Performance Tuning of Computer Systems

Subhasis Banerjee

Outline

Outline

What is Profiling?

- *Profile = a set of data often in graphic form portraying the significant features of something* (Merriam-Webster)
- Focus on dynamic execution (static program analysis is part of software engineering - Formal Method for software)
- Profiling reveals interaction between software and underlying machine architecture
- Indicates areas of improvement $\Rightarrow$ performance tuning

How is Profiling Done?

- Software tools are used to collect profile data
- Tools are often supported by operating system
- Some popular profiling tools:
  - *gprof*, *oprofile*, *valgrind*, *pin*
- Profile is collected during program execution $\Rightarrow$ dynamic profile
- Profile data analysis $\Rightarrow$ Performance Engineering

Which Profile Data is Collected?

- Call graph profile in function level
- Call graph in Basic Block
- Memory performance
- Architectural events *e.g.*, branch misprediction, exception, cache hit/miss
- Monitoring performance counters

Outline

Characterizing Programs: Basic Block

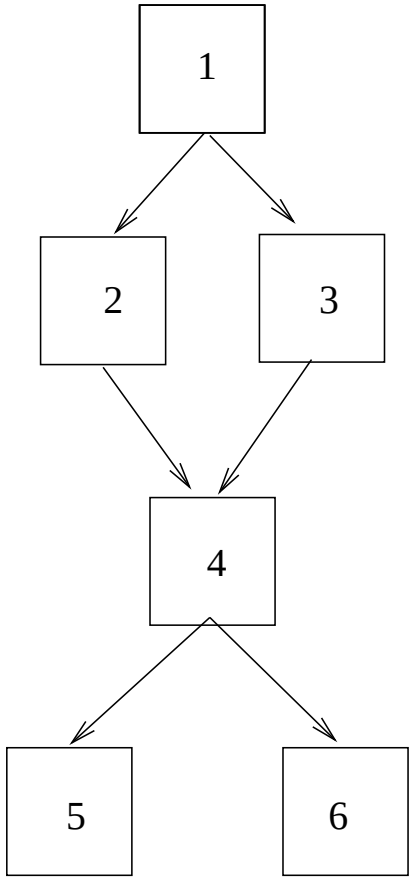
- Programs best viewed in architecture level in intermediate form *e.g.*, in assembly language
- Basic Block (BB) is the section of code sequence which has one entry and one exit point
- BBs are used as building blocks in majority of program analysis tools and optimization policies
- Once a BB is hit all subsequent instructions are executed till it exits at the end of BB
- More general definition - a sequence of instructions where every instruction *dominates* all subsequent following instructions and no other instruction executes between two instructions in the sequence

Algorithm to Identify BB

- Step 1. Identify the leaders (the first instruction of the basic block) in the code. Leaders are instructions which come under any of the following 3 categories:
  - The first instruction
  - The target of a conditional or an unconditional branch instruction
  - The instruction that immediately follows a conditional or an unconditional branch instruction
- Step 2. Starting from a leader, the set of all following instructions until and not including the next leader is the basic block corresponding to the starting leader

Programs in Terms of Directed Graph

- Program is a directed graph with BBs as nodes
- Edges are indicated by branch instructions



Outline

Profilers

- Call graph profiler:
  - Shows the call times and frequency of functions/subroutines.
  - Static or dynamic - depending on degree of accuracy required
  - Dynamic graph can be built fully context sensitive, i.e, for each call of a subroutine graph includes a node with call stack $\Rightarrow$ large memory requirement
  - Widely used in program analysis - Valgrind, gprof, codeviz, doxygen
  - Drawback - slow execution, intrusive
- Event based profiler:
  - Programming languages supports event based profiling (Java, Python, Ruby)
  - Runtime provides various callback to profile agent
  - Customizable in profile collection
  - Drawback - slow execution, intrusive

Profilers

- Statistical Profiler:
  - Usually sampling is done in hardware (program counter)
  - OS interrupts samples the hardware counters
  - Sampling is always lossy, not as accurate as others in collecting data
  - Nearly as fast as the original execution
  - Advantage - non-intrusive
  - Although theoretically other software profilers provides accurate information statistical profiler has exhibited near accurate performance without any loss of execution speed, without modifying program characteristics
  - Tools - AMD CodeAnalysit, Intel VTune, OProfile (open source), gprof, MIPS with JTAG interface
- Instrumenting Program:
  - Instrument code in appropriate place to collect profile
  - Different types of instrumentation - manual, compiler assisted, runtime instrumentation, binary translation
  - gprof (works with instrumentation and sampling) using -pg option with compiler
  - PIN uses runtime instrumentation
  - ATOM (DEC Alpha processors with TRU 64 OS) uses binary translation (obsolete)

Simulation Profiler

- Simulators can be used for detail profiling - slow execution
- Simulation can be of different type: Trace driven simulator, execution driven simulator
- Some simulator can simulate cycle level: Data dependence analysis
- Simulation is done with smaller data input - realistic representation of actual data

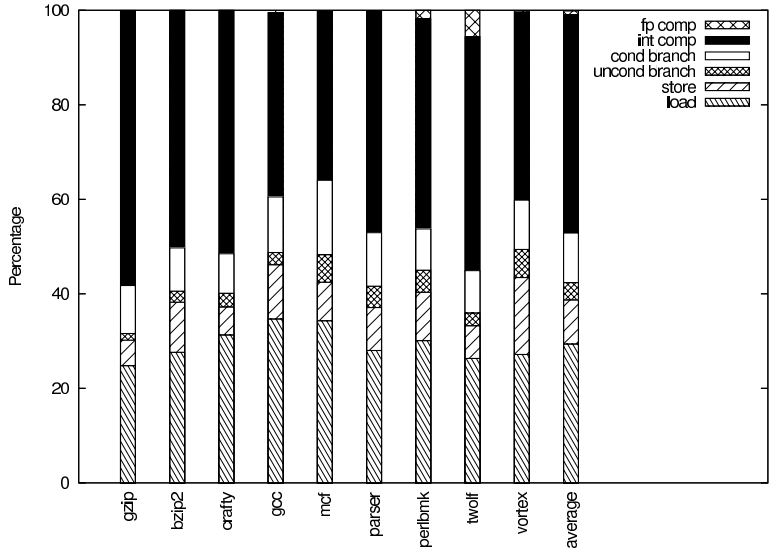
Outline

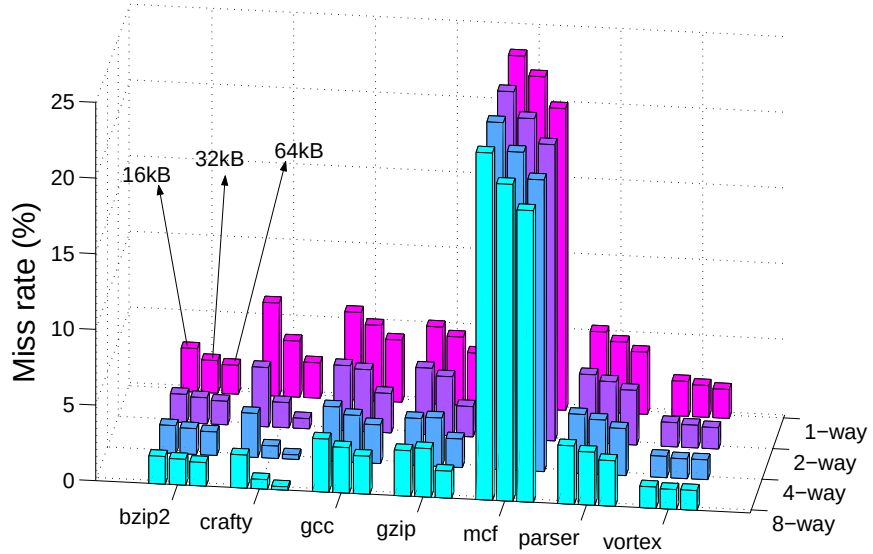
How to Collect Meaningful Data

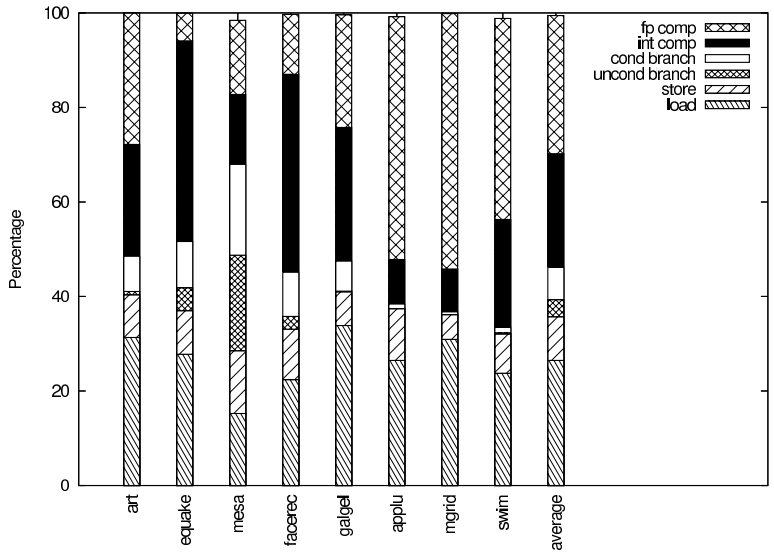
- What is useful data for a profiler?
  - Trace: Collection of instructions and data at runtime, all dynamic instances
  - Events: Architectural events *e.g.*, cache miss, branch misprediction, page fault
  - Counts: Architecture specific metrics *e.g.*, number of instructions retired, number of hit/miss in cache, interrupts/exception
  - Dataflow analysis: Instructions share data for computation - dataflow dependency ensures program order
- What to do with collected data?
  - Analysis of bottleneck in program execution
  - Identify the performance metric: Number of instruction retired in a given time? Power? If multithreaded program - is it adequately parallel?
  - Suggest possible improvement: Automatically tune code? Hint to compiler to generate better code? Architectural support to eliminate/reduce performance bottleneck?

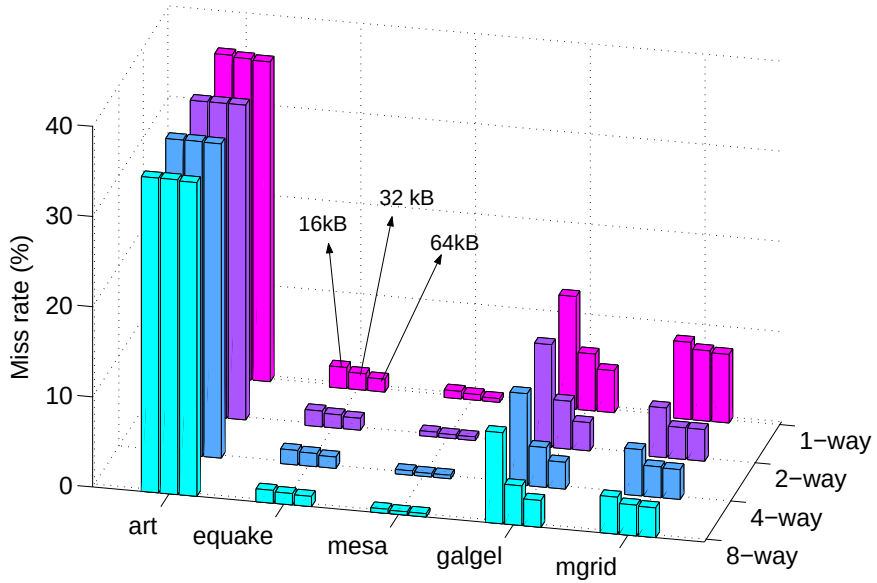
Outline

SPEC2000 Benchmark: Instruction Mix and Cache Profile

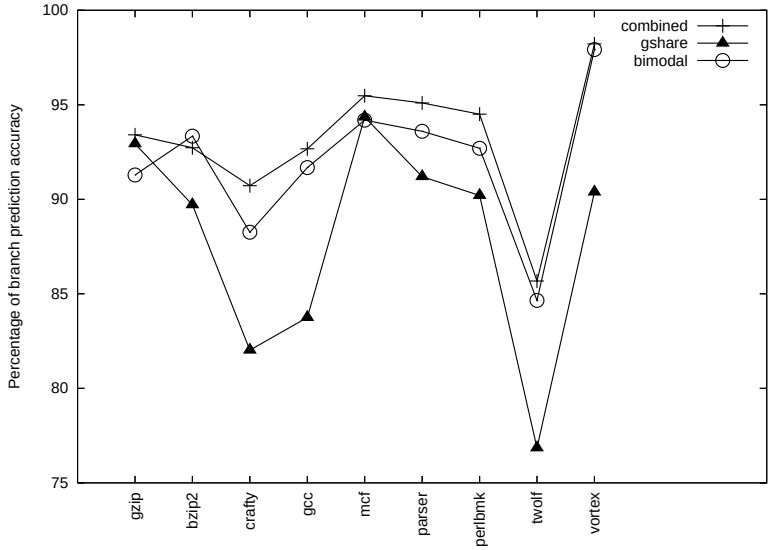


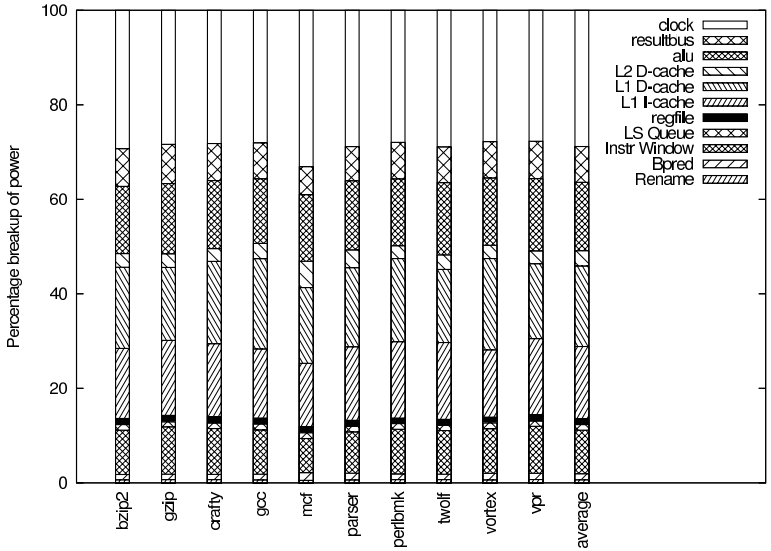


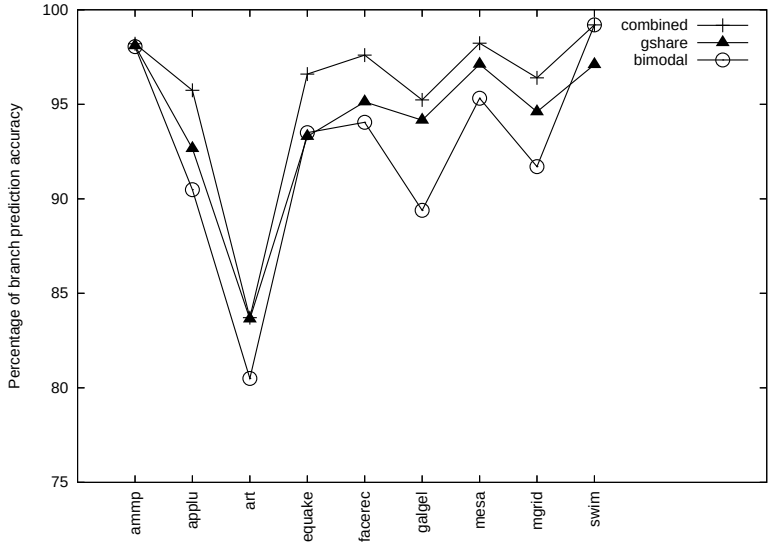


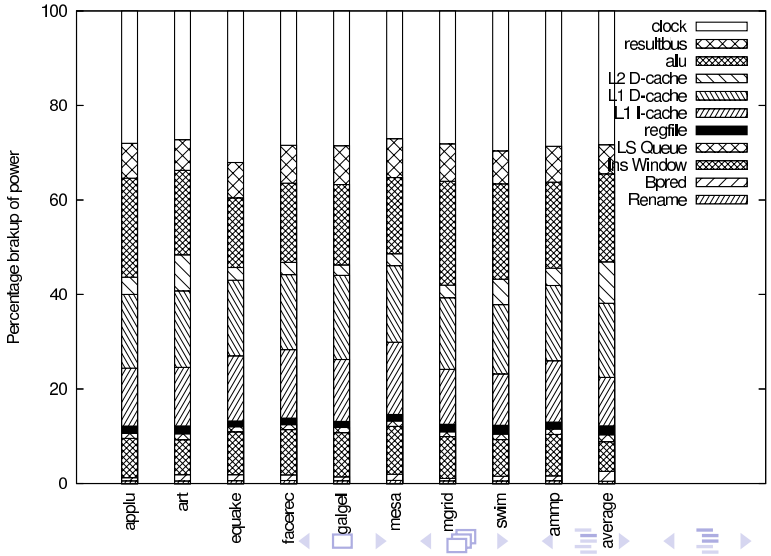


SPEC2000 Benchmark: Branch and Power Profile









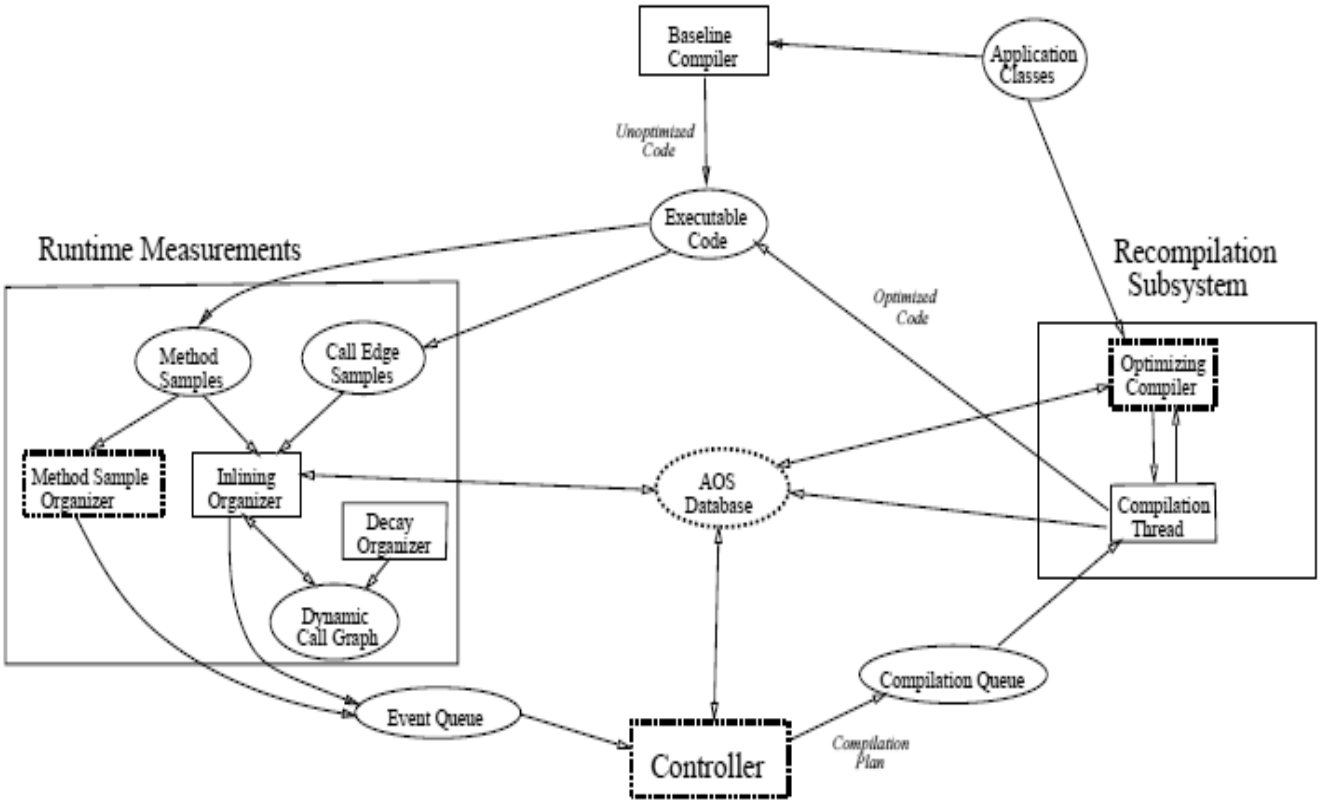
Outline

Hot Spot Detection

- Detection of program hot spots:
  - Program exhibits temporal locality
  - Detect collection of basic blocks which are frequently executed
  - Detection mechanism can be improved by hardware support ([1])
  - Focus on the code section representing hot spot
- Modify hot spot code section aggressively (loop unrolling, instruction fusion, prefetching)

Profile Guided Optimization in Java

- Java Virtual Machines (JVM) employ optimizing compiler
- Profile information is collected at the time of program execution ([2])



Outline

Microarchitecture Optimization

- One architecture never fits all
- Profiling is the key to characterize program behavior and adapt architecture
- Architecture reconfiguration is often done with feedback obtained from program profile
- Runtime configuration is challenging due to several constraint in dynamic profiles: profile overhead, storing time sensitive data

Trace Cache Design

- Traces are sequence of decoded instructions with operand and data
- Trace cache is an instruction cache which stores sequences of basic blocks with decoded instructions and operands with set of branch predictions
- A hit is registered if all branch outcomes are true - decoding of instructions is omitted
- Optimization: Traces can be selectively stored depending of the frequency of execution of a given trace

Power Optimization: Adaptive Issue Queue Design

- Issue queue is one of the power hungry resource
- Size of issue queue can be selectively enabled / disabled depending on available parallelism in the code section
- A profiler can indicate degree of parallelism in the code section

Summary

- Software and hardware optimizations are primarily guided by profiling information
- Future architecture trend: many simple on on one chip $\Rightarrow$ requires scalable solution to extract thread / process level parallelism from program
- Profiler will play important role in defining model of tuning compiler and architecture
- Hardware support extends the capability of compiler to generate efficient code
- *Autotuner*: Traditional compiler may be replaced by feedback driven autotuner.


M. C. Martin, C. George, J. Gyllenhaal, and W. Hwu, "A hardware driven profiling scheme for identifying program hot spots to support runtime optimization," Proc. of the Intl. Symp. on Computer Architecture, 1999.
21


M. Arnold, M. Hind, and B. G. Ryder, "Online feedback-directed optimization of java," in *OOPSLA '02: Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, 2002, pp. 111–129.
22

THANK YOU

QUESTIONS ?

BACKUP SLIDES