A Preliminary Workload Analysis of SPECjvm2008

Hitoshi Oi

The University of Aizu

January 24, 2009

International Conference on Computer Engineering and Technology 2009

Outline

- Introduction: SPECjvm2008
- Workload Descriptions
- Performance Metrics and Run Rules
- Evaluation Methodology and Environment
- Experimental Results
- Conclusions and Future Work

Objectives

- Introduction to this new client-side JRE benchmark and understand the workload.
- Run SPECjvm2008 on three machines with difference cache sizes and clock frequencies but the same CPU microarchitecture.
- Measure the performance metrics and relate them to the workload parameters including cache reference and miss rates, clock speed and multi-threading.

Introduction: SPECjvm2008

- A new benchmark suite from SPEC for the client-side Java Runtime Environment.
- Released on May this year and available for free download (thus far).
- It replaces SPEC JVM98, released a decade ago.
- It should reflect the trend of current Java applications as well as JRE technologies.
- More benchmarks (8 to 38) and new types of workloads (e.g. XML processor).

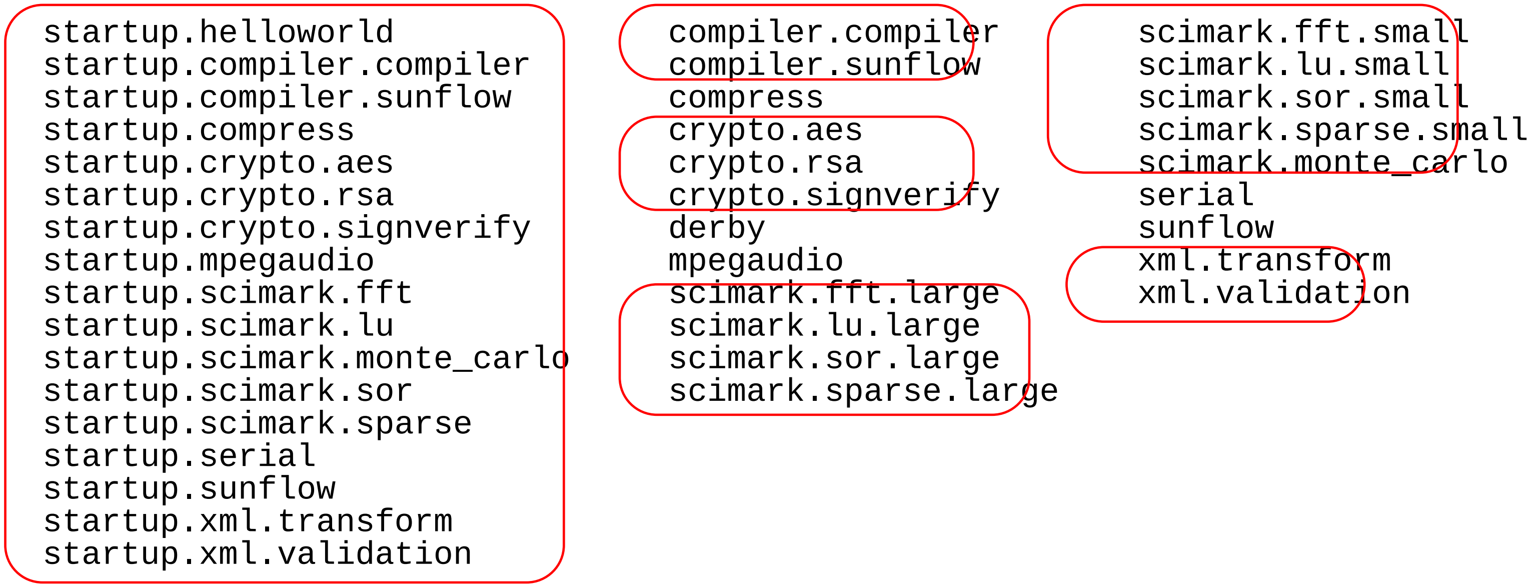
Workload Descriptions

38 Benchmark Programs

- startup.helloworld
- startup.compiler.compiler
- startup.compiler.sunflow
- startup.compress
- startup.crypto.aes
- startup.crypto.rsa
- startup.crypto.signverify
- startup.mpegaudio
- startup.scimark.fft
- startup.scimark.lu
- startup.scimark.monte_carlo
- startup.scimark.sor
- startup.scimark.sparse
- startup.serial
- startup.sunflow
- startup.xml.transform
- startup.xml.validation
- compiler.compiler
- compiler.sunflow
- compress
- crypto.aes
- crypto.rsa
- crypto.signverify
- derby
- mpegaudio
- scimark.fft.large
- scimark.lu.large
- scimark.sor.large
- scimark.sparse.large
- scimark.fft.small
- scimark.lu.small
- scimark.sor.small
- scimark.sparse.small
- scimark.monte_carlo
- serial
- sunflow
- xml.transform
- xml.validation

Workload Descriptions (cont.)

Eleven Groups



```
startup.helloworld
startup.compiler.compiler
startup.compiler.sunflow
startup.compress
startup.crypto.aes
startup.crypto.rsa
startup.crypto.signverify
startup.mpegaudio
startup.scimark.fft
startup.scimark.lu
startup.scimark.monte_carlo
startup.scimark.sor
startup.scimark.sparse
startup.serial
startup.sunflow
startup.xml.transform
startup.xml.validation
```

```
compiler.compiler
compiler.sunflow

compress

crypto.aes
crypto.rsa
crypto.signverify

derby

mpegaudio

scimark.fft.large
scimark.lu.large
scimark.sor.large
scimark.sparse.large
```

```
scimark.fft.small
scimark.lu.small
scimark.sor.small
scimark.sparse.small
scimark.monte_carlo

serial

sunflow

xml.transform
xml.validation
```

Workload Descriptions (cont.)

Compiler Compilation of javac (compiler.compiler) and another benchmark in SPECjvm2008 (compiler.sunflow).

Compress File compression using LZW algorithm (ported from SPEC95).

Crypto Encrption and decryption using AES (crypto.aes), RSA (crypto.rsa) and sign verification (crypto.signverify).

Derby Database in Java emphasizing BigDecimal computations.

MPEGaudio Mp3 audio decoder in Java stressing floating-point operations.

Workload Descriptions (cont.)

Scimark Five floating point sub-benchmarks in Java (fft, lu, monte_carlo, sor and sparse). Run with large (32MB) and small (512KB) data sets (scimark.*.large and scimark.*.small, except monte_carlo).

Serial Primitives and objects from JBoss benrhmark are serialized, sent over the socket and de-serialized in a producer-consumer manner.

Startup A new JVM is started for each benchmark in SPECjvm and a single iteration of the benchmark is executed.

Workload Descriptions (cont.)

Sunflow Multi-threaded image rendering benchmark.

XML Benchmark of XML document processing: Applying style sheets to XML documents using javax.xml.transform (xml.transform), and validating XML documents by javax.xml.validation (xml.validation).

Compared to JVM98

- Small changes: compress and mpegaudio
- Significan changes: compiler, derby and sunflow
- New for SPECjvm2008: crypto, scimark, serial, startup and xml.

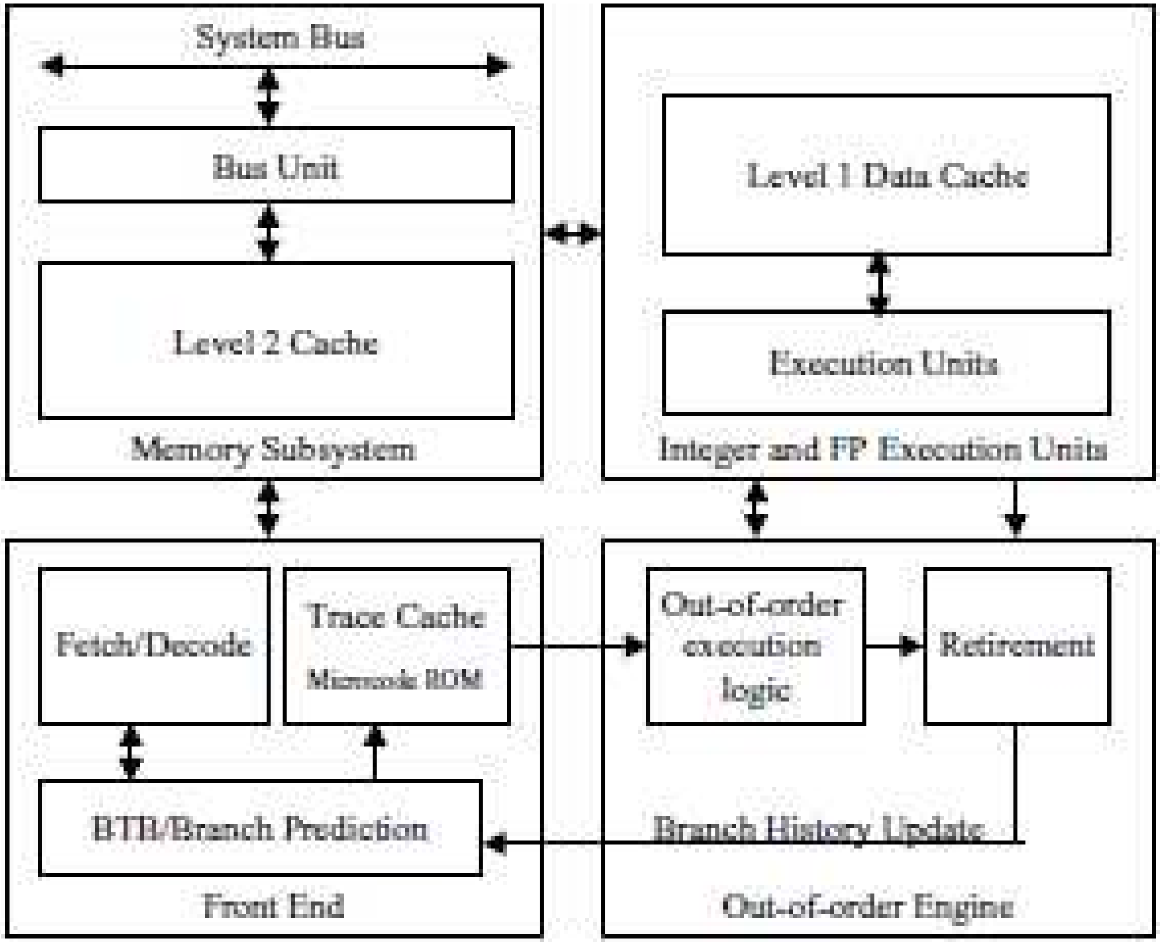
Performance Metrics and Run Rules

- The performance metrics is the geometric mean of the number of executions for each benchmark group.
- For the benchmark group with multiple sub-benchmarks, geometric mean of sub-benchmarks represents the metrics of the group.
- The number of threads spawned is equal to the number of logical processors (cores × SMT), except sunwlow ($2 \times CPU$).
- No JRE tuning is allowed in Base metrics, while it is allowed in Peak metrics.

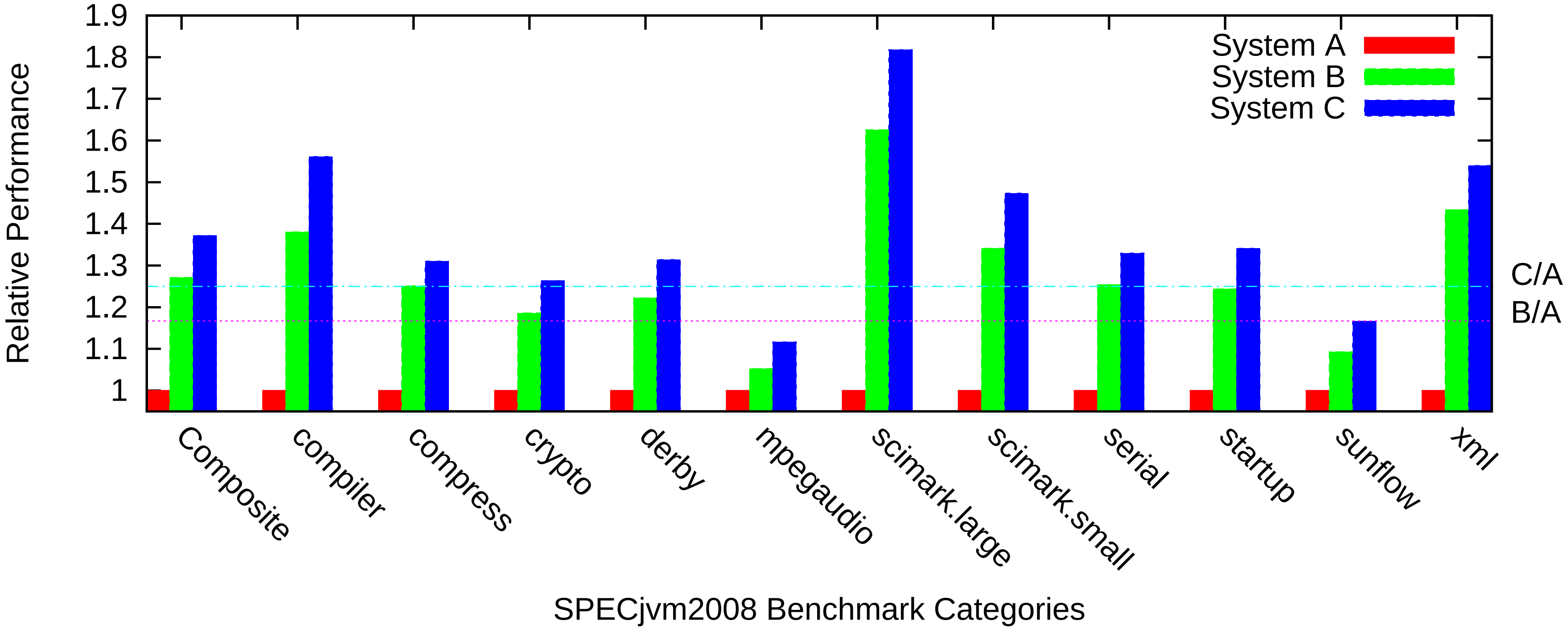
Experimental Environment

System	A	B	C
Processor	Pentium 4	Pentium D	
Clock Speed	2.4GHz	2.8GHz	3GHz
Trace Cache	12K uOps		
L1 Data Cache	8KB	16KB	
L2 Cache	512KB	1MB	2MB
ITLB	128 Entries		
Operating Systems	Linux (CentOS) Kernel 2.6.18		
Java Version	1.6.0_04		
JVM Name	Java HotSpot(TM) Client VM		

Pentium 4 Microarchitecture

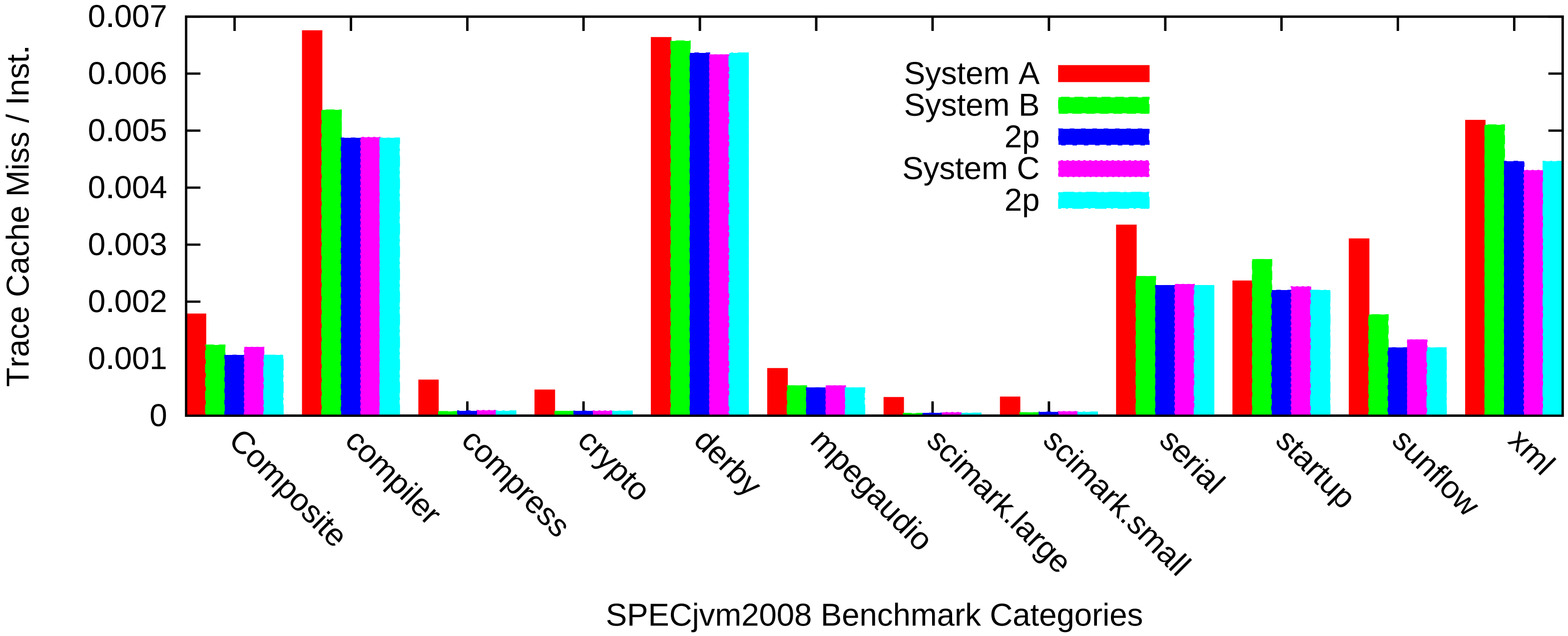


Result: SPECjvm2008 Performance Metrics



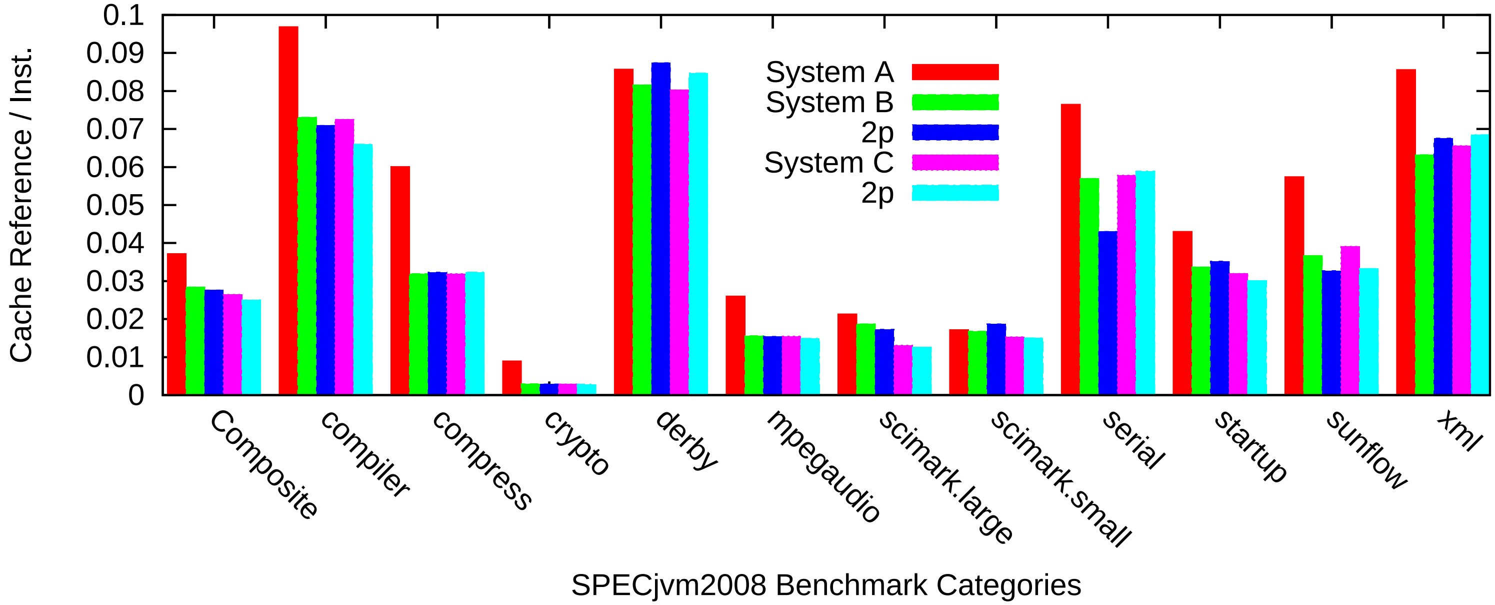
- mpegaudio, sunflow < Relative Clock Speed
- crypto ≈ Relative Clock Speed
- All others > Relative Clock Speed

Result: Trace Cache Miss Rates



- Low TC miss rates for Loop-intensive applications (e.g. scimark)
- Application with conditional branchs have high TC miss rates
- Why compiler and sunflow high TC miss rates for A ?

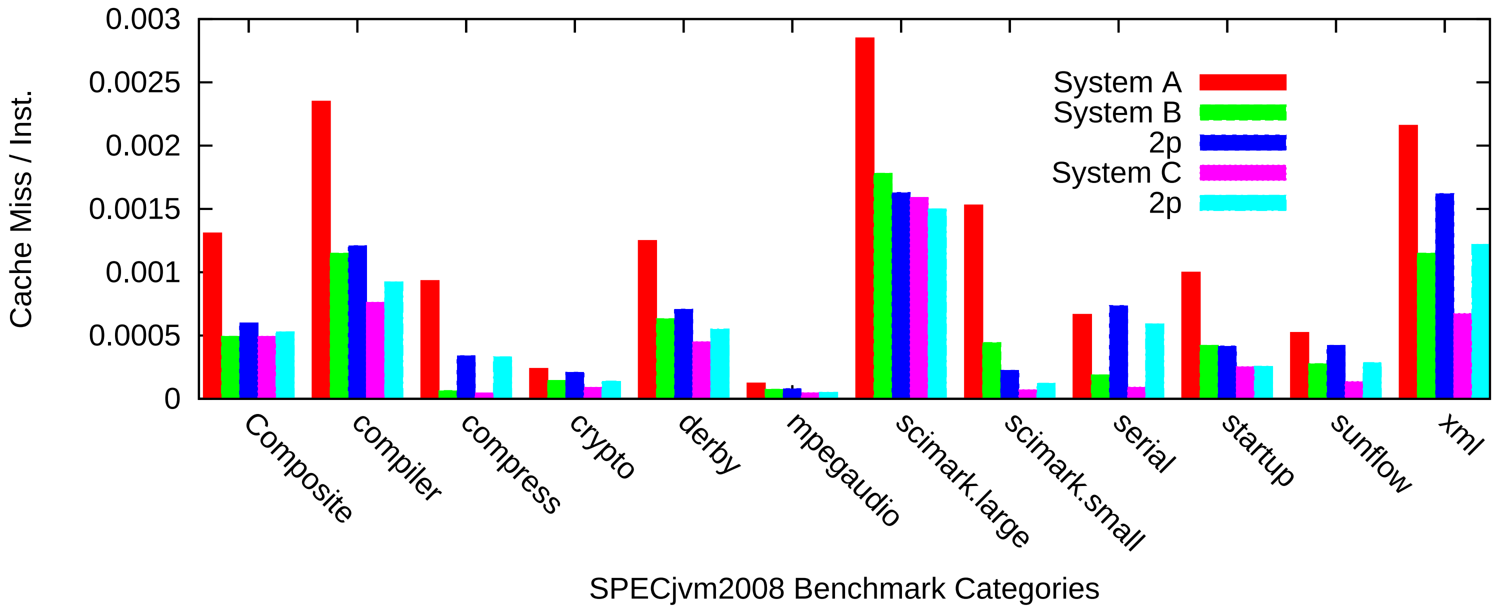
Result: L2 Cache Reference Rates



· L1 data miss rates >> TC miss rates

· 8KB L1 Date Cache too small for compiler, compress serial, sunflow and xml

Result: L2 Cache Miss Rates

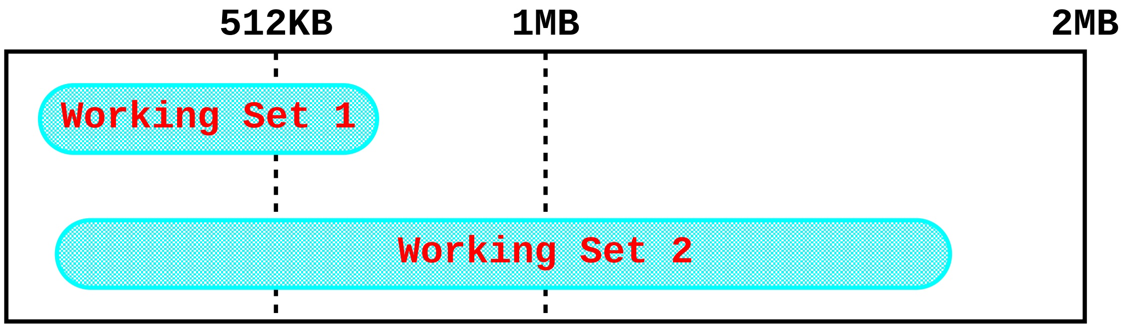


- 512KB not enough for most benchmarks
- 1MB: compress, crypto, mpegaudio and scimark.large
- 2MB: compiler, derby, scimark.small, startup, sunflow and xml

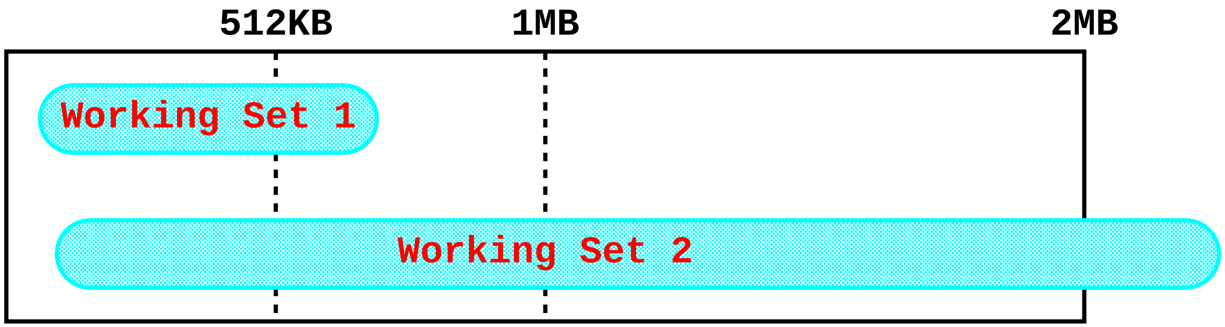
Why 2MB L2 not effective for scimark.large ?

A possible reason:

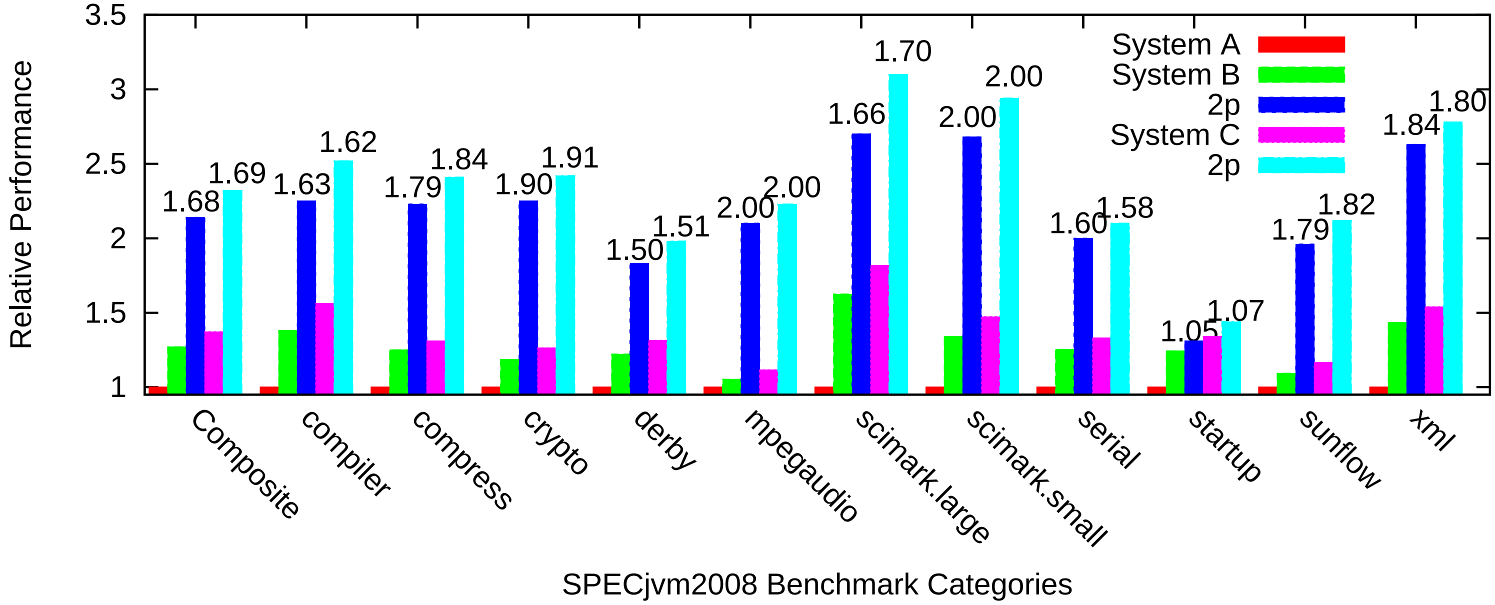
scimark.small



scimark.large



Result: Multi-Threading



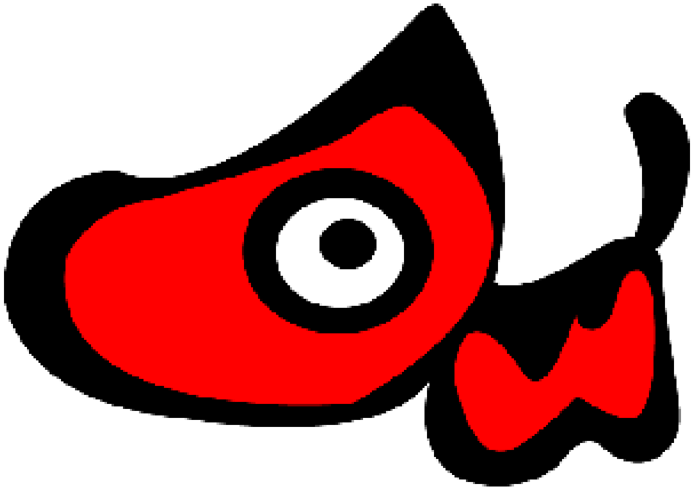
- 50 to 100% speed-up compared to single core cases
- 5 to 7% speed-up for single threaded startup (outside JRE ?)
- L2 miss rates increased for compress, serial, sunflow and xml.

Summary

- Memory access behaviors of SPECjvm2008 are profiled on the netburst CPUs.
- Roughly speaking, 1MB/core L2 cache looks sufficient.
- Further investigation needed for internal behavior of JVM (e.g. method invocation, bytecode exec. frequency), not possible with oprofile alone.
- Sub-benchmark behavior (e.g. scimark: fft and monte carlo are quite different).
- Measurements with different microarchitecture, JVM implementation.

Thanks for You Attention

Any question ?



(Akabeko – a local handicraft of Aizu)