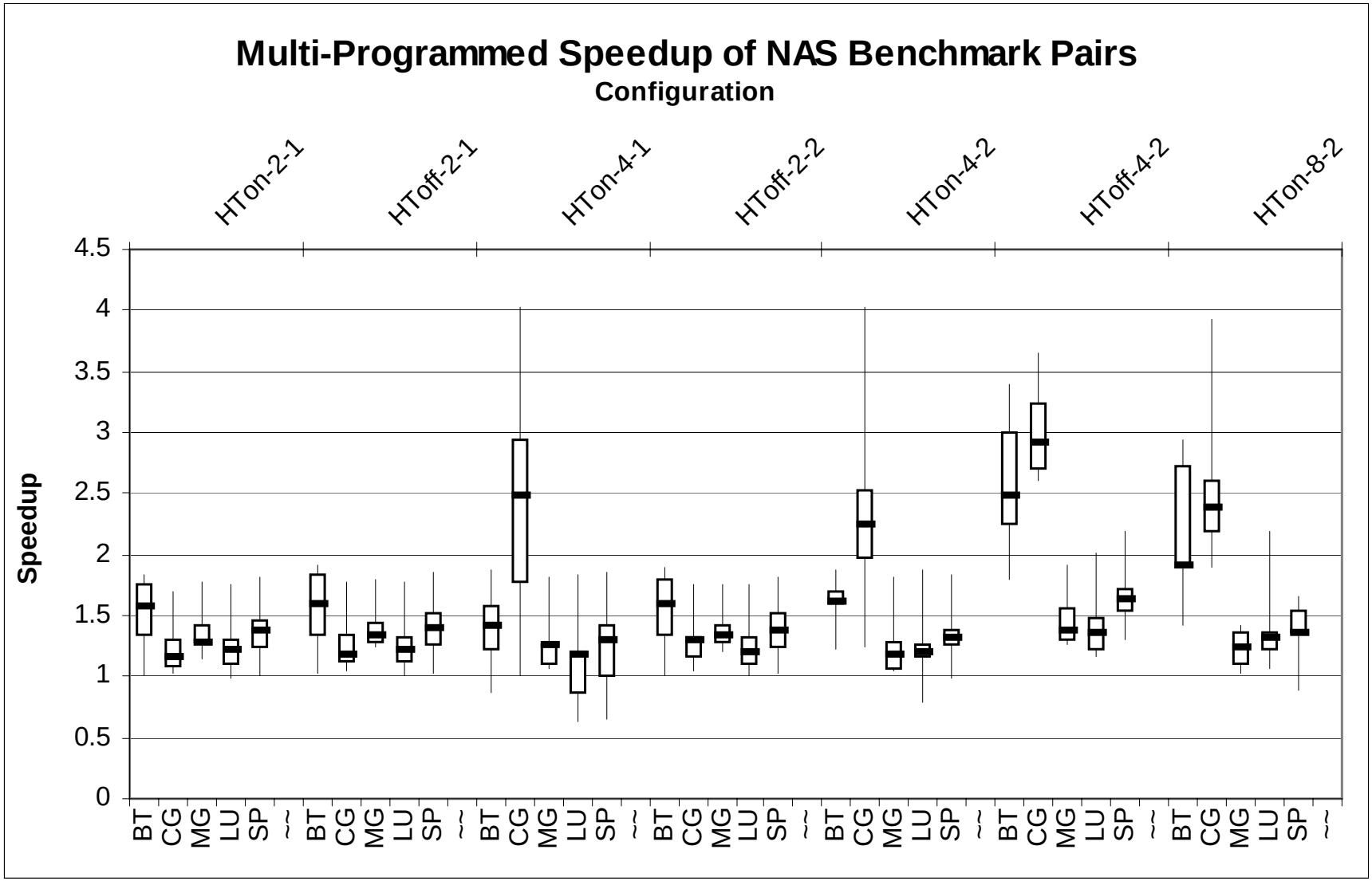


Cross-Product Multi-Program Results

- **Speedup of NAS Benchmark Pairs**
  - The configurations were tested using pairs of applications, and completed for all possible two-program configurations.
  - The program pairs were run with enough evenly distributed threads as to fully load the architecture under test.
  - The boxes in the next slide represent the interquartile ranges of data (the 25th and 75th percentile of the data falls within the box), while the whiskers represent the maximum and minimum of the data.
  - From the results in the next slide, we can conclude that the HT_{off}-4-2 (CMP-based SMP) architecture provides the overall best performance for the majority of program pairs across all of the benchmarking programs. However, for certain program pairs, the HT_{on} architectures can provide better overall performance.

Cross-Product Multi-Program Results

- **Speedup of NAS Benchmark Pairs**



Presentation Outline

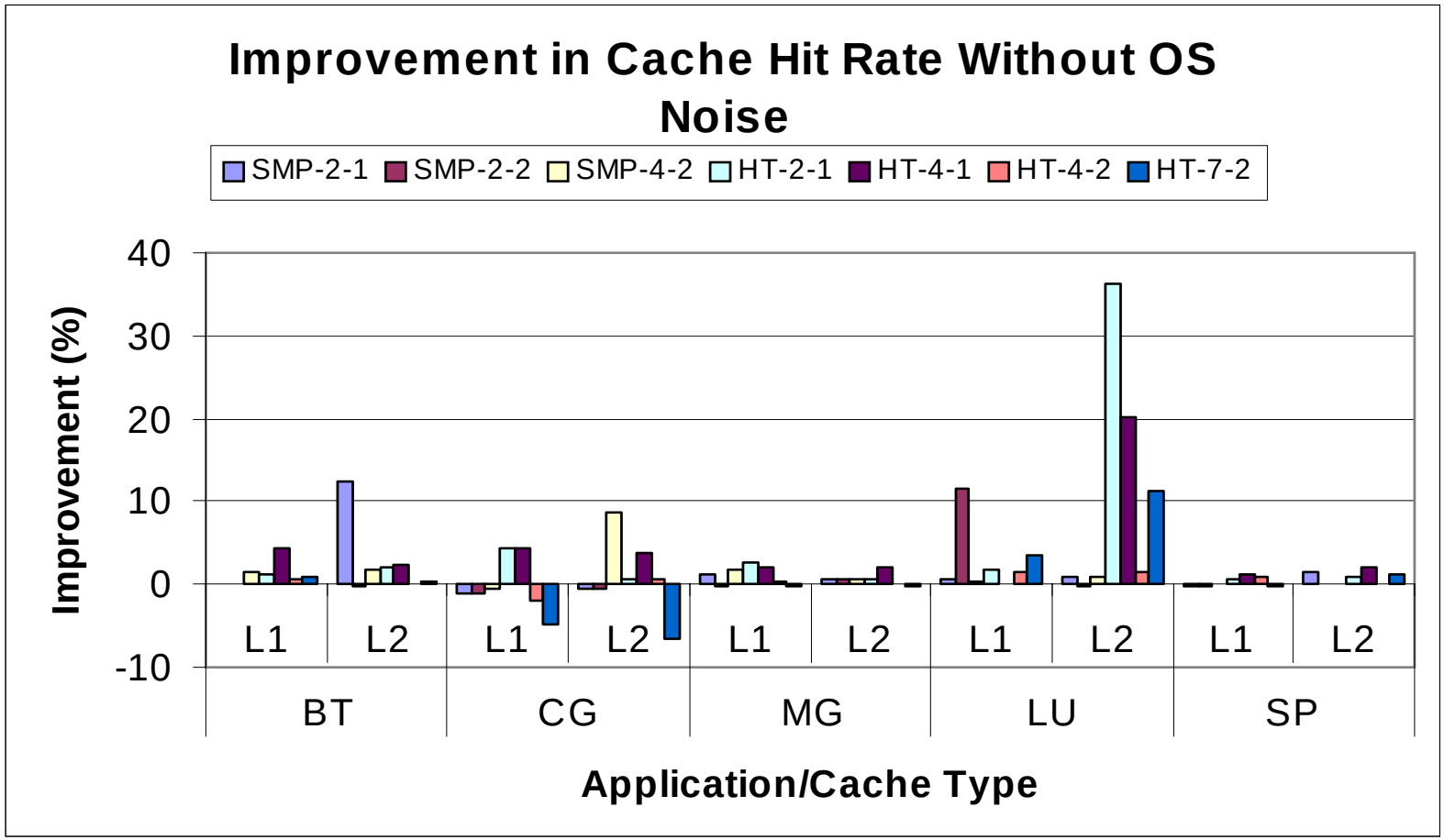
- Introduction
- Experimental Framework
- Multithreaded, Single-Program Results
- Multithreaded, Multi-Program Results
- Work in Progress
- Conclusions and Future Work

Work in Progress

- Asymmetric multiprocessors (AMP): a form of single-ISA heterogeneous architecture, made up of multiple processors operating at different speeds.
- AMP with Power-Saving Scheduler:
  - Bind all OS activities to (logical) processor zero that runs at a lower frequency than the rest of processors in the AMP.
  - In order to sustain performance for the parallel OpenMP threads, all other (logical) processors run at their maximum frequency.
- By offloading system noise onto a single (logical) processor, we can speculate that by avoiding swapping the threads in and out of the CPU we can increase the time available to user threads.
- This reduced noise will correspond to an increased performance of the user threads such that the impact of reserving a processor for system tasks at lower frequency is minimized. This frequency scaling of the reserved processor has a power saving effect.

Work in Progress

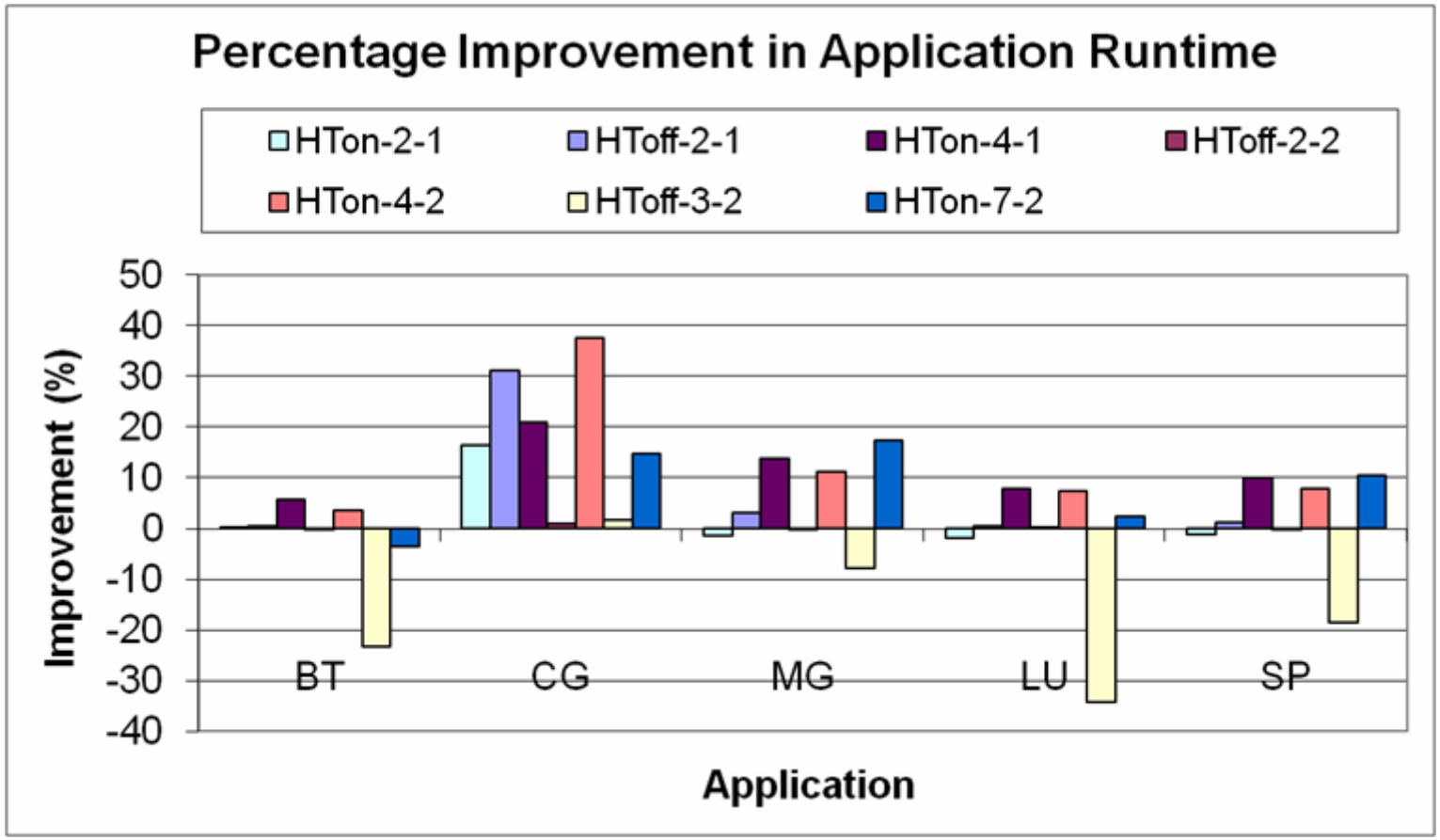
- Removing O/S Noise on CMP/CMT-based SMPs



Two of the configurations (HTon-7-2 and HToff-3-2) presented here are special cases in that they were tested using one thread less than their operating system loaded counterparts.

Work in Progress

- Impact on applications:



Presentation Outline

- Introduction
- Experimental Framework
- Multithreaded, Single-Program Results
- Multithreaded, Multi-Program Results
- Work in Progress
- Conclusions and Future Work

Conclusions

- This work sought to identify the optimal configurations for running OpenMP applications and to discover the shared resources that might become a bottleneck to performance under the different hardware configurations available in chip multithreaded SMPs.
- The most efficient architecture (single-program case) is a single dual-core processor with HT enabled, in terms of total computing power per system resources available. However, only one application enjoyed a performance gain due to HT on both dual-core processors.
- We discovered that the CMP-based SMP (HT_{off}-4-2) provides the best overall performance for the majority of program pairs across all of the benchmarking programs for the multi-program cases.

Future Work

- It is clear that the CPI, L1 and L2 cache miss ratios, and other metrics are not foolproof indicators of machine performance. Therefore, we intend to use the knowledge presented here to devise methods/models of measuring the overall efficiency of the system to achieve performance.
- We would like to extend our study with real applications on larger systems.
- The decisions made by the scheduler are crucial to the performance of multithreading architectures. We are currently experimenting with other schedulers to improve the performance of OpenMP applications on multi-core platforms.

Acknowledgment

- This work was supported by:
  - Natural Sciences and Engineering Research Council of Canada (NSERC)
  - Canada Foundation for Innovation (CFI)
  - Ontario Innovation Trust (OIT)
  - Queen's University

Questions?