Evaluating Computers: Bigger, better, faster, more?

Key Points

- What does it mean for a computer to be good?
- What is latency? bandwidth?
- What is the performance equation?

What do you want in a computer?

- Low response time
  - s
- High throughput
  - work/s
- High RAS
  - Reliability (avoid and detect faults)
  - Availability: 5 nines
  - Servicability
  - Mean time to data loss
  - MTBF
  - Hardware
  - Software
- Low power consumption
  - W -- power
  - MIPS/W
  - Idle power
  - Wall power efficiency
  - Energy
  - Energy * Delay
  - Energy * Energy * Delay
  - Peak Power
  - Hot Spot Power
- Security
  - p0wns/week
- Cheap
  - GOPS/$ (integer/sec)
  - FLOPS/$ (FP /sec)
  - B/$
  - MIPS/$
  - Ops/$
- Lookin' good
- Small
- Lots of storage
- Usable
  - Happiness
- Fast IO
  - Ops/sec
  - GB/sec
- High FPS @ 4000x6000
- Clairvoyance
- Standards-based
  - s*$/solution

What do you want in a computer?

- Low latency -- one unit of work in minimum time
  - 1/latency = responsiveness
- High throughput -- maximum work per time
  - High bandwidth (BW)
- Low cost
- Low power -- minimum joules per time
- Low energy -- minimum joules per work
- Reliability -- Mean time to failure (MTTF)
- Derived metrics
  - responsiveness/dollar
  - BW/$
  - BW/Watt
  - Work/Joule
  - Energy * latency -- Energy Delay Product
  - MTTF/$

Latency

- This is the simplest kind of performance
- How long does it take the computer to perform a task?
  - The task at hand depends on the situation.
- Usually measured in seconds
- Also measured in clock cycles
  - Caution: if you are comparing two different system, you must ensure that the cycle times are the same.

Where latency matters

- Application responsiveness
  - Any time a person is waiting.
  - GUIs
  - Games
  - Internet services (from the users perspective)
- "Real-time" applications
  - Tight constraints enforced by the real world
  - Anti-lock braking systems -- "hard" real time
  - Manufacturing control
  - Multi-media applications -- "soft" real time
- The cost of poor latency
  - If you are selling computer time, latency is money.

Latency and Performance

- By definition:
- Performance = 1/Latency
- If Performance(X) > Performance(Y), X is faster.
- If Perf(X)/Perf(Y) = S, X is S times faster than Y.
- Equivalently: Latency(Y)/Latency(X) = S

- When we need to talk about specifically about other kinds of "performance" we must be more specific.

Making Meaningful Comparisons

Latency = Instructions * Cycles/Instruction * Seconds/Cycle

- Meaningful CPI exists only:
  - For a particular program with a particular compiler
  - ....with a particular input.
- You MUST consider all 3 to get accurate latency estimations or machine speed comparisons
  - Instruction Set
  - Compiler
  - Implementation of Instruction Set (386 vs Pentium)
  - Processor Freq (600 Mhz vs 1 GHz)
  - Same high level program with same input
- "wall clock" measurements are always comparable.
  - If the workloads (app + inputs) are the same

Benchmarks: Standard Candles for Performance

- It's hard to convince manufacturers to run *your* program (unless you're a BIG customer)
- A benchmark is a set of programs that are representative of a class of problems.
- To increase predictability, collections of benchmark applications, called *benchmark suites*, are popular
  - "Easy" to set up
  - Portable
  - Well-understood
  - Stand-alone
  - Standardized conditions
  - These are all things that real software is not.
- Benchmarks are created by a human (i.e., a political) process. The creators may not have your best interests at heart.

Classes of benchmarks

- Microbenchmark – measure one feature of system
  - e.g. memory accesses or communication speed
- Kernels – most compute-intensive part of applications
  - e.g. Linpack and NAS kernel b'marks (for supercomputers)
- Full application:
  - SpecInt / SpecFP (int and float) (for Unix workstations)
  - Other suites for databases, web servers, graphics,...

Limits on Speedup: Amdahl's Law

- "The fundamental theorem of performance optimization"
- Coined by Gene Amdahl (one of the designers of the IBM 360)
- Optimizations do not (generally) uniformly affect the entire program
  - The more widely applicable a technique is, the more valuable it is
  - Conversely, limited applicability can (drastically) reduce the impact of an optimization.

Always heed Amdahl's Law!!!

It is central to many many optimization problems

How to Summarize Performance

- Arithmetic Mean

$$\frac{1}{n}\sum_{i=1}^{n} Time_i$$

- Weighted Arithmetic Mean

$$\sum_{i=1}^{n} Time_i * Weight_i$$ where the sum of the weights is 1.

- Harmonic Mean

$$\frac{n}{\sum_{i=1}^{n} \frac{1}{Rate_i}}$$

- Geometric Mean

$$\sqrt[n]{\prod_{i=1}^{n} ExecutionTimeRatio_i}$$

Currently in vogue.

Adapted from Brad Calder's Slides.

Summarizing Performance

- Even the unweighted arithmetic mean implies a weighting
  - the longer the running time the larger the impact
- Geometric means of normalized execution times are consistent no matter which machine is faster
  - ratios of geometric means never change, and always give equal weight to all benchmarks
- Geometric mean does not necessarily predict execution time for any mix of the programs

Adapted from Brad Calder's Slides.

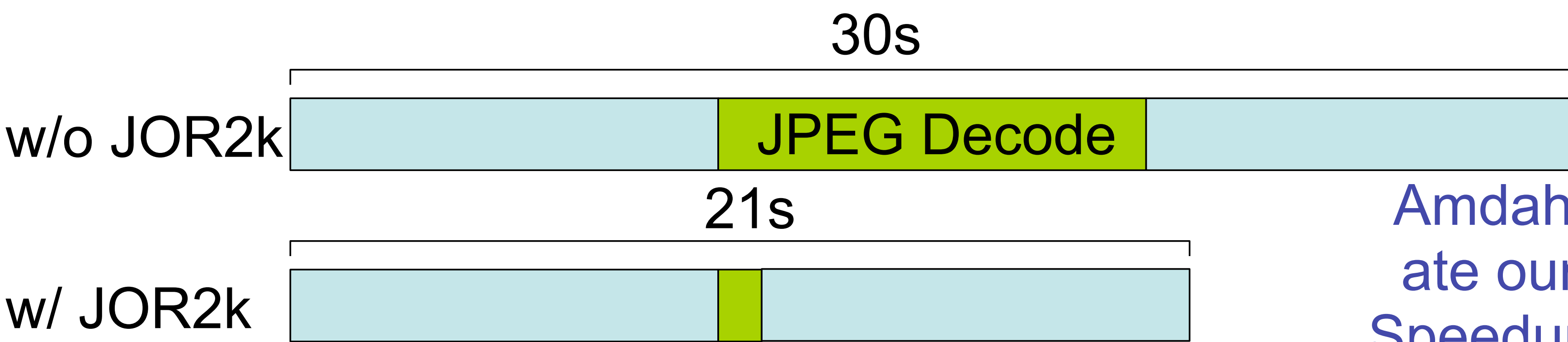
Amdahl's Law in Action

- SuperJPEG-O-Rama2010 ISA extensions **
  - –Speeds up JPEG decode by 10x!!!
  - –Act now! While Supplies Last!

** Increases processor cost by 45%

Amdahl's Law in Action

- SuperJPEG-O-Rama2010 in the wild
- PictoBench spends 33% of it's time doing JPEG decode
- How much does JOR2k help?



Amdahl ate our Speedup!

Performance: 30/21 = 1.4x Speedup != 10x

Is this worth the 45% increase in cost?

Amdahl's Law

- The second fundamental theorem of computer architecture.
- If we can speed up *X* of the program by *S* times
- Amdahl's Law gives the total speed up, S_{tot}

$$S_{tot} = \frac{1}{(x/S + (1-x))}.$$

Sanity check:

$$x = 1 \Rightarrow S_{tot} = \frac{1}{(1/S + (1-1))} = \frac{1}{1/S} = S$$

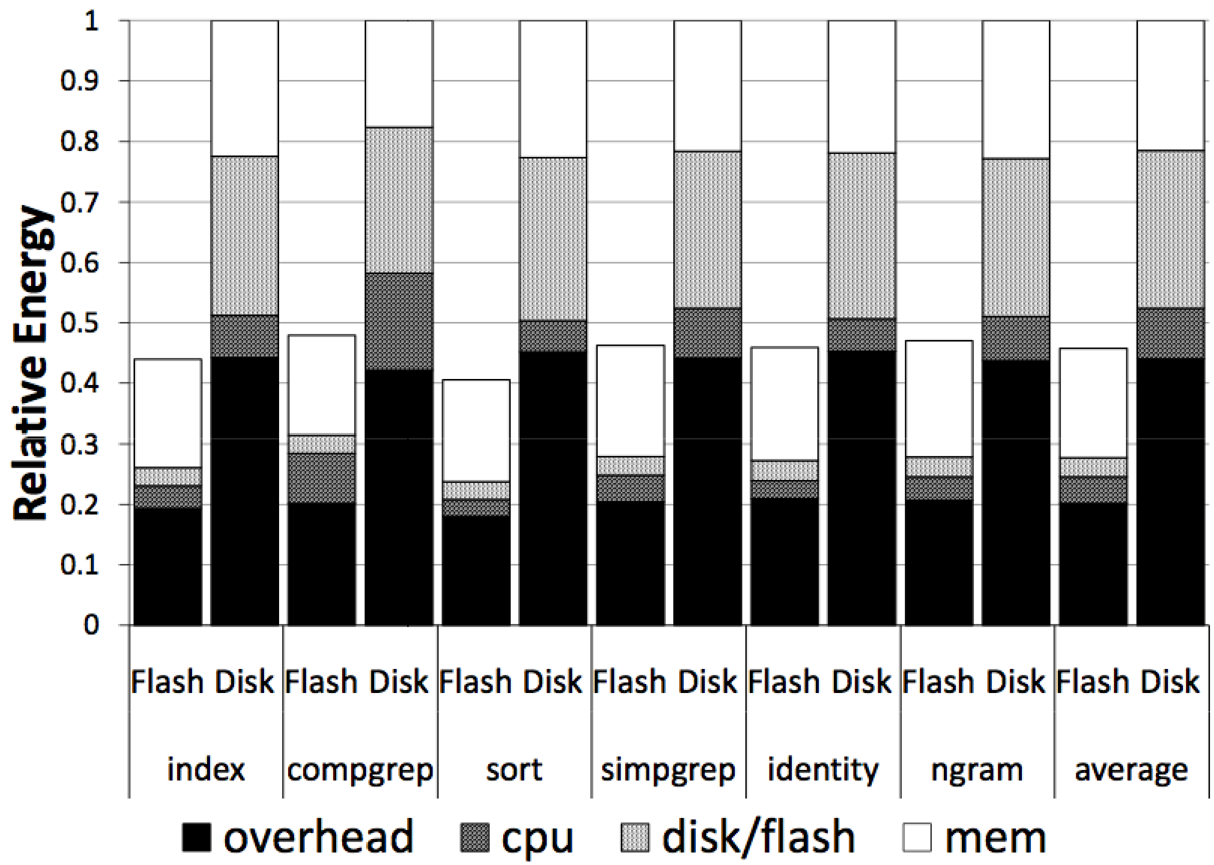
Amdahl's Corollary #1

- Maximum possible speedup, S_{max}

$$S = \textit{infinity}$$

$$S_{max} = \frac{1}{(1-x)}$$

Amdahl’s Law Applies All Over



- SSDs use 10x less power than HDs
- But they only save you ~50% overall.

Amdahl’s Corollary #2

- Make the common case fast (i.e., *x* should be large)!
  - Common == “most time consuming” not necessarily “most frequent”
  - The uncommon case doesn’t make much difference
  - Be sure of what the common case is
  - The common case changes.
- Repeat…
  - With optimization, the common becomes uncommon and vice versa.

Amdahl's Corollary #3

- Benefits of parallel processing
- *p* processors
- *x*% is *p*-way parallelizable
- maximum speedup, S_{par}

$$S_{par} = \frac{1}{(x/p + (1-x))}$$

x is pretty small for desktop applications, even for p = 2

Does Intel's 80-core processor make much sense?

Amdahl's Non-Corollary

- Amdahl's law does not bound slowdown

  $L_{new} = (L_{base} / S) * x + L_{base} * (1-x)$

- L_{new} is linear in $1/S$
- Example: $x = 0.01$ of execution, $L_{base} = 1$
  - $S = 0.001$;
    - $E_{new} = 1000 * L_{base} * 0.01 + L_{base} * (0.99) \sim 10 * L_{base}$
  - $S = 0.00001$;
    - $E_{new} = 100000 * L_{base} * 0.01 + L_{base} * (0.99) \sim 1000 * L_{base}$
- Things can only get so fast, but they can get arbitrarily slow.
  - Do not hurt the non-common case too much!

Latency versus Bandwidth

Plane	DC to Paris	Speed	Passengers	Bandwidth (p-mph)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

- **Time to run the task (ExTime)**
  - **Execution time, response time, latency**
- **Tasks per day, hour, week, sec, ns … (Performance)**
  - **Throughput, bandwidth**

High Speed Data Transfer

- IPv6 Internet 2: 272,400 terabit-meters per second
  - 585GB in 30 minutes over 30,000 Km
  - 9.08 Gb/s

- Comparison: Subaru outback wagon
  - Max load = 408Kg
  - 21Mpg
- MHX2 BT 300 Laptop drive
  - 300GB/Drive
  - 0.135Kg
- 906TB
- Legal speed: 75MPH (33.3 m/s)
- BW = 8.2 Gb/s
- Latency = 10 days
- 241,535 terabit-meters per second