Metrics for System Investigation Performance Evaluation of Computer Systems

Vojtěch Horký Peter Libič Petr Tůma

Department of Distributed and Dependable Systems Faculty of Mathematics and Physics Charles University

2010 - 2023

- Intermezzo: Out Of Order Execution
- Intermezzo: Branch Prediction
- Processor Behavior Metrics
- 5 Intermezzo: Memory Hierarchy
- 6 Memory Related Behavior Metrics

System Investigation

Measuring for the purpose of understanding system behavior.

Requirements:

- Directly related to specific system components.
- Configurable to fit variety of investigated systems.
- Reasonably simple to measure during development or operation.

Pitfalls:

- Metric design often influenced by what we can measure.
- Behavior of specific components may be difficult to isolate.
- Relationship to practically observed performance questionable.

Overview

Intermezzo: Out Of Order Execution

- Intermezzo: Branch Prediction
- Processor Behavior Metrics
- 5 Intermezzo: Memory Hierarchy
- 6 Memory Related Behavior Metrics

Current Processor Characteristics

Pipelined

Multiple instructions processed at different execution stages.

Superscalar

Multiple instructions dispatched simultaneously to multiple execution units.

Out Of Order Processing

Instructions scheduled for execution and retired based on dependencies.

Speculative Program Execution

Instructions may be executed based on speculation about future state.

General terminology may not fit when applied on particular processor.

- 2 Intermezzo: Out Of Order Execution
- Intermezzo: Branch Prediction
 - Processor Behavior Metrics
 - 5 Intermezzo: Memory Hierarchy
 - 6 Memory Related Behavior Metrics

Branch Prediction

Condition Prediction

Trying to guess whether a conditional jump will jump or not.

- Concerns most loops and branches in source.
- Short branches also done with conditional instructions.

Target Prediction

Trying to guess where an indirect jump will jump.

- Concerns all virtual method invocations in source.
- Concerns all return statements in source.
- Concerns some switch statements.

Static Branch Prediction

Static Prediction

Predicting without knowledge of past behavior.

Not much can be done:

- Forward jumps predicted as not taken.
- Backward jumps predicted as taken.
- Guess why ?

Prediction With Counters

Single Bit

Remember last state as taken or not taken. Predict same behavior as last time.

- Works for loops with many iterations.
- Poor for many common patterns.

Saturating Two Bits

Use saturating counter that increments vs decrements depending on branch being taken vs not taken. Predict behavior depending on counter value.

• Still poor for many common patterns.

Prediction With History

History

Remember recent history as string of taken or not taken bits. Use history as index to table of saturating counters.

- We already have a hash table of counters anyway.
- Fixes behavior with short patterns that break counters alone.
- History either local for one branch or global across all branches.

Branch Target Buffer

One Target

Simply store last branch target in hash table.

- Not very good with polymorphic targets.
- Some benchmarks suggest success around half of the time.

More Targets

Store multiple targets indexed by history.

- History of past addresses or parts of those.
- Some benchmarks suggest global history better than local.

In Reality ?

Real designs mix more prediction principles.

Intel Sandy Bridge

- Two level predictor with 32 bits global history.
- Branch target buffer size probably around 4096 entries.
- Return target stack for up to 16 nested calls.

AMD Ryzen

- Hybrid predictor with perceptron.
 - Sounds arcane but in fact linear combination of selected history bits.
 - Of course many details are hidden in the training phase.
- Branch target buffer architecture and size not reported.
- Return target stack for up to 32 nested calls.

- 2 Intermezzo: Out Of Order Execution
- Intermezzo: Branch Prediction
- Processor Behavior Metrics
- 5) Intermezzo: Memory Hierarchy
- 6 Memory Related Behavior Metrics

Processor Behavior Metrics: Processor View

Overview

Metrics characterising application execution effectivity:

- Instructions per cycle (IPC or inverse CPI).
- Branch prediction hit (miss) count or rate.
- Memory accesses per instruction.
- ...

Metric properties:

- Useful for example to appraise code optimisations.
- Typically very much platform specific.

Processor Behavior Metrics: Application View

Overview

• ...

Metrics characterising application execution demands:

- Instruction mix in general terms.
- Average lifetime of register values.
- General predictability of branch instructions.

Metric properties:

- Very hard to define meaningful metrics and values.
- Platform independent measurement possible. http://boegel.kejo.be/ELIS/mica

- 2 Intermezzo: Out Of Order Execution
- Intermezzo: Branch Prediction
- Processor Behavior Metrics
- 5 Intermezzo: Memory Hierarchy
 - Memory Related Behavior Metrics

Memory Hierarchy Features

Translation Caching

Adress translation caches remember recent virtual to physical mappings.

Content Caching

Content caches remember recent data and hold recent writes.

Prefetching

Regular access patterns trigger prefetching.

Coherency

Single memory illusion maintained.

- 2 Intermezzo: Out Of Order Execution
- Intermezzo: Branch Prediction
- Processor Behavior Metrics
- Intermezzo: Memory Hierarchy



Cache Relevant Behavior Metrics

Overview

Metrics characterize memory access patterns.

- Cache misses (hits) per memory access (rate).
 - Individually for each cache level.
 - Also for address translation caches.
- Stack (reuse) distance. Number of accesses to unique addresses between reuses of the same address.
- Average memory access time usually in clock cycles. $T_{avg} = p_{hit} \cdot T_{cache} + (1 - p_{hit}) \cdot T_{memory}$

Metric properties:

- Depends on many platform properties (timing, prefetching, replacement strategies).
- Can guide application specific optimizations (data layout modifications, tiling, compute to fetch ratio).

Allocation Behavior Metrics

Overview

Metrics characterize dynamic (heap) memory allocation patterns.

- Allocation rate, deallocation rate. Should be the same, on average.
- Live size.

Total size of usable (reachable) memory.

Object lifetime.

What time elapses between object allocation and deallocation (becoming unreachable). Time unit is usually a byte allocated or an object allocated.

- Object size.
- Avg live size = Avg object size · Avg object lifetime