Microarchitecture Independent Workload Characterization

Lecture 4
February 18, 2025

To Dos

Reading for next time

Program 1 submission

Program 2

Algorithm Presentations

Microarchitecture Independent Workload Characterization

Goals of Workload Characterization

- Understand resource demands of emerging applications
- Evaluate whether those resource demands differ from typical resource demands of current applications
- Assess whether different resource demands require changes to future hardware resources

Program Characteristics

μarchitecture-dependent

- Instructions / cycle
- Cache miss rates
- Branch misprediction rates
- TLB miss rates
- ...

μarchitecture-independent

- Instruction mix
- Instruction-level parallelism
- Register traffic
- Working set size
- Data stream strides
- Branch predictability
- Parallelism granularity and degree

Program Characteristics (2)

μarchitecture-dependent

- HW performance counters

μarchitecture-independent

- Binary instrumentation
- Not ISA or compiler independent

Problem with μarch-dependent characteristics

- Can hide underlying program characteristics
- Similar uarch-dependent characteristics do not necessarily imply similar inherent behavior of software
- When run on different HW, might have very dissimilar characteristics

Advantage of μarch-independent characteristics

- Similar uarch-independent behavior implies similar uarch-dependent behavior (and dissimilar implies dissimilar)

Methodology

- Goal: find representative subset of benchmarks, including outliers

- Collect uarch-independent characteristics (p=47) w/ binary instrumentation tool
  - Benchmark = benchmark-input pair
- Reduce dimensionality (most informative characteristics)
  - Principal Component Analysis or Genetic Algorithm
- Cluster benchmarks based on reduced dimensionality

What were some interesting take-aways?

Parallel Performance

Limitations to Performance Improvements

- Amdahl's Law

$$T_{enhanced} = (1 - fraction_{enhanced}) \times T_{unenhanced} + \left(\frac{fraction_{enhanced}}{Speedup_{ehnhancement}} \times T_{unenhanced}\right)$$

- Inherently sequential parts of code
- Overhead in parallel parts of code
  - Communication of data between processes
  - Load balancing
  - Synchronization