Performance Analysis

EE380, Fall 2015

Hank Dietz

http://aggregate.org/hankd/

UNIVERSITY OF KENTUCKY

Why Measure Performance?

- Performance is important
- Identify HW/SW performance problems
- Compare & choose wisely
    - Which system configuration is better?
    - Which ISA is better?
    - Which ISA implementation is better?
- Expose significant issues, ignore others

What Does Performance Measure?

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

Measures of Computer Performance

- What to measure?
  - Execution time for application
  - Power / temperature / battery life
  - Reliability / availability
  - Cost for acceptable functionality
  - Size
- Measure what matters to you...

Measures of Computer Performance

- Response Time & Throughput
  - Time to complete an operation
  - Jobs completed per unit time
  - Often can trade one for the other
- Performance(X) = 1/ExecutionTime(X)
- X is Performance(X)/Performance(Y) times faster than Y, also:

  ExecutionTime(Y)/ExecutionTime(X)

For Whom The Clock Ticks

- Posix uses real, user, system time
- Real "Wall Clock" time always ticks
- CPU time ticks only when CPU is yours
  - User time while in your code
  - System time while in OS code for you
  - Multiplied by #PEs in multiprocessors
- I/O time not reported under Posix

What Is The Clock?

- Not as simple as you think...
- Used to count AC zero crossings in SW
- Legacy of the IBM PC:
  - Clock chip counts seconds (w/battery)
  - Counter/timer chip @ 1.193181 MHz
- Processor tick count performance register
- Borrowed clocks: NTP, PTP, & GPS
- Jiffy is system interrupt interval (1-10ms)
- Posix counts seconds since 1970, but knows timezones, leaps, etc.

Running What program?

- Different program, different performance
- Application (all that really matters!)
- "Toy" program
- Benchmark: representative application
- Micro Benchmark: tests a certain feature
- Synthetic Benchmark: a program written solely to perform like a particular application, but doing nothing useful
- Benchmark Suite: multiple benchmarks

Common Metrics

- Application/benchmark time for specific data: e.g.: Quake updates/s
- LIPS: Logical Inferences/s
- FLOPS: Floating-Point OPerations/s
- MB/s, Mb/s: MegaBytes/Megabits per s
- MIPS: Millions of Instructions/s
- MOPS: “” OPerations/s
- Hz: clock cycles/s
- CPI: Cycles Per Instruction
  IPC: Instructions Per Cycle

CPI

- Clock ticks at a (mostly) constant rate
- Can express program runtime as:

$$\frac{\text{Seconds}}{\text{Program}} = \frac{\text{Cycles}}{\text{Program}} * \frac{\text{Seconds}}{\text{Cycle}}$$

$$\frac{\text{Seconds}}{\text{Program}} = \frac{\text{Cycles}}{\text{Program}} \; / \; \text{Frequency}$$

Cycles / Program?

- Programs are made of instructions
- Can use CPI to compute:

$$\frac{\text{Cycles}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} * \text{CPI}$$

$$\frac{\text{Cycles}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} / \text{IPC}$$

All Instructions Alike?

- Different instruction type, different CPI
- Can sum over types separately:

$$\sum \left(\frac{\text{Instructions}}{\text{Program}} * \text{CPI} \right)$$

Parameters

- Instruction types can be “classes”
- Instructions / Program
  - Expected execution counts
  - % or ratios for relative performance
- CPI
- Clock period same for everything
  (analyze separately for each clock rate
  If the processor dynamically throttles)

An Example

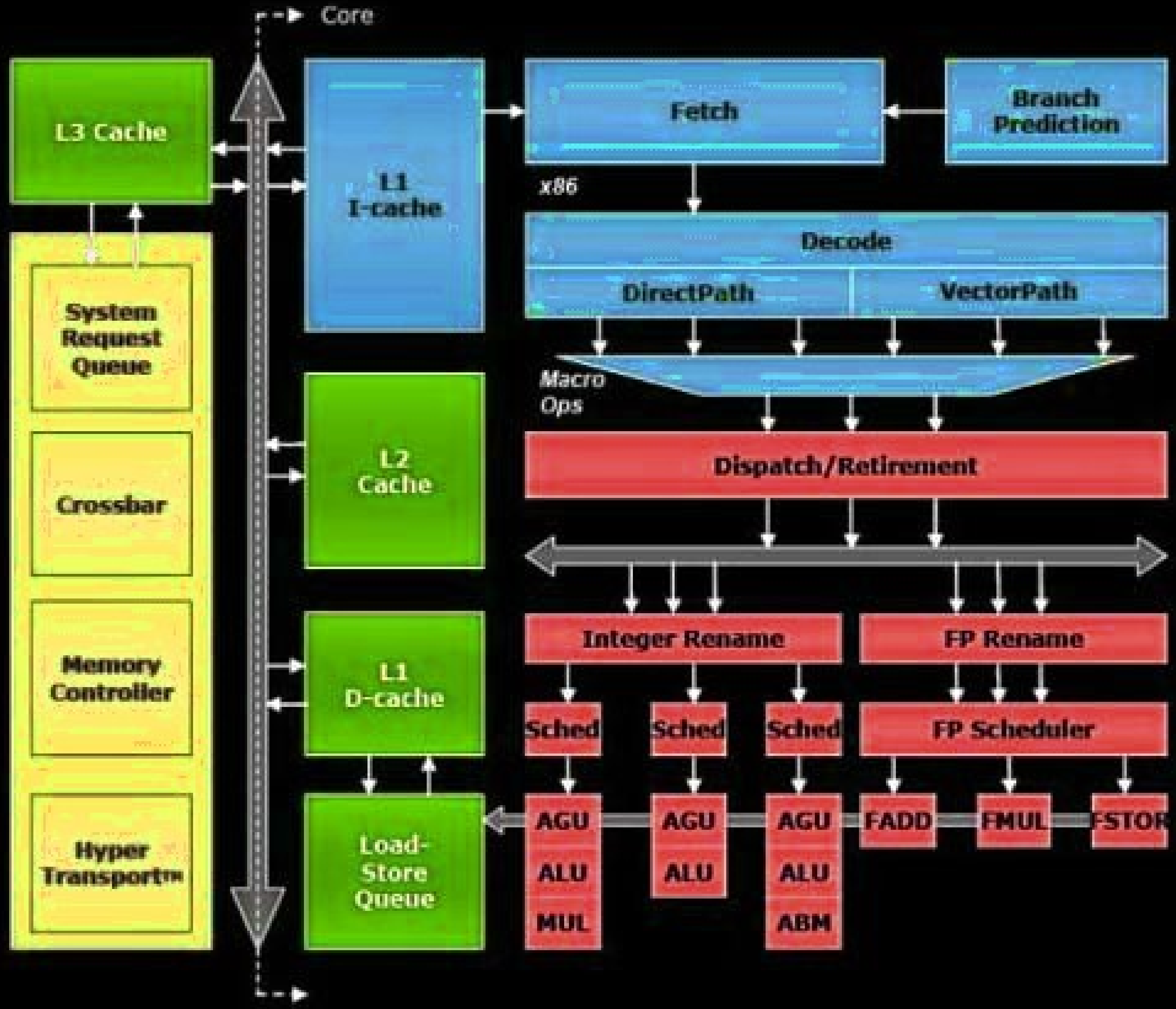
Instruction Type	Execution Count	CPI	Clock Period
A	20	10	10ns
B	10	30	10ns

- This program takes: ((20*10)+(10*30))*10ns = 5us
- What can be changed to make it 4us?

A Little Disclaimer...

That CPI analysis assumes sequential execution of instructions, but most modern processors are parallel in various ways...

the model is still useful, but approximate, using 1/IPC to approximate CPI; it also works to analyze sequential portions of a design


Core
L3 Cache
L1
I-cache
Fetch
Branch
Prediction
x86
Decode
DirectPath
VectorPath
Macro
Ops
System
Request
Queue
Crossbar
Memory
Controller
Hyper
Transport™
L2
Cache
Dispatch/Retirement
L1
D-cache
Integer Rename
FP Rename
Sched
Sched
Sched
FP Scheduler
Load-
Store
Queue
AGU
ALU
MUL
AGU
ALU
AGU
ALU
ABM
FADD
FMUL
FSTOR

What Effects What?

	Instruction Count	CPI	Clock Rate
Program (Algorithm)			
Compiler			
ISA			
Impl. Arch.			
VLSI			

What Effects What?

	Instruction Count	CPI	Clock Rate
Program (Algorithm)	Yes!	Indirectly...	No!
Compiler	Yes!	Indirectly...	Power?
ISA	Yes!	Yes!	Indirectly...
Impl. Arch.	uOps?	Yes!	Yes!
VLSI	No!	Indirectly...	Yes!

How Does A Change Affect Design?

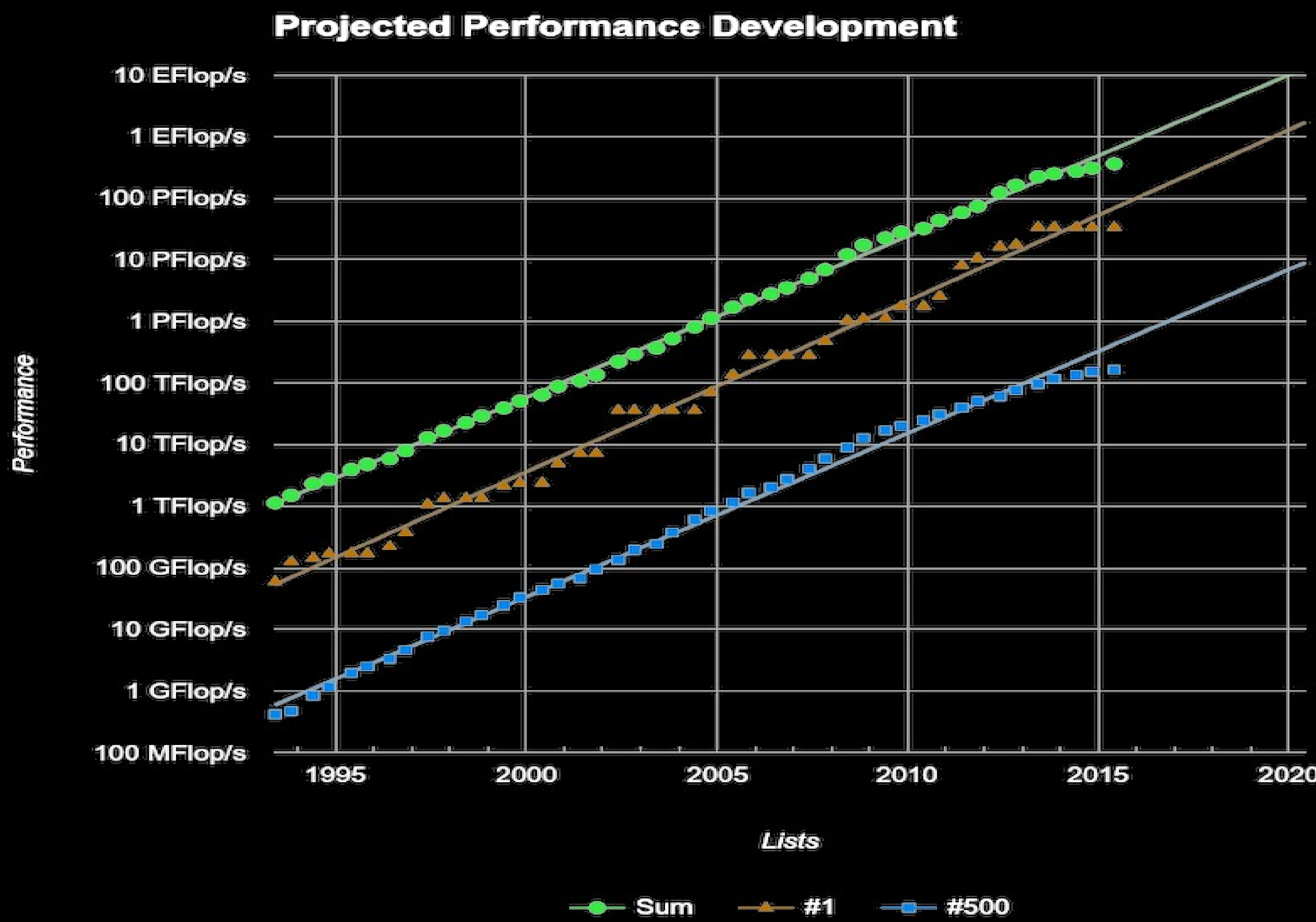
- For a particular application
- For a particular compiler
- For a particular ISA
- For a particular implementation arch.
- For a particular VSI technology
- Etc.

Amdahl's Law

- If 1/N time is not affected by a change, the best possible speedup is only N
- Originally for sequential overhead in parallel code, but applies for any change

Suppose a program spends 80% of its time doing multiplies... you can't get more than a 5X speedup by improving only multiplies!

A Lesson From `Top500.Org`



UNIVERSITY OF KENTUCKY

Summary

- Most performance numbers not relevant; measure what you care about
- Relate performance to causes
- Best designs usually make everything (and hence nothing) the bottleneck
- Stuff is getting better fast... Don't base design decisions on now, but on when you will need/market it