Critical-path Aware Performance Analysis

Dongju Chae Hanhwi Jang Jangwoo Kim
High Performance Computing Lab., POSTECH

Motivation

- How to **easily** identify performance bottlenecks of modern CPU **exactly**?
- How to apply **both** performance modeling theory and simulation technique?

Goal

- We propose a new performance modeling method which combines critical-path analysis theory [3] and stall-based CPI stack analysis [1].
- Our scheme can identify performance bottlenecks and their impacts using only a minimal number of simulations.
- Our result is similar to the result of full spectrum of design space explorations.

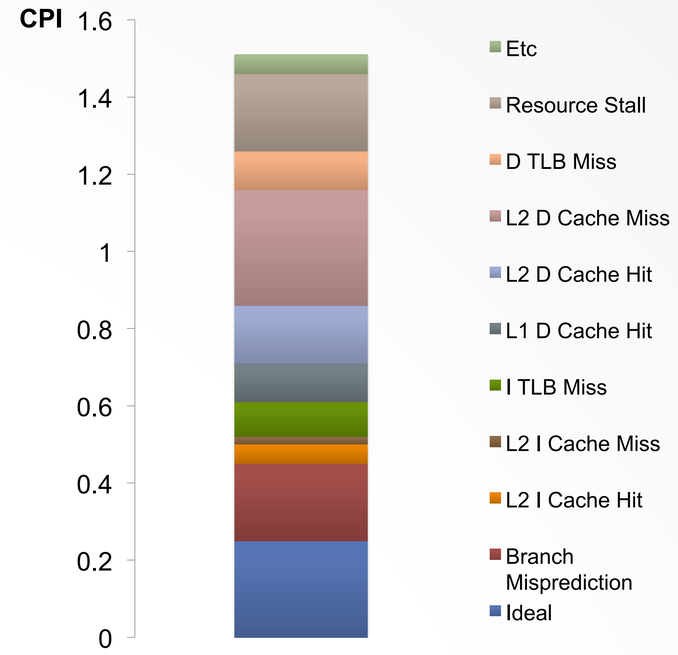


Figure 1. CPI Stack

Background

- CPI : **C**ycles **P**er **I**nstructions
  > how many cycles to execute one instruction?

- FMT (interval analysis [1][2])
  > CPI stack: ∑ (CPI components per uArch stall event)
  > how many cycles lost due to a specific stall event?
  (e.g., cache miss, branch misprediction, slow execution)

  However, inaccurate analysis due to overestimation

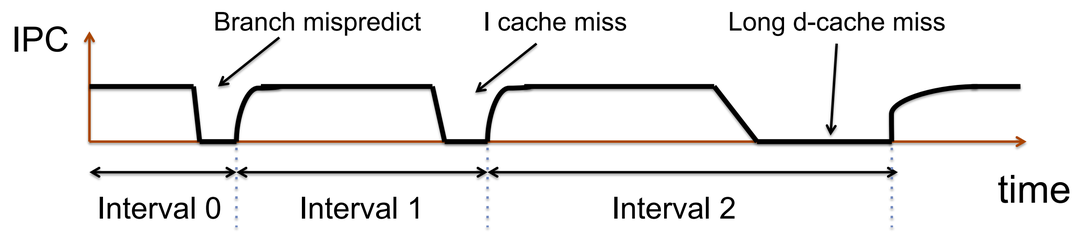


Figure 2. interval analysis

- Critical-path instruction stream [3]
  > a chain of events taking **longest cycles**
  > shorter chains do not affect overall performance

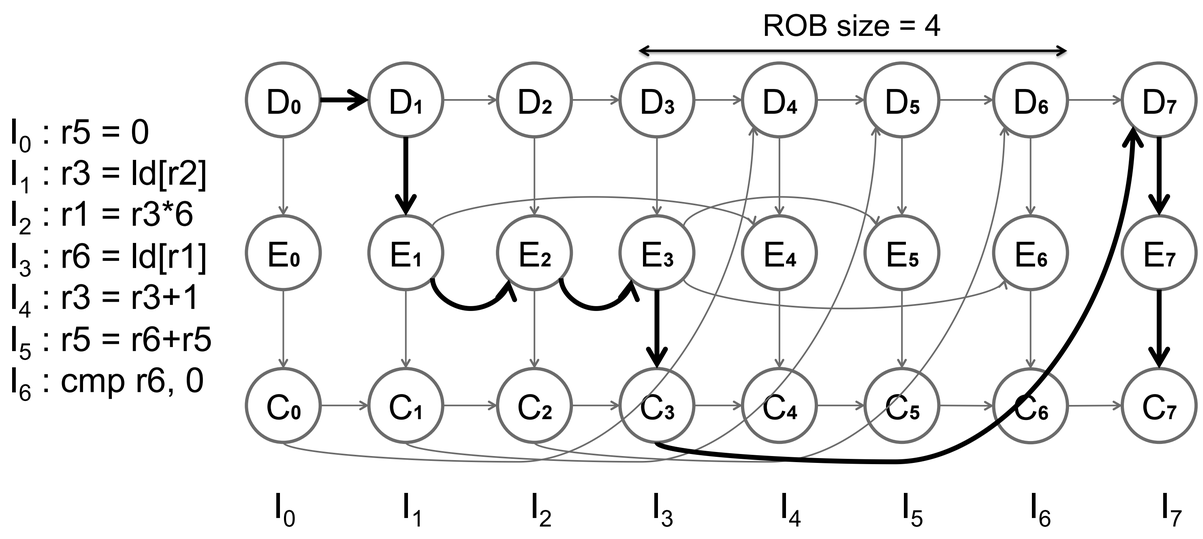


Figure 3. critical path

Our Approach

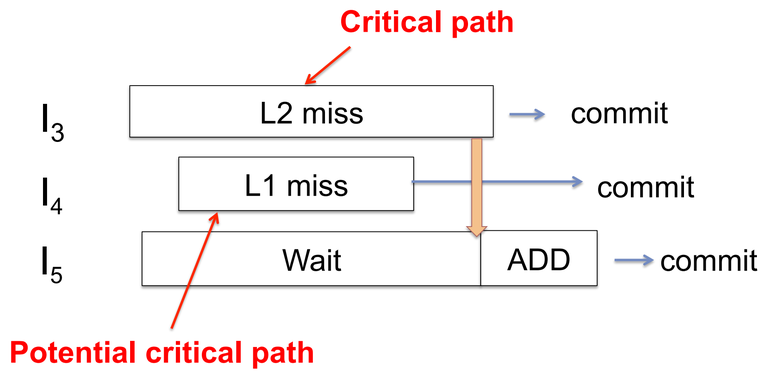
- Trace-based CPI stacking
  > use 'event- tagged' instruction traces.
  > penalty is distributed into each component.

Penalty : 180 cycle

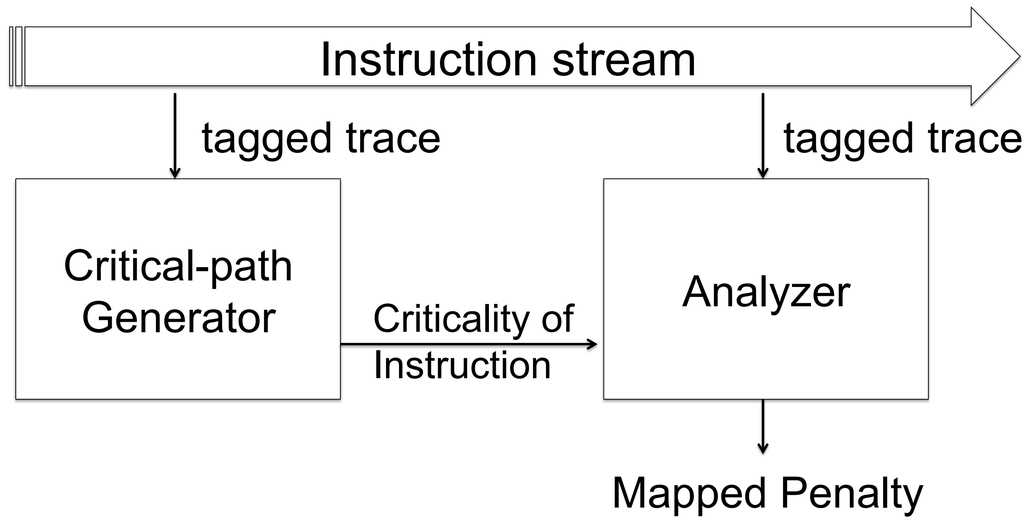
	afetch	fetch	dispatch	ready	issue	complete	commit	info[3]	dep[3]
…									
I_2	104	104	111	111	112	120	124	0 0 0	3 0 1
I_3	104	104	111	120	120	300	304	0 5 0	1 0 6
I_4	104	113	120	120	121	122	304	1 0 2	3 0 3
I_5	113	113	120	300	301	302	306	5 0 0	5 6 5
…									

Penalty : 2 cycle

- Critical-path awareness
  > apply different weights on different event streams



- Simulation and analysis process



Current Work

- We are implementing our hybrid scheme on top of various timing simulators (e.g., SimpleScalar, PTLsim, Marssx86.)
- We are extending our scheme to analyze the performance multi-core, multi-threaded CPUs.

References

[1] S. Eyerman, et al, "A performance counter architecture for computing accurate CPI components" ACM international Conference on Architectural Support for P rogramming Languages and operating Systems, 2006, p175-184
[2] S. Eyerman, et al, "A Mechanistic Performance Model for Superscalar Out-of-Order Processors" ACM Transactions on Computer Systems, 2009, p3:1~3:36
[3] Brian Fields, et al, "Focusing Processor Policies via Critical-Path Prediction" IS CA, 2001, p1~12