ECE 571 – Advanced Microprocessor-Based Design Lecture 1

Vince Weaver

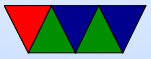
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

2 September 2014

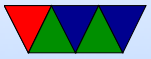
Introduction

- Distribute and go over syllabus
- Talk about the class



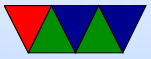
Advanced Microprocessor Based Design

- Advanced "Microprocessor Based Design"
- "Advanced Microprocessor" Based Design



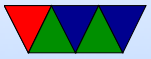
What is an Advanced Microprocessor?

- Desktop.
- Server.
- Embedded.
- They are all converging.

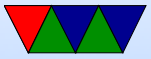


Moore's Law

- Memory Wall
- Power Wall
- Tiny tiny transistors
- More and More Cores
- Something's Got To Give

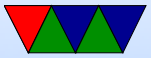


Introduction to Performance Analysis



What is Performance?

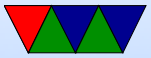
- Getting results as quickly as possible?
- Getting *correct* results as quickly as possible?
- What about Budget?
- What about Development Time?
- What about Hardware Usage?
- What about Power Consumption?



Motivation for HPC Optimization

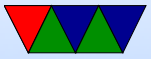
HPC environments are expensive:

- Procurement costs: ∼$40 million
- Operational costs: ∼$5 million/year
- Electricity costs: 1 MW / year ∼$1 million
- Air Conditioning costs: ??

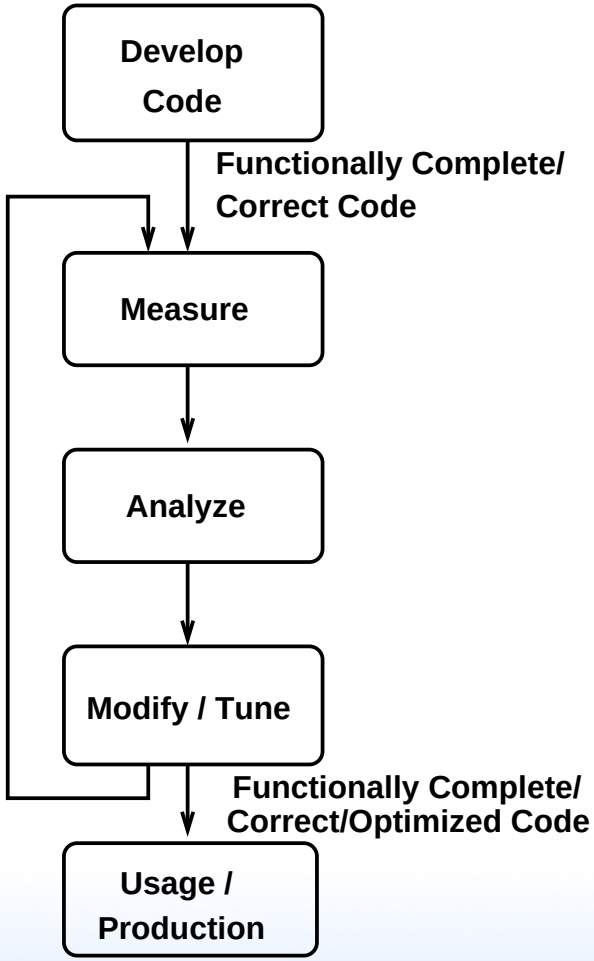


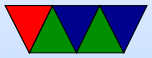
Know Your Limitation

- CPU Constrained
- Memory Constrained (Memory Wall)
- I/O Constrained
- Thermal Constrained
- Energy Constrained



Performance Optimization Cycle



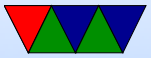


Wisdom from Knuth

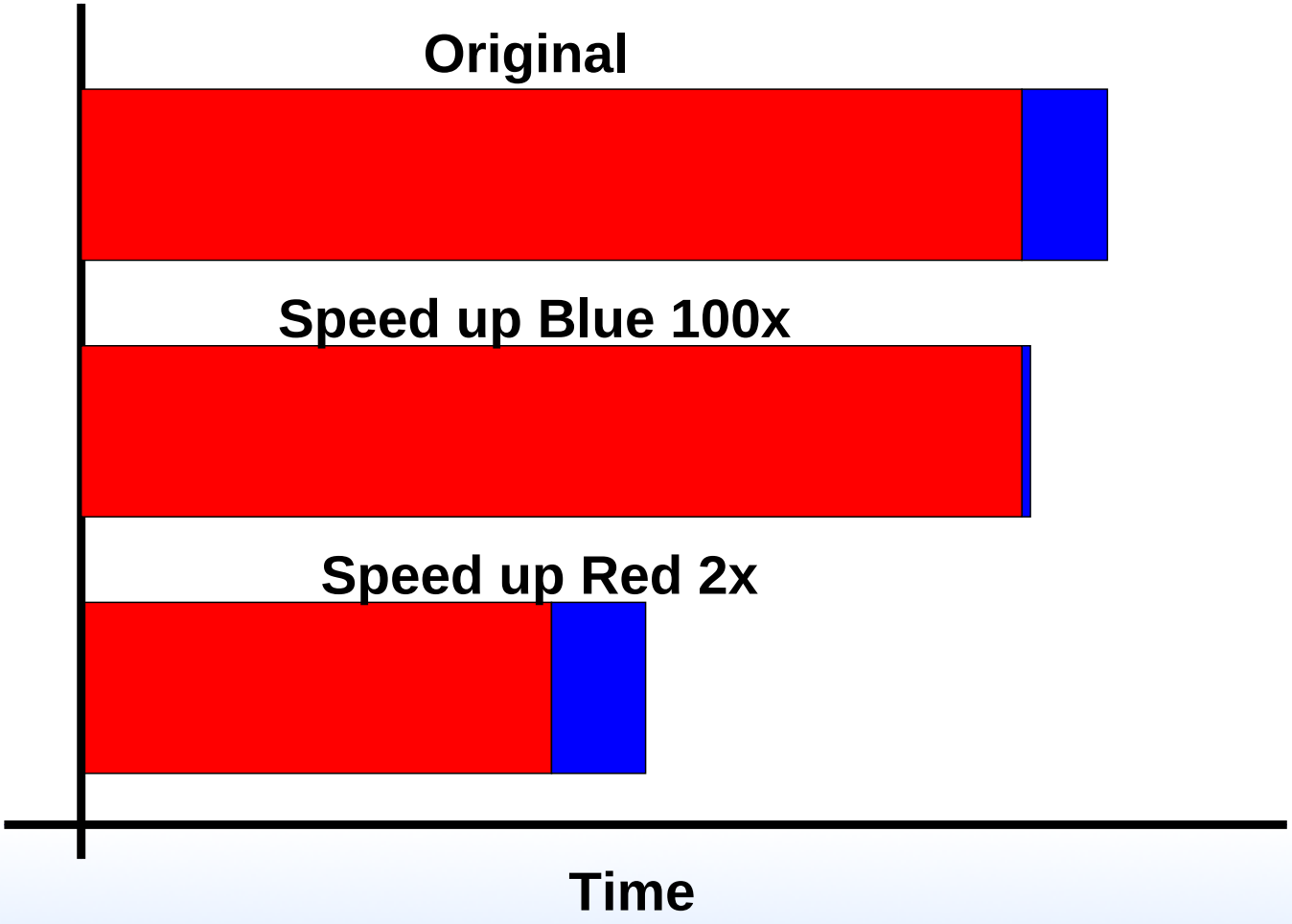
"We should forget about small efficiencies, say about 97% of the time:

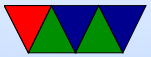
premature optimization is the root of all evil.

Yet we should not pass up our opportunities in that critical 3%. A good programmer will not be lulled into complacency by such reasoning, he will be wise to look carefully at the critical code; but only after that code has been identified" — Donald Knuth



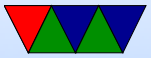
Amdahl's Law



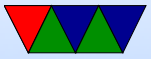


Gathering Performance Info

- User Level (instrumentation)
- Kernel Level (kernel metrics)
- Hardware Level (performance counters)



User Level: Profiling vs Tracing



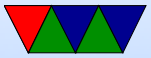
Profiling and Tracing

Profiling

- Records aggregate performance metrics
- Number of times routine invoked
- Structure of invocations

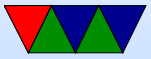
Tracing

- When and where events of interest took place
- Time-stamped events
- Shows when/where messages sent/received



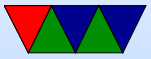
Profiling Details

- Records summary information during execution
- Usually Low Overhead
- Implemented via **Sampling** (execution periodically interrupted and measures what is happening) or **Measurement** (extra code inserted to take readings)



Tracing Details

- Records information on significant events
- Provides timestamps for events
- Trace files are typically *huge*
- When doing multi-processor or multi-machine tracing, hard to line up timestamps



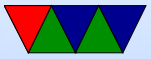
Performance Data Analysis

Manual Analysis

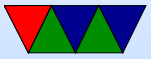
- Visualization, Interactive Exploration, Statistical Analysis
- Examples: TAU, Vampir

Automatic Analysis

- Try to cope with huge amounts of data by automatic analysis
- Examples: Paradyn, KOJAK, Scalasca, Perf-expert

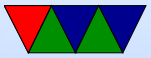


Software Tools for Performance Analysis



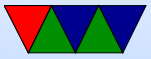
Simulators

- Architectural Simulators
- Can generate traces, profiles, or modeled metrics
- Slow, often 1000x or more slower
- Not real hardware, only a model
- Did I mention, slow?
- m5, gem5, simplescalar, etc



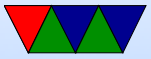
Dynamic Binary Instrumentation

- Pin, Valgrind (cachegrind), Qemu
- Still slow (10-100x slower)
- Can model things like cache behavior (can model parameters other than system running on)
- Complicated fine-tuned instrumentation can be created
- Architecture availability – Pin (no longer ARM), Valgrind, Qemu most architectures, hardest to use



Compiler Profiling

- gprof
- gcc -pg
- Adds code to each function to track time spent in each function.
- Run program, gmon.out created. Run "gprof executable" on it.
- Adds overhead, not necessarily fine-tuned, only does



time based measurements.

- Pro: available wherever gcc is.