ECE 571 – Advanced Microprocessor-Based Design Lecture 1

Vince Weaver

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

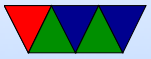
vincent.weaver@maine.edu

19 January 2016

Introduction

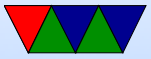
- Distribute and go over syllabus

  http://web.eece.maine.edu/~vweaver/classes/ece571_2016s/ece571_2016s.pdf



Advanced Microprocessor Based Design

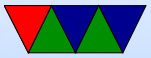
- *NOT* a direct continuation of ECE471 (Embedded Systems) No blinking LEDs on embedded boards.
- Power and Energy concerns on modern systems.
- Will involve some computer architecture. Don't worry if not a Computer Engineer, will try to review completely.
- Will involve reading some papers.
- Will involve logging into Linux boxes and running experiments.



Advanced Microprocessor Based Design

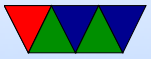
What is an Advanced Microprocessor?

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



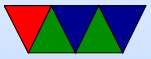
Moore's Law

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



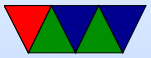
What do people want out of a Microprocessor?

- Performance?
- How do you analyze performance?



What is Performance?

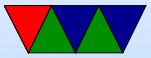
- Getting results as quickly as possible?
- Getting *correct* results as quickly as possible?
- What about Budget?
- What about Development Time?
- What about Hardware Usage?
- What about Power Consumption?



Motivation for HPC Optimization

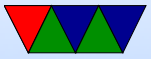
HPC environments are expensive:

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

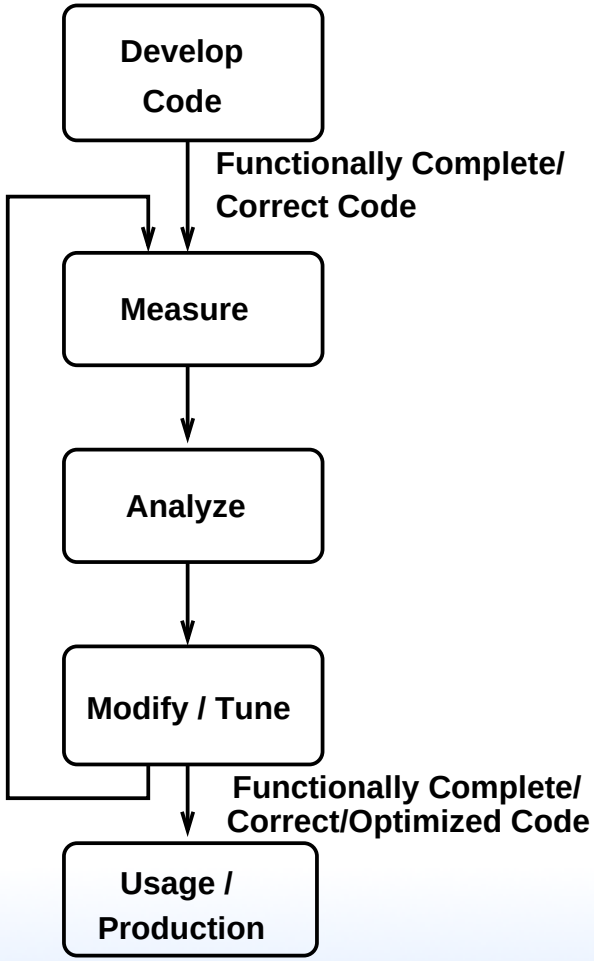


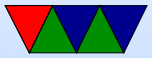
Know Your Limitation

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



Performance Optimization Cycle



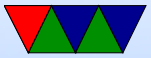


Wisdom from Knuth

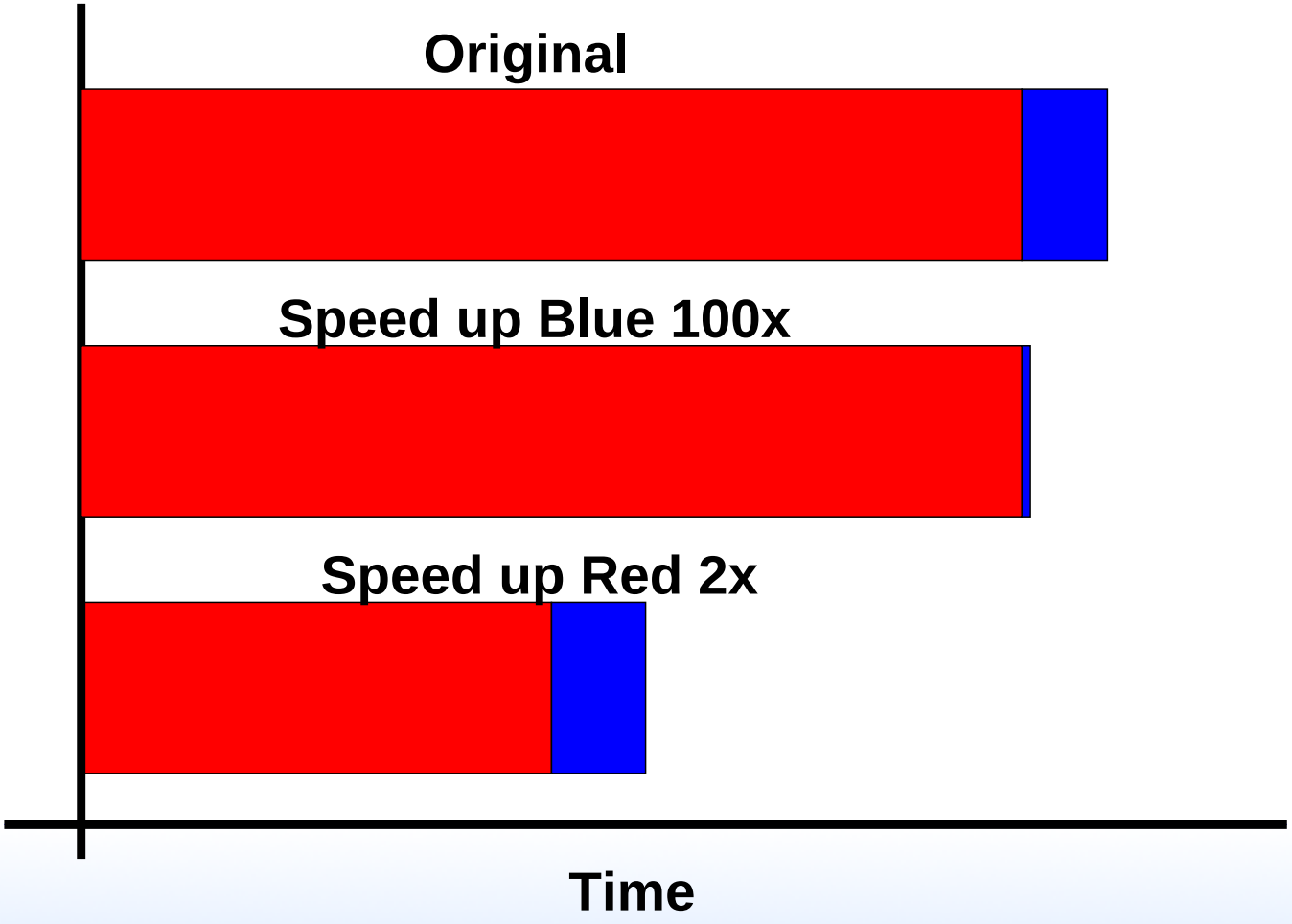
"We should forget about small efficiencies, say about 97% of the time:

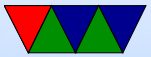
premature optimization is the root of all evil.

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



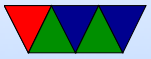
Amdahl's Law





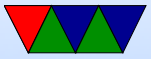
Gathering Performance Info

- Aggregate counts (total instructions, total cycles, etc)
  Actual measurements: `perf`, `time`
  DBI measurements: `valgrind`, `qemu`
  Simulators: `gem5`, `simplescalar`
- Sampled counts – periodically interrupt program, note the instruction pointer. Can use info to statistically determine which part of code where most time (or other metric) is spent
  hardware: `perf`



DBI: `valgrind`
compiler: `gprof`

- Tracing – gather a record of every event (instruction?) that is executed. Can then replay this trace through various tools for analysis.



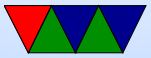
Performance Data Analysis

Manual Analysis

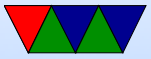
- Visualization, Interactive Exploration, Statistical Analysis
- Examples: TAU, Vampir

Automatic Analysis

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

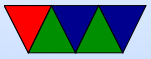


Software Tools for Performance Analysis



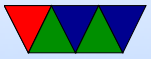
Simulators

- 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



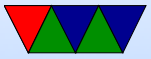
Dynamic Binary Instrumentation

- 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



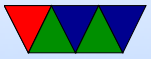
Compiler Profiling

- 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

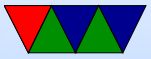


time based measurements.

- Pro: available wherever gcc is.



time



perf