SimPoints

http://cseweb.ucsd.edu/~calder/simpoint/index.htm

MO801

Timothy Sherwood, Erez Perelman, Greg Hamerly, Suleyman Sair, and Brad Calder, Discovering and Exploiting Program Phases, IEEE Micro: Micro's Top Picks from Computer Architecture Conferences, December 2003.

Why?

- Computer Architects need to simulate large workloads
  - Cycle-level simulators are slow
  - Design Space Exploration requires multiple simulations
- Full simulations can take months to complete
- Phase analysis provides an accurate and efficient workaround for this problem

Alternatives

- Smaller input sizes
- Crop execution time
  - Run the first X instructions → like 5 billion instructions
  - Skip the first X instructions and run for the next Y instructions → like skip 1 billion instructions and run for 5 billion instructions

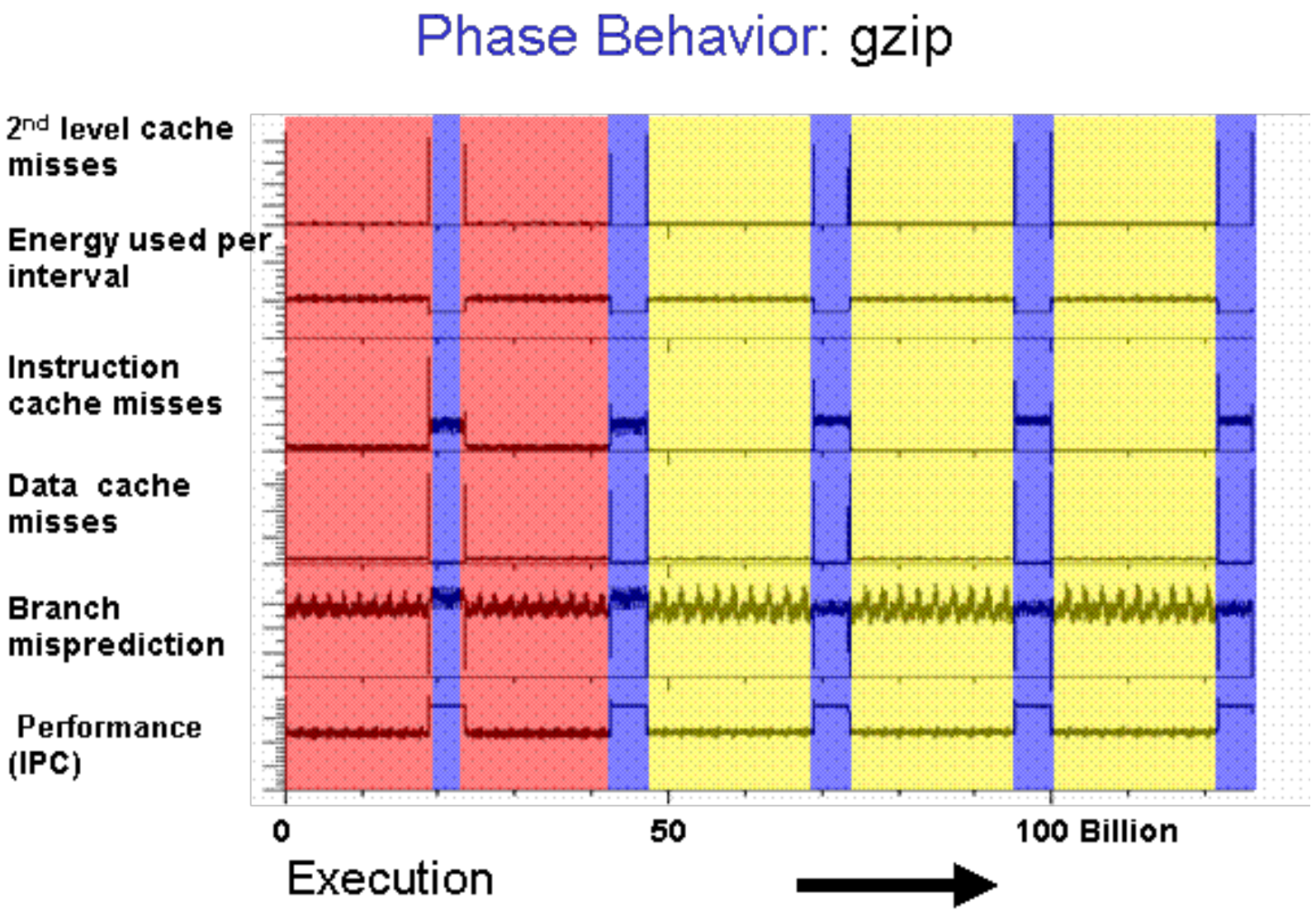
Program Phases

- Specific parts of the program that mimics the whole program execution
- You can get parameters from program execution by considering these small parts
  - Very good correlation
- Programs have several different phases
- There are similar phases
  - One SimPoint is the best representative point to a phase

Definitions

- Interval
  - A section of continuous execution within a program → 1, 10, or 100 million instructions (can be variable)
- Phase
  - A set of intervals with similar behavior, regardless of temporal adjacency
- Phase classification
  - Breaks program's execution into phases with similar behavior
- Similarity
  - Show how close the behavior of two intervals are
- Similarity metric
  - Metric to detect similarity (independent of hardware)
- Phase change
  - Noticeable and sudden change in program behavior over time

Example


Phase Behavior: gzip
2nd level cache misses
Energy used per interval
Instruction cache misses
Data cache misses
Branch misprediction
Performance (IPC)
0
50
100 Billion
Execution

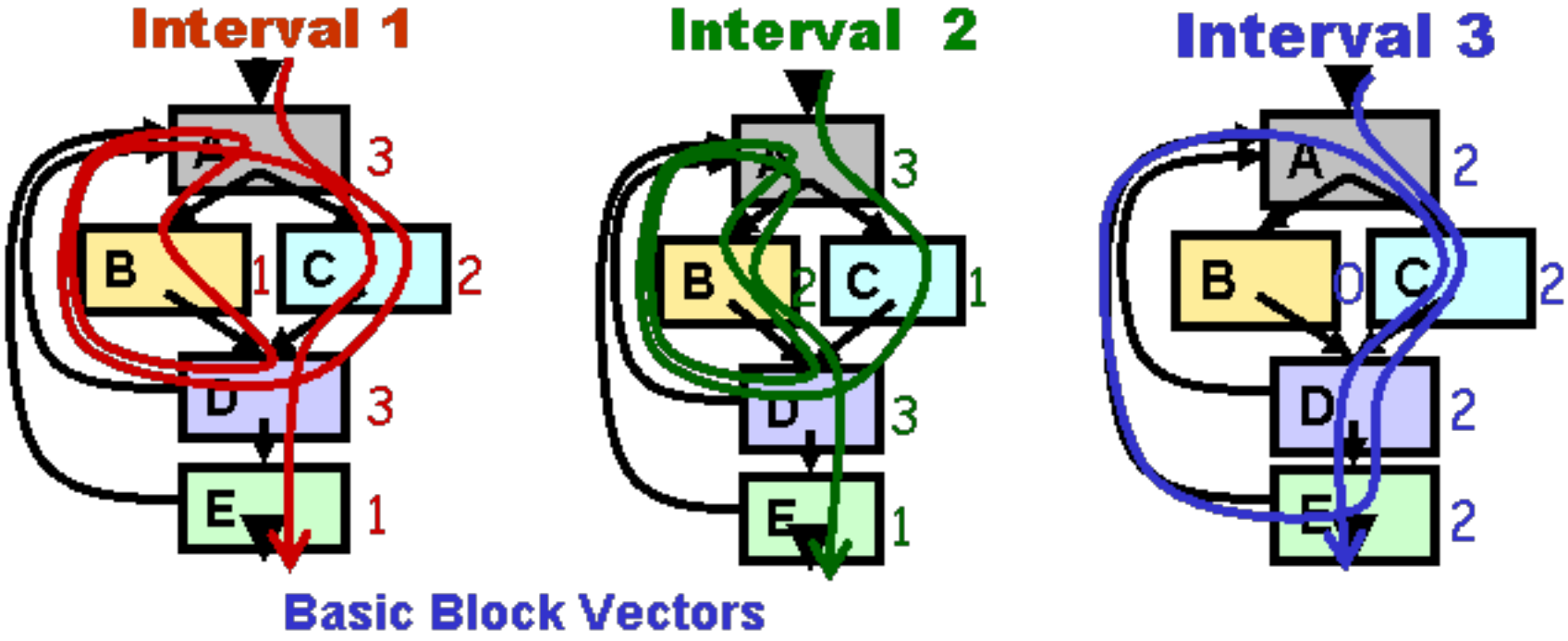
Phase observations

- Three phases: red, blue and yellow
- The behavior of each architectural metric shifts in unison with the other metrics
- Phases range over large sections of execution
- The behavior between phases can vary significantly
- There is strong structure and repetition for all the phases

SimPoint Technique

- Basic Block Vector Analysis
  - Architecture independent code profiling
- Random Projection
  - Reduces dimentionality of data
- Phase Classification using Kmeans clustering
  - Classifies all intervals into a set of phases
- Picking simulation points
  - Finds a good representation of the execution by using a sample from each phase

Basic Block Vector Analysis



Basic Block Vectors

	A	B	C	D	E
Interval 1 <	3,	1,	2,	3,	1 >
Interval 2 <	3,	2,	1,	3,	1 >
Interval 3 <	2,	0,	2,	2,	2 >

Basic Block Vectors

- One-dimensional array with one element for each static basic block in the program
- During each interval, counts the number of execution for each basic block, weighed by the number of instructions in the basic block
- Normalize by dividing each element by the sum of all the elements in the vector

Similarity

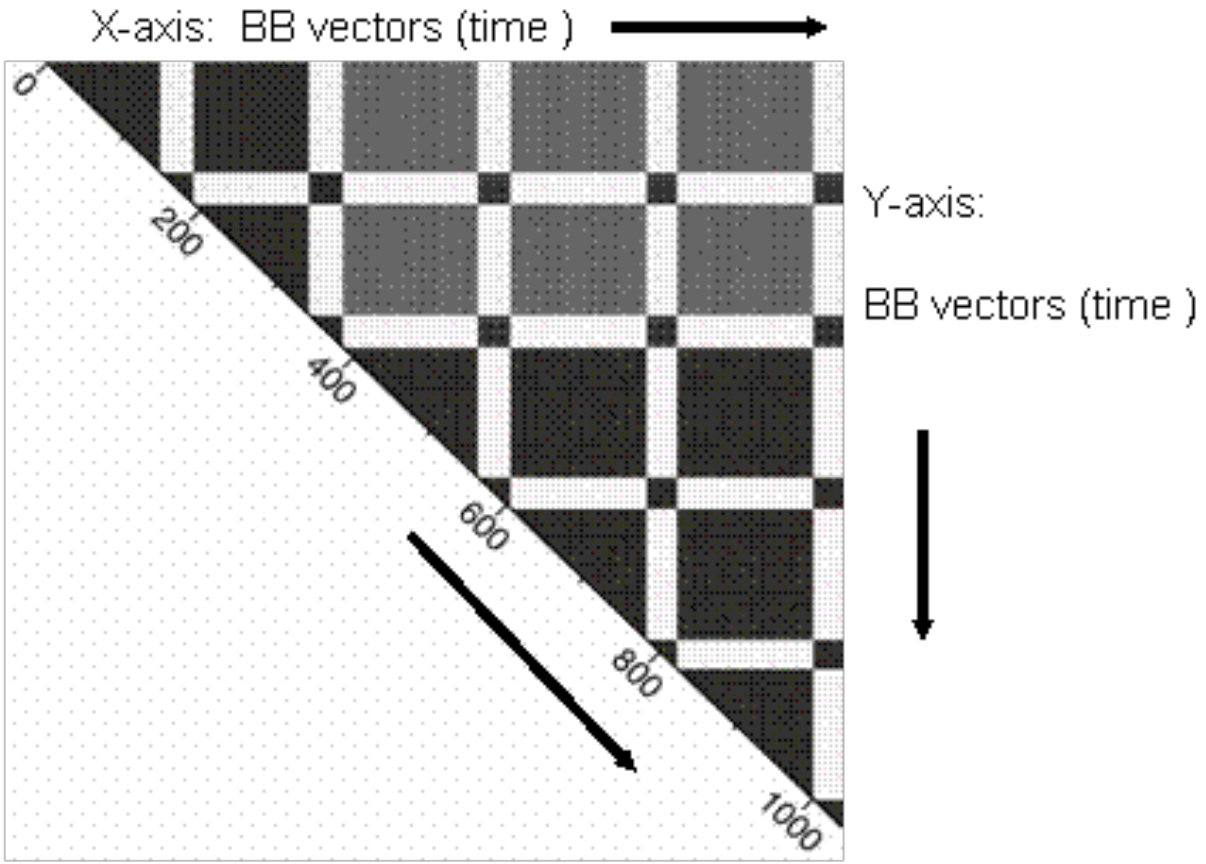
- Calculate the Manhattan Distance among BBV
  - Manhattan Distance = sum of the absolute value of the element-wise subtraction of two vectors
- Small distance means vectors are similar
  - May be in the same phase

Interval 1	< 3,	1,	2,	3,	1 >
Interval 2	< 3,	2,	1,	3,	1 >
Manh. Dist.	< 0,	1,	1,	0,	0> ≡ 2

Interval 2	< 3,	2,	1,	3,	1 >
Interval 3	< 2,	0,	2,	2,	2 >
Manh. Dist.	< 1,	2,	1,	1,	1> ≡ 6

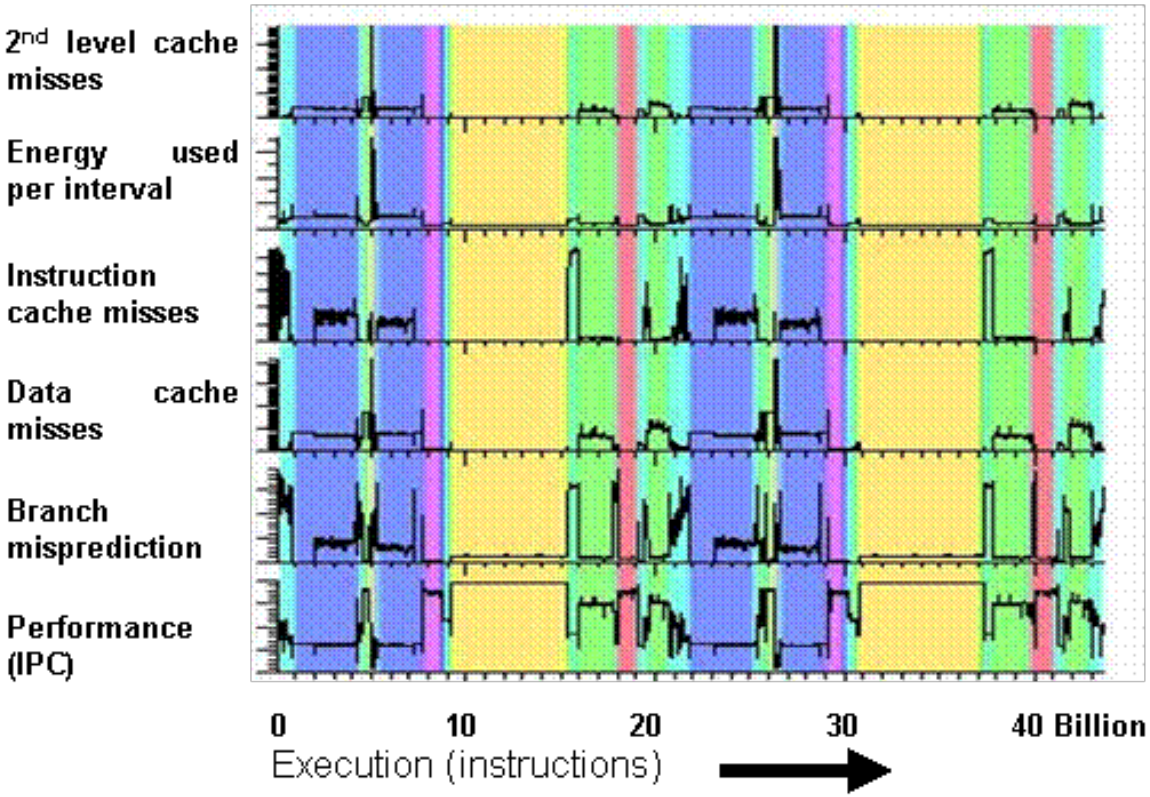
Basic Block Similarity Matrix

Similarity Matrix for gzip

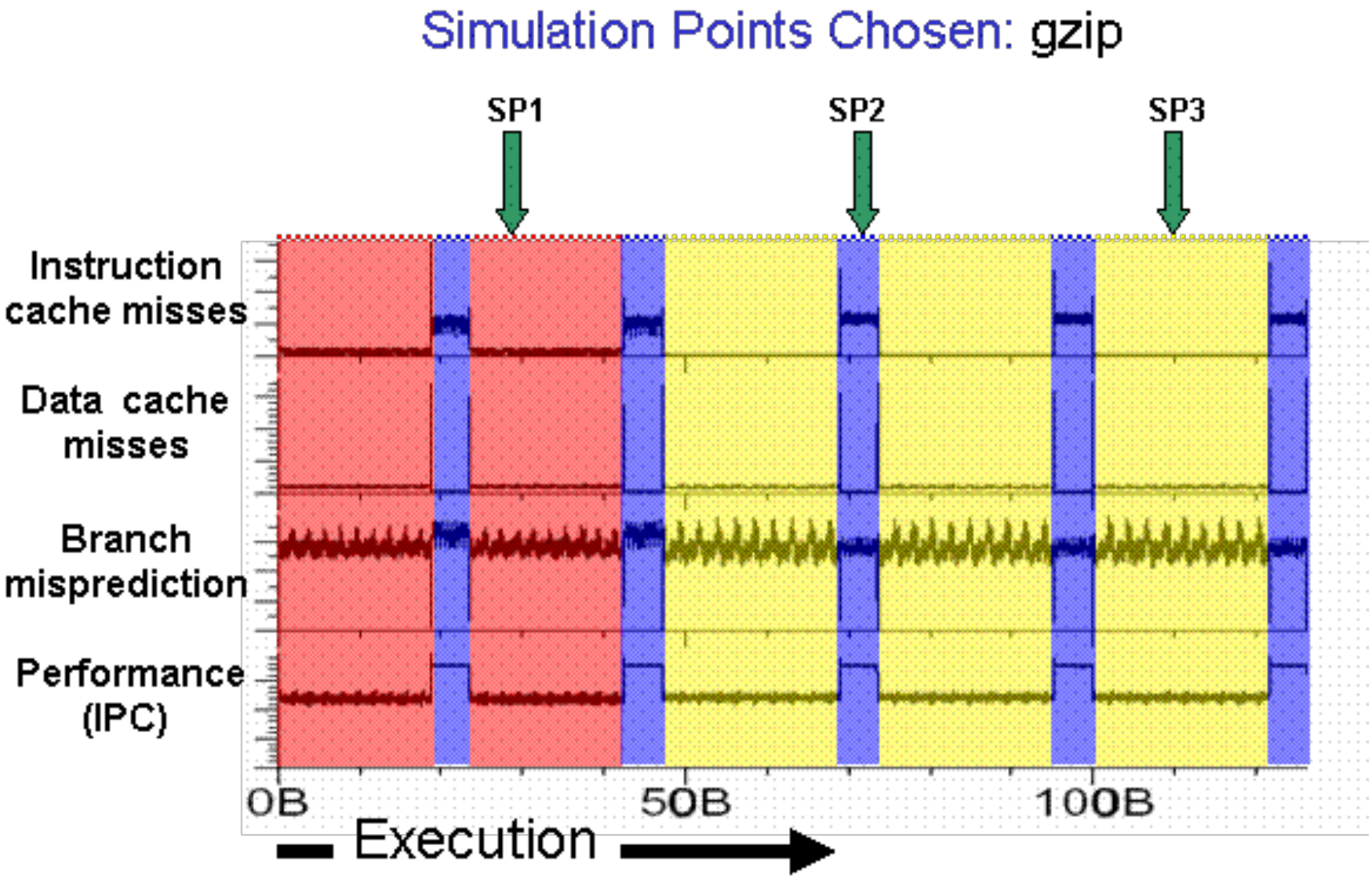


Example: gcc

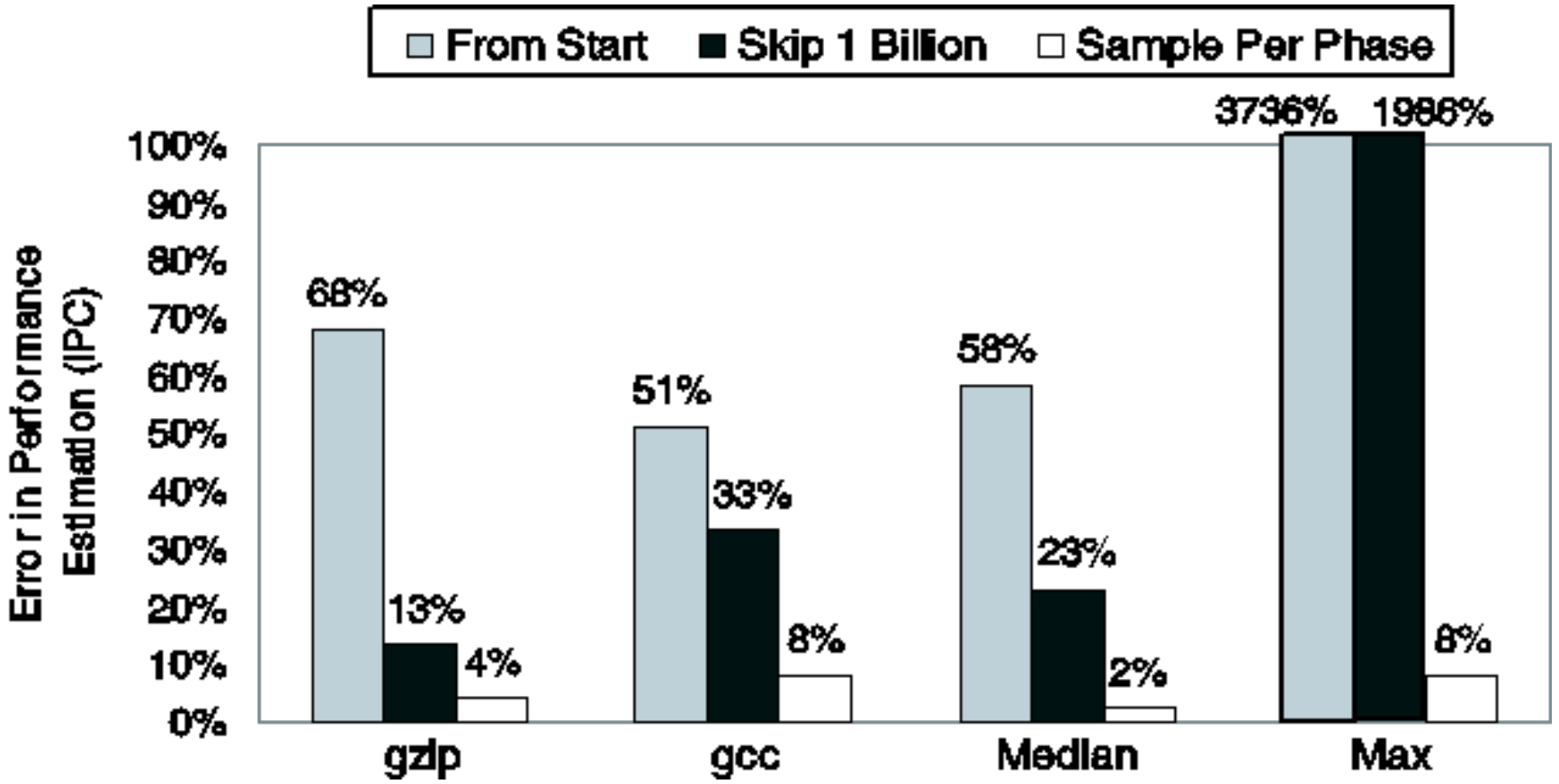
Phases Discovered: gcc



Example: gzip



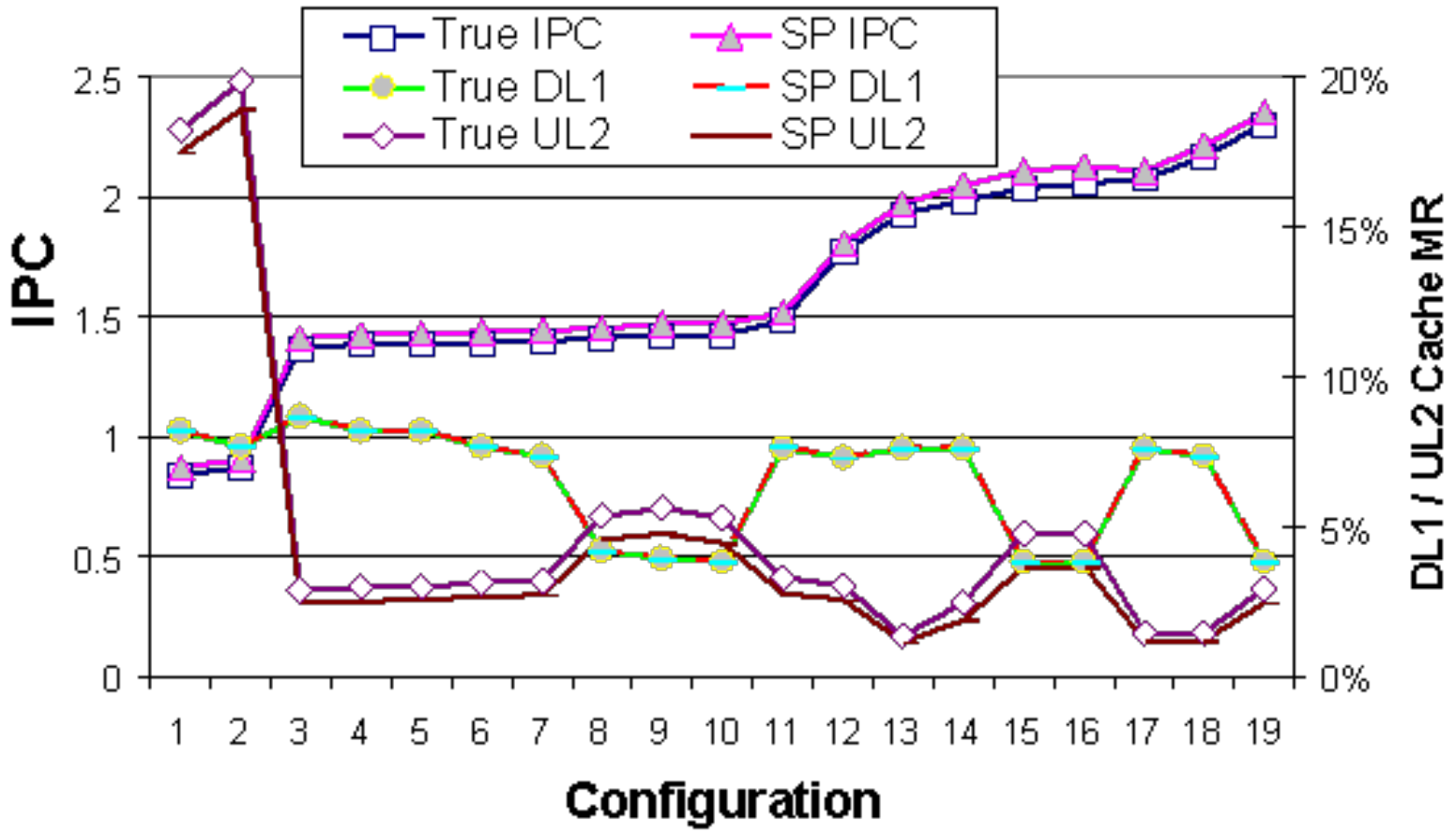
How good is it?



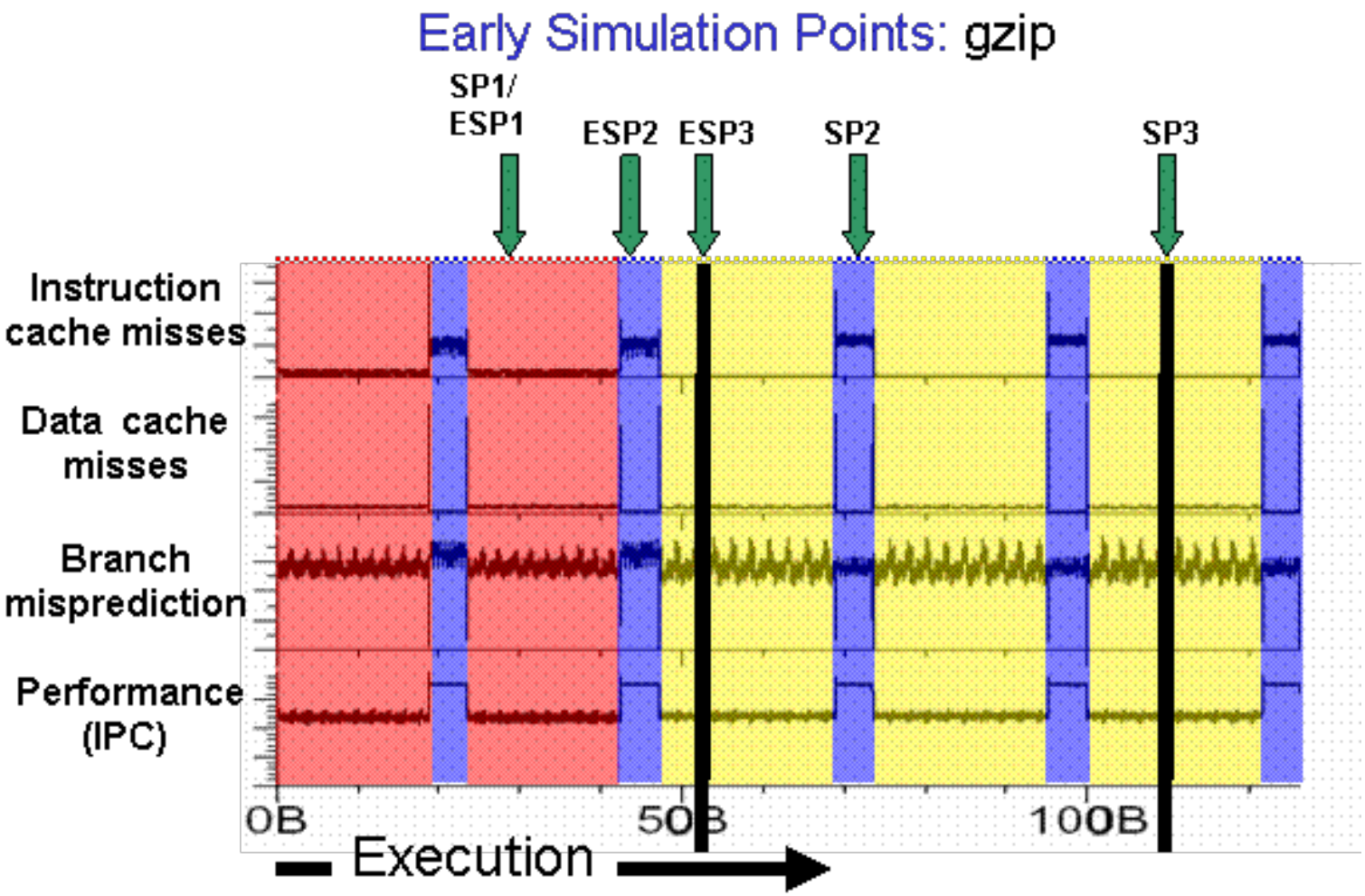
What happened?

- Used 100 million instructions per cluster
- From start
  - Simulates 300 million instructions
- Skip 1 billion
  - Simulates 300 million instructions after the first 1 billion
- Sample per phase
  - Simulates 3 samples for 100 million instructions each

Architecture Independence



Early Simulation Points



Group Exercise

- How would you implement such tool/toolset?