# CSCI-564 Advanced Computer Architecture

## Lecture 2: Performance Metrics

Ismet Dagli

Credit to: Dr. Bo Wu

Colorado School of Mines

# Course Staff

- Instructor: Ismet Dagli (ismetdagli@mines.edu)
  - Lectures Mondays + Wednesdays (4.30pm to 5.45pm)
  - Office hours Monday 6.00-7.00pm or by appointment
- TA: Chang Liu (liuchang@mines.edu)
  - Office hours: Friday 8.45am to 9.45am
  - **For homework & projects**
- 2nd TA Akshit Sharma (akhsitsharma@mines.edu)
  - Office hours: Wednesday 1.00pm to 2.00pm
    Friday 3.00pm to 4.00pm
  - **Again for homework & projects**
- Course Pages: Canvas & Ed Discussion

# What do you want in a computer?

- Quietness (dB)
- Speed
- Preceived speed
  - Responsiveness
- Batter life
- Good lookin'
- Volume
- Dimensions
- Portability
  - Weight
  - Size
- Flexibilty
- Reliability
- Expandability/ Upgradability
- Workmanship
- Memory bandwidth
- Power consumption
- Good support
- Popularity/ Facebook likes
- Thermal performance
- Display quality
- Is it a mac?
- Ergonomics
- FPS
  - Crysis metric
  - But at what Res?
- Sound quality
- Network speed
- Connectivity
  - USB 3.0
  - Thunderbolt
  - HDMI
  - Ethernet
  - Bluetooth
  - Floppy
- Warranty
- Storage capacity
- Storage speed
- Peripherals
  - quality
- bells and whistles
- Price!!!
- Awesome
  - Beiber

# Metrics

# Example: Latency

- Latency is the most common metric in architecture
  - Speed = 1/Latency
  - Latency = Run time
  - “Performance” usually, but not always, means latency
- A measured latency is for some particular task
  - A CPU doesn’t have a latency
  - An application has a latency on a particular CPU

# 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
  - Multi-media applications -- "soft" real time

# Example: Speedup

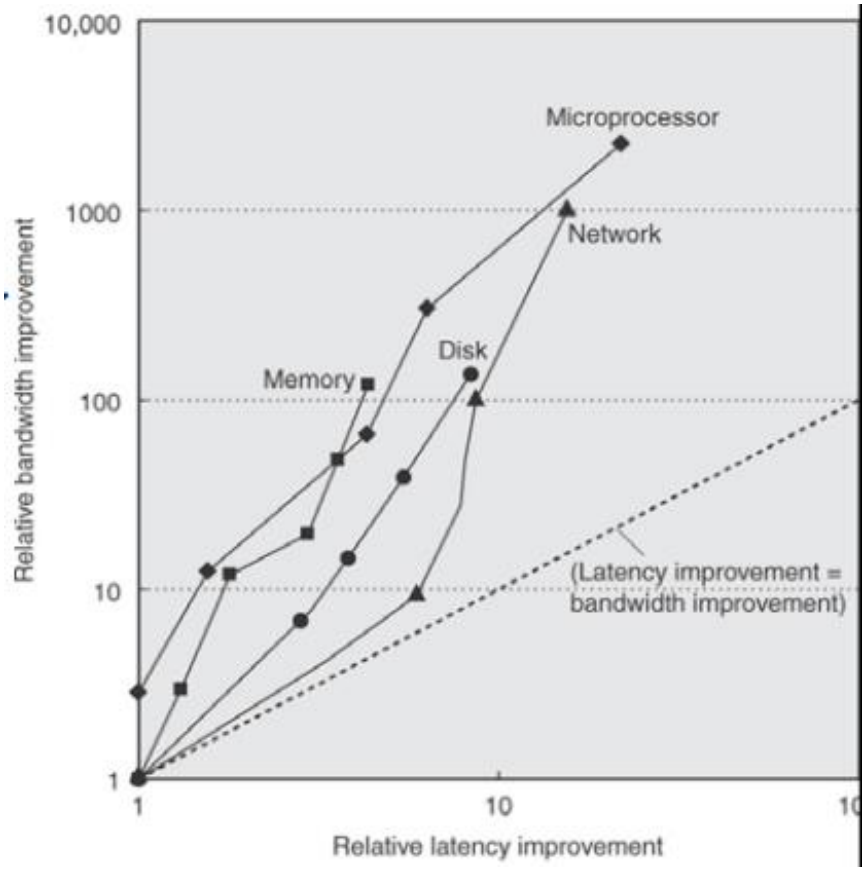
- Speedup is the ratio of two latencies
  - Speedup = Latencyold/Latencynew
  - Speedup > 1 means performance increased
  - Speedup < 1 means performance decreased
- If machine A is 2x faster than machine B
  - LatencyA = LatencyB/2
  - The speedup of B relative to A is 1/2x or 0.5x.
- Speedup (and other ratios of metrics) allows the comparison of two systems without reference to an absolute unit
  - We can say “doubling the clock speed with give 2x speedup” without knowing anything about a concrete latency.
  - It’s much easier than saying “If the program’s latency was 1,254 seconds, doubling the clock rate would reduce the latency to 627 seconds.”

# Example: Throughput

- Throughput (aka Bandwidth)
  - number of tasks completed per unit time
  - independent from the exact total number of tasks
  - important in which scenarios?
    - data center servers (i.e., Youtube or Netflix)
    - high-performance computing

# Latency lags bandwidth

- Bandwidth has outpaced latency across the main computer technologies



# Why?

"There is an old network saying: Bandwidth problems can be cured with money. Latency problems are harder because the speed of light is fixed —- you can't bribe God."

[Anonymous]

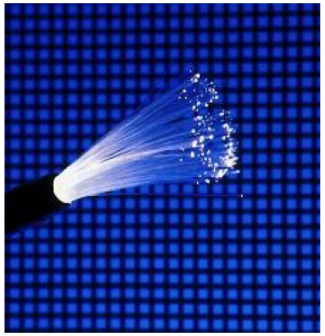
Some jobs are just hard to parallelize!

Dr. Bo Wu

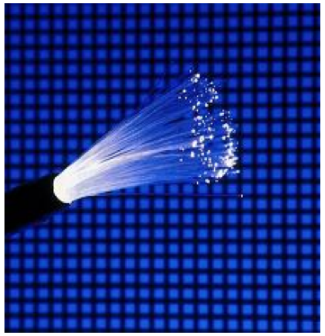
Question: I have two processors A and B. I'm only interested in application C. The latency of running C on A is much lower than that on B. What can I say about the bandwidth difference between A and B?

Answer: absolutely nothing!

# The Internet “Land”-Speed Record

| | Fiber-optic cable |
|---|---|
| |  |
| | State of the art networking medium (sent 585 GB) |
| Latency (s) | 1800 |
| BW (GB/s) | 1.13 |

# The Internet “Land”-Speed Record

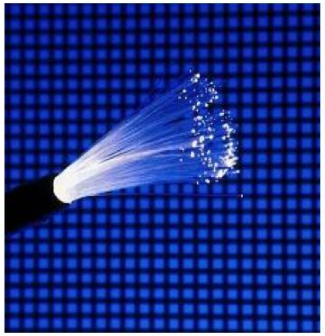
| Fiber-optic cable |
| --- |
|  |
| State of the art networking medium (sent 585 GB) |

| | |
| --- | --- |
| Latency (s) | 1800 |
| BW (GB/s) | 1.13 |

"Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway."

-- Andrew S. Tanenbaum

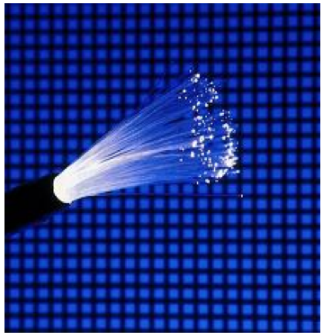
# The Internet “Land”-Speed Record

| Fiber-optic cable |
| --- |
|  |
| State of the art networking medium (sent 585 GB) |

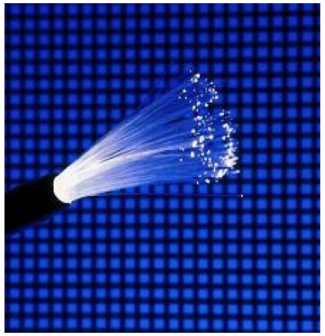
| | |
| --- | --- |
| Latency (s) | 1800 |
| BW (GB/s) | 1.13 |

3.5 in Hard drive
3 TB
0.68 kg

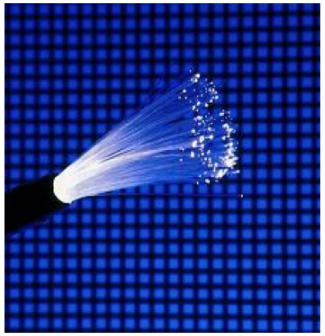
# The Internet “Land”-Speed Record

| | Fiber-optic cable | Subaru Outback |
|---|---|---|
| |  |  |
| | State of the art networking medium (sent 585 GB) | Sensible station wagon |
| Cargo | | 183 kg |
| Speed | | 119 MPH |
| Latency (s) | 1800 | |
| BW (GB/s) | 1.13 | |

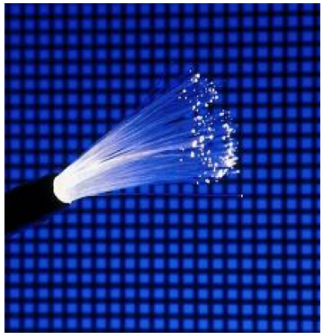
# The Internet “Land”-Speed Record

| | Fiber-optic cable | Subaru Outback |
|---|---|---|
| |  |  |
| | State of the art networking medium (sent 585 GB) | Sensible station wagon |
| Cargo | | 183 kg |
| Speed | | 119 MPH |
| Latency (s) | 1800 | 563,984 |
| BW (GB/s) | 1.13 | 0.0014 |

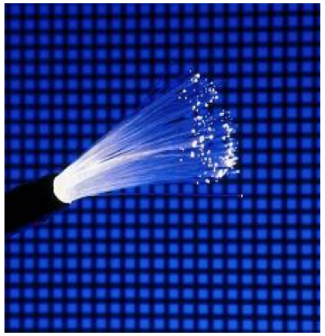
# The Internet “Land”-Speed Record

| | Fiber-optic cable | Subaru Outback | B1-B |
|---|---|---|---|
| |  |  |  |
| | State of the art networking medium (sent 585 GB) | Sensible station wagon | Supersonic bomber |
| Cargo | | 183 kg | 25,515 kg |
| Speed | | 119 MPH | 950 MPH |
| Latency (s) | 1800 | 563,984 | |
| BW (GB/s) | 1.13 | 0.0014 | |

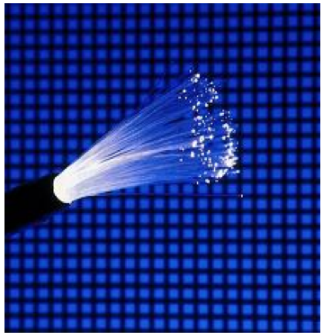
# The Internet “Land”-Speed Record

| | Fiber-optic cable | Subaru Outback | B1-B |
|---|---|---|---|
| |  |  |  |
| | State of the art networking medium (sent 585 GB) | Sensible station wagon | Supersonic bomber |
| Cargo | | 183 kg | 25,515 kg |
| Speed | | 119 MPH | 950 MPH |
| Latency (s) | 1800 | 563,984 | 70,646 |
| BW (GB/s) | 1.13 | 0.0014 | 1.6 |

# The Internet “Land”-Speed Record

| | Fiber-optic cable | Subaru Outback | B1-B | Hellespont Alhambra |
|---|---|---|---|---|
| |  |  |  |  |
| | State of the art networking medium (sent 585 GB) | Sensible station wagon | Supersonic bomber | World’s largest supertanker |
| Cargo | | 183 kg | 25,515 kg | 400,975,655 kg |
| Speed | | 119 MPH | 950 MPH | 18.9 MPH |
| Latency (s) | 1800 | 563,984 | 70,646 | |
| BW (GB/s) | 1.13 | 0.0014 | 1.6 | |

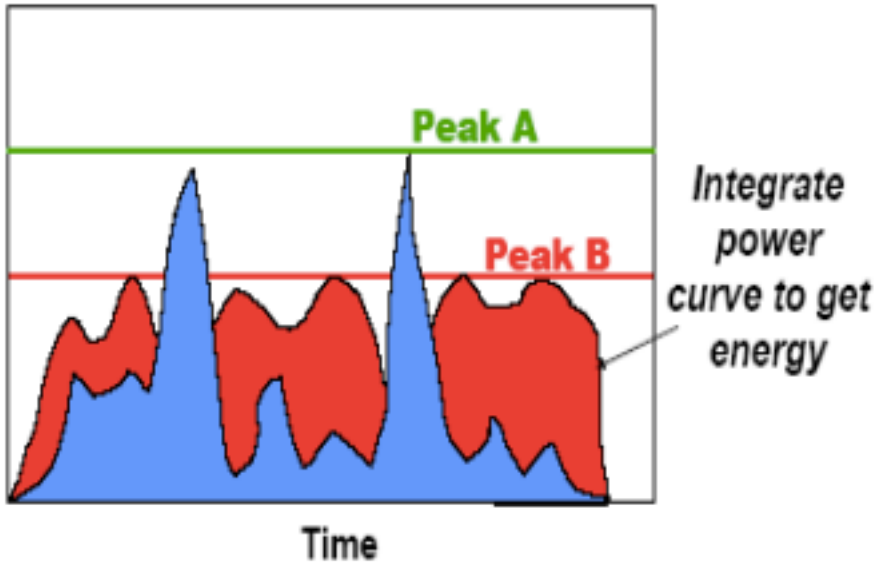
# The Internet “Land”-Speed Record

| | Fiber-optic cable | Subaru Outback | B1-B | Hellespont Alhambra |
|---|---|---|---|---|
| |  |  |  |  |
| | State of the art networking medium (sent 585 GB) | Sensible station wagon | Supersonic bomber | World’s largest supertanker |
| Cargo | | 183 kg | 25,515 kg | 400,975,655 kg |
| Speed | | 119 MPH | 950 MPH | 18.9 MPH |
| Latency (s) | 1800 | 563,984 | 70,646 | 1,587,301 |
| BW (GB/s) | 1.13 | 0.0014 | 1.6 | 1114.5 |
| Tb-m/s | 272,400 | 344,690 | 382,409,815 | 267,000,000,000 |

# Example: Power

- Energy
  - measured in Joules
- Power
  - rate of energy consumption

- Example
  - System A
    - higher peak power
  - System B
    - lower peak power



# Example: $$ (price)

- The importance is self-explanatory.

# Derived metrics

- Often we care about multiple metrics at once.
- Examples (Bigger is better)
  - Bandwidth per dollar (e.g., in networking (GB/s)/$)
  - BW/Watt (e.g., in memory systems (GB/s)/W)
  - Work/Joule (e.g., instructions/joule)
  - In general: Multiply by big-is-better metrics, divide by smaller-is-better
- Examples (Smaller is better)
  - Cycles/Instruction (i.e., Time per work)
  - Latency * Energy -- “Energy Delay Product”
  - In general: Multiply by smaller-is-better metrics, divide by bigger-is-better

# Example: Energy-Delay

- Mobile systems must balance latency (delay) and battery (energy) usage for computation.
- The energy-delay product (EDP) is a "smaller is better" metric
  - Base units: Delay in seconds; Energy in Joules;
  - EDP units: Joules*seconds

Some use energy * delay$^2$

# What's the Right Metric?

- There is not universally correct metric
- You can use *any* metric you like to evaluate computer systems
  - Latency for gcc
  - Frames per second on Crysis
  - (Database transactions/second)/$

- The right metric depends on the situation.
  - What does the computer need to accomplish?
  - What constraints is it under?
- Usually some, relatively simple, combination of metrics on the "basic metrics" slide.
- We will mostly focus on performance (latency and/or bandwidth)

Now we have the metrics, how do we compare different architectures?

# Benchmarks

# Benchmarks: Make comparable measurements

- A benchmark suite is a set of programs that are representative of a set of problems
  - Server computing (SPECINT)
  - Scientific computing (SPECFP)
- There is no best benchmark suite
  - Reason? There are so many different problems
- Real software is none of those things

# Classes of benchmarks

- Microbenchmarks measure one feature of system
  - e.g. memory accesses or communication speed
- Kernels – most compute-intensive part of applications
  - Amdahl's Law tells us that this is fine for some applications.
  - e.g. Linpack and NAS kernel benchmarks
- Full application:
  - SpecInt / SpecFP (for servers)
  - Other suites for databases, web servers, graphics,…

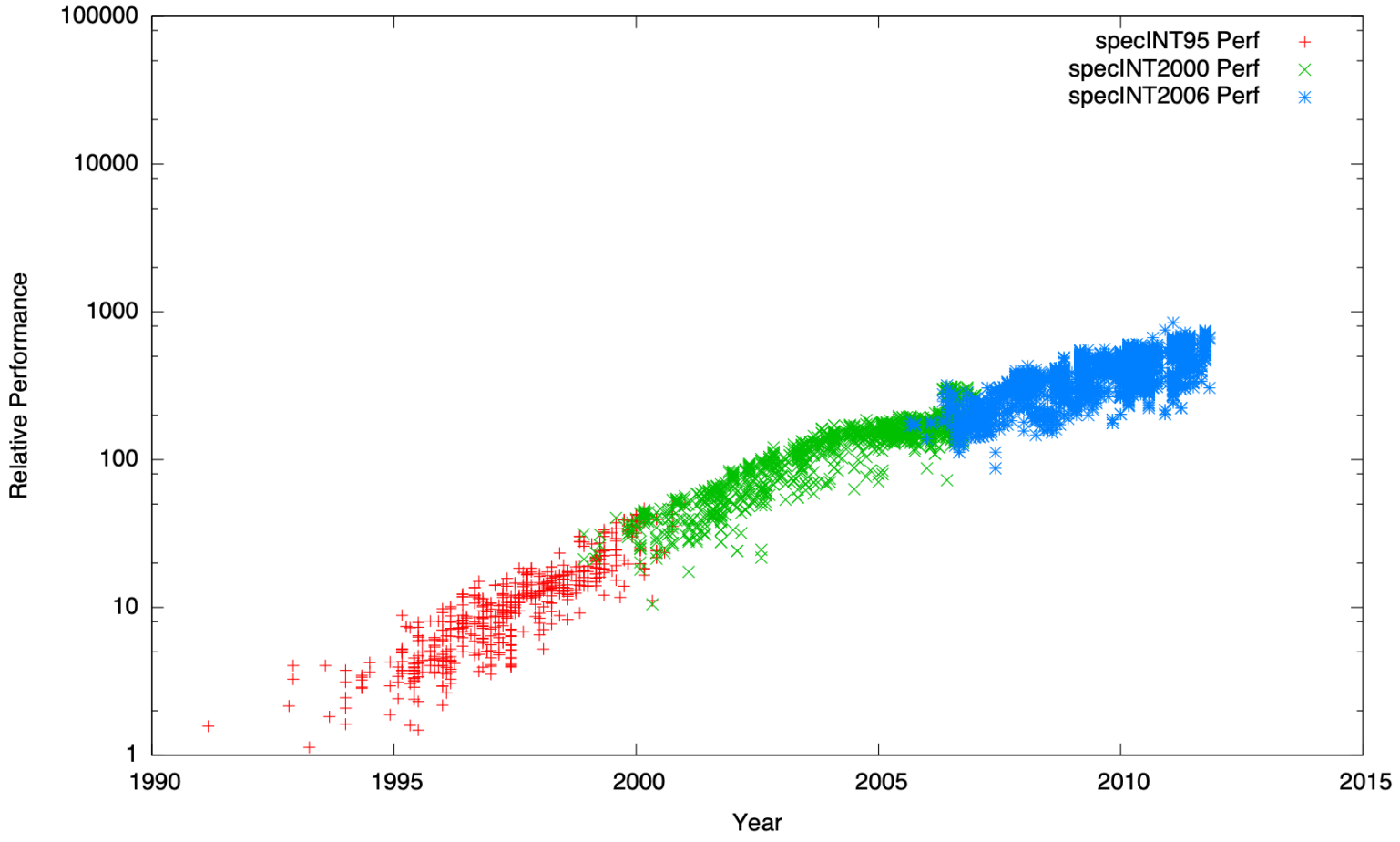
# SPECINT 2006

- In what ways are these not representative?

| Application | Language | Description |
|---|---|---|
| 400.perlbench | C | PERL Programming Language |
| 401.bzip2 | C | Compression |
| 403.gcc | C | C Compiler |
| 429.mcf | C | Combinatorial Optimization |
| 445.gobmk | C | AI: go |
| 456.hmmer | C | Search Gene Sequence |
| 458.sjeng | C | AI: chess |
| 462.libquantum | C | Quantum Computing |
| 464.h264ref | C | Video Compression |
| 471.omnetpp | C++ | Discrete Event Simulation |
| 473.astar | C++ | Path-finding Algorithms |
| 483.xalancbmk | C++ | XML Processing |

# SPECINT 2006

- Despite all that, benchmarks are quite useful.
  - e.g., they allow long-term performance comparisons



# Machine Learning Performance Benchmarking

ML Commons

Menu

September 08, 2022 - Inference: Datacenter

## v2.1 Results

Other Rounds

Inference Datacenter v2.1

Closed | Closed - Power | Closed - Network | Open | Open - Power

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2.1-0049 | HPE | HPE Apollo 6500 Gen10+ (675d - 8x A100-SXM-80GB-MIG-1x1g.10gb), TensorRT | 1 | AMD EPYC 7763 | 2 | NVIDIA A100-SXM-80GB (1x1g.10gb MIG) | 8 | TensorRT 8.4.0, CUDA 11.6 | | 3,526.96 | 5,559.41 |
| 2.1-0050 | HPE | HPE Apollo 6500 Gen10+ (675d - 8x A100-SXM-80GB-MIG-1x1g.10gb), TensorRT, Triton | 1 | AMD EPYC 7763 | 2 | NVIDIA A100-SXM-80GB (1x1g.10gb MIG) | 8 | TensorRT 8.4.0, CUDA 11.6 | | 3,437.40 | 5,357.92 |
| 2.1-0051 | HPE | HPE Apollo 6500 Gen10+ (645d - 4x A100 SXM 40GB) | 1 | AMD EPYC 7702 | 1 | NVIDIA A100-SXM-40GB | 4 | TensorRT 8.4.0, CUDA 11.6 | | 99,012.10 | 158,608.00 |
| 2.1-0052 | HPE | HPE Apollo 6500 Gen10+ (675d - 8x A100 SXM 40GB) | 1 | AMD EPYC 7763 | 2 | NVIDIA A100-SXM-40GB | 8 | TensorRT 8.4.0, CUDA 11.6 | | 280,066.00 | 311,662.00 |
| 2.1-0053 | HPE | HPE Apollo 6500 Gen10+ (675d - 8x A100 SXM 80GB) | 1 | AMD EPYC 7763 | 2 | NVIDIA A100-SXM-80GB | 8 | TensorRT 8.4.0, CUDA 11.6 | | 300,064.00 | 339,166.00 |
| 2.1-0055 | Inspur | NF5468M6(4x BR104-300W PCIe, sulnfer) | 1 | Intel(R) Xeon(R) Gold 6354 CPU | 2 | BR104-300W PCIe | 4 | sulnfer | | 150,027.00 | 212,391.00 |
| 2.1-0056 | Inspur | NF5488A5 (8x A100-SXM-80GB, TensorRT) | 1 | AMD EPYC 7713 | 2 | NVIDIA A100-SXM-80GB | 8 | TensorRT 8.4.2, CUDA 11.7 | | 290,066.00 | 346,954.00 |
| 2.1-0057 | Inspur | NF5688M6 (8x A100-SXM-80GB, TensorRT) | 1 | Intel(R) Xeon(R) Platinum 8358 | 2 | NVIDIA A100-SXM-80GB | 8 | TensorRT 8.4.2, CUDA 11.7 | | 313,069.00 | 347,202.00 |
| 2.1-0079 | Lenovo | Lenovo SR670v2 (8x A100-PCIe-80GB, TensorRT) | 1 | Intel(R) Xeon(R) Platinum 8360Y CPU @ 2.40GHz | 2 | NVIDIA A100-PCIe-80GB | 8 | TensorRT 8.4.0, CUDA 11.6 | | 262,068.00 | 318,162.00 |
| 2.1-0080 | Lenovo | Lenovo SR670v2 (4x A100-SXM-80GB_aarch64, TensorRT) | 1 | Intel(R) Xeon(R) Platinum 8360Y CPU @ 2.40GHz | 2 | NVIDIA A100-SXM-80GB | 4 | TensorRT 8.4.0, CUDA 11.6 | | 150,027.00 | 174,180.00 |
| 2.1-0082 | NVIDIA | NVIDIA DGX A100 (1x A100-SXM-80GB-MIG-1x1g.10gb, TensorRT) | 1 | AMD EPYC 7742 | 2 | NVIDIA A100-SXM-80GB (1x1g.10gb MIG) | 1 | TensorRT 8.4.2, CUDA 11.6 | | 3,526.96 | 5,589.01 |
| 2.1-0083 | NVIDIA | NVIDIA DGX A100 (1x A100-SXM-80GB-MIG-1x1g.10gb, TensorRT, Triton) | 1 | AMD EPYC 7742 | 2 | NVIDIA A100-SXM-80GB (1x1g.10gb MIG) | 1 | TensorRT 8.4.2, CUDA 11.6 | | 3,437.40 | 5,387.57 |
| 2.1-0084 | NVIDIA | Gigabyte G482-Z54 (8x A100-PCIe-80GB, TensorRT) | 1 | AMD EPYC 7742 64-Core Processor | 2 | NVIDIA A100-PCIe-80GB | 8 | TensorRT 8.4.2, CUDA 11.6 | | 236,057.00 | 316,342.00 |
| 2.1-0085 | NVIDIA | Gigabyte G482-Z54 (8x A100-PCIe-80GB, MaxQ, TensorRT) | 1 | AMD EPYC 7742 64-Core Processor | 2 | NVIDIA A100-PCIe-80GB | 8 | TensorRT 8.4.2, CUDA 11.6 | | 185,047.00 | 252,721.00 |
| 2.1-0086 | NVIDIA | Gigabyte G482-Z54 (8x A100-PCIe-80GB, TensorRT, Triton) | 1 | AMD EPYC 7742 64-Core Processor | 2 | NVIDIA A100-PCIe-80GB | 8 | TensorRT 8.4.2, CUDA 11.6 | | 230,054.00 | 316,169.00 |
| 2.1-0087 | NVIDIA | NVIDIA DGX A100 (1x A100-SXM-80GB, TensorRT) | 1 | AMD EPYC 7742 | 2 | NVIDIA A100-SXM-80GB | 1 | TensorRT 8.4.2, CUDA 11.6 | | | |
| 2.1-0088 | NVIDIA | NVIDIA DGX A100 (8x A100-SXM-80GB, TensorRT) | 1 | AMD EPYC 7742 | 2 | NVIDIA A100-SXM-80GB | 8 | TensorRT 8.4.2, CUDA 11.6 | | 300,064.00 | 335,144.00 |
| 2.1-0089 | NVIDIA | NVIDIA DGX A100 (8x A100-SXM-80GB, MaxQ, TensorRT) | 1 | AMD EPYC 7742 | 2 | NVIDIA A100-SXM-80GB | 8 | TensorRT 8.4.2, CUDA 11.6 | | 229,055.00 | 288,733.00 |
| 2.1-0090 | NVIDIA | NVIDIA DGX A100 (8x A100-SXM-80GB, TensorRT, Triton) | 1 | AMD EPYC 7742 | 2 | NVIDIA A100-SXM-80GB | 8 | TensorRT 8.4.2, CUDA 11.6 | | 200,052.00 | 326,137.00 |
| 2.1-0097 | Nettrix | X660G45L (8x A100-SXM4-80GB, TensorRT) | 1 | Intel(R) Xeon(R) Platinum 8368Q | 2 | NVIDIA A100-SXM4-80GB CTS | 8 | TensorRT 8.4.2, CUDA 11.6 | | 307,265.00 | 347,298.00 |
| 2.1-0099 | Qualcomm | Gigabyte G292-Z43 (16x QAIC100 Pro) | 1 | AMD EPYC 7713 64-Core Processor | 2 | QUALCOMM Cloud AI 100 PCIe/HHHL Pro | 16 | Qualcomm Cloud AI SDK v1.7.1 | | | |
| 2.1-0100 | Qualcomm | Gigabyte G292-Z43 (18x QAIC100 Pro) | 1 | AMD EPYC 7713 64-Core Processor | 2 | QUALCOMM Cloud AI 100 PCIe/HHHL Pro | 18 | Qualcomm Cloud AI SDK v1.7.1 | | 330,079.00 | 428,786.00 |
| 2.1-0101 | Qualcomm | Gigabyte R282-Z93 (8x QAIC100 Pro) | 1 | AMD EPYC 7282 16-Core Processor | 2 | QUALCOMM Cloud AI 100 PCIe/HHHL Pro | 8 | Qualcomm Cloud AI SDK v1.7.1 | | 173,044.00 | 182,343.00 |
| 2.1-0109 | SAPEON | SUPERMICRO SYS-620C-TN12R X220-compact | 1 | Intel(R) Xeon(R) Gold 6330 CPU | 2 | SAPEON X220-compact | 1 | SAPEON SDK v0.52 | | 6,145.15 | 6,769.07 |
| 2.1-0110 | SAPEON | SUPERMICRO SYS-220GP-TNR X220-enterprise | 1 | Intel(R) Xeon(R) Gold 6330 CPU | 2 | SAPEON X220-enterprise | 1 | SAPEON SDK v0.52 | | 12,036.30 | 13,100.90 |
| 2.1-0111 | Supermicro | AS-4124GS-TNR (8xA100-PCIe-80GB) | 1 | AMD EPYC 7713 64-Core Processor | 2 | NVIDIA A100-PCIe-80GB | 8 | CUDA 11.6 | | 262,068.00 | 280,063.00 |

# The CPU Performance Equation

# The Performance Equation (PE)

- We would like to model how architecture impacts performance (latency)
- This means we need to quantify performance in terms of architectural parameters.
  - Instruction Count -- The number of instructions the CPU executes
  - Cycles per instructions -- The ratio of cycles for execution to the number of instructions executed.
  - Cycle time -- The length of a clock cycle in seconds
- The first fundamental theorem of computer architecture:

Latency = Instruction Count * Cycles/Instruction * Seconds/Cycle

L = IC * CPI * CT

# The PE as Mathematical Model

Latency = Instructions * Cycles/Instruction * Seconds/Cycle

- Good models give insights into the system they model
  - Latency changes linearly with IC
  - Latency changes linearly with CPI
  - Latency changes linearly with CT
- It suggests several ways to improve performance
  - Reduce IC
  - Reduce CPI
  - Reduce CT
- It also allows us to evaluate potential trade-offs
  - Reducing CT by 50% and increasing CPI by 2 give us a net win

# Reducing Cycle Time

- Cycle time is a function of the processor's design
  - If the design does less work during a clock cycle, it's cycle time will be shorter.
  - More on this later, when we discuss pipelining.
- Cycle time is a function of process technology.
  - If we scale a fixed design to a more advanced process technology, it's clock speed will go up.
  - However, clock rates aren't increasing much, due to power problems.
- Cycle time is a function of manufacturing variation
  - Manufacturers "bin" individual CPUs by how fast they can run.
  - The more you pay, the faster your chip will run.

# The Clock Speed Corollary

Latency = Instructions * Cycles/Instruction * Seconds/Cycle

- We use clock speed more than second/cycle
- Clock speed is measured in Hz (e.g., MHz, GHz, etc.)
  - x Hz => 1/x seconds per cycle
  - 2.5GHz => $1/2.5 \times 10^9$ seconds (0.4ns) per cycle

Latency = (Instructions * Cycle/Insts)/(Clock speed in Hz)

# A Note About Instruction Count

- The instruction count in the performance equation is the “dynamic” instruction count
- “Dynamic”
  - Having to do with the execution of the program or counted at run time
  - ex: When I ran that program it executed 1 million dynamic instructions.
- “Static”
  - Fixed at compile time or referring to the program as it was compiled
  - e.g.: The compiled version of that function contains 10 static instructions.

# Reducing Instruction Count (IC)

- There are many ways to implement a particular computation
  - Algorithmic improvements (e.g., quicksort vs. bubble sort)
  - Compiler optimizations (e.g., pass -O4 to gcc)
- If one version requires executing fewer dynamic instructions, the PE predicts it will be faster
  - Assuming that the CPI and clock speed remain the same
  - A x% reduction in IC should give a speedup of 1/(1-0.01*x) times
  - e.g., 20% reduction in IC => 1/(1-0.2) = 1.25x speedup

# Other Impacts on Instruction Count

- Different programs do different amounts of work
  - e.g., Playing a DVD vs writing a word document
- The same program may do different amounts of work depending on its input
  - e.g., Compiling a 1000-line program vs compiling a 100-line program
- The same program may require a different number of instructions on different ISAs

- To make a meaningful comparison between two computer systems, they must be doing the same work.
  - They may execute a different number of instructions (e.g., because they use different ISAs or a different compilers)
  - But the task they accomplish should be exactly the same.

# Cycles Per Instruction

- CPI is the most complex term in the PE, since many aspects of processor design impact it
  - The compiler
  - The program's inputs
  - The processor's design (more on this later)
  - The memory system (more on this later)
- It is not the cycles required to execute one instruction
- It is the ratio of the cycles required to execute a program and the IC for that program. It is an average.
- I find 1/CPI (Instructions Per Cycle; IPC) to be more intuitive, because it emphasizes that it is an average.

# Wrap-up Exercise

Compute the CPI for the following program and machine:

- Program: 10% floating-point operations, 20% memory access instructions, 40% branch/jump instructions, and the rest are ALU operations.
- Machine: cycles required for each of the following operations:
  - floating-point operations: 11 cycles
  - memory access instructions: 50 cycles*
  - branch/jump instructions: 5 cycles*
  - ALU operations: 2 cycles

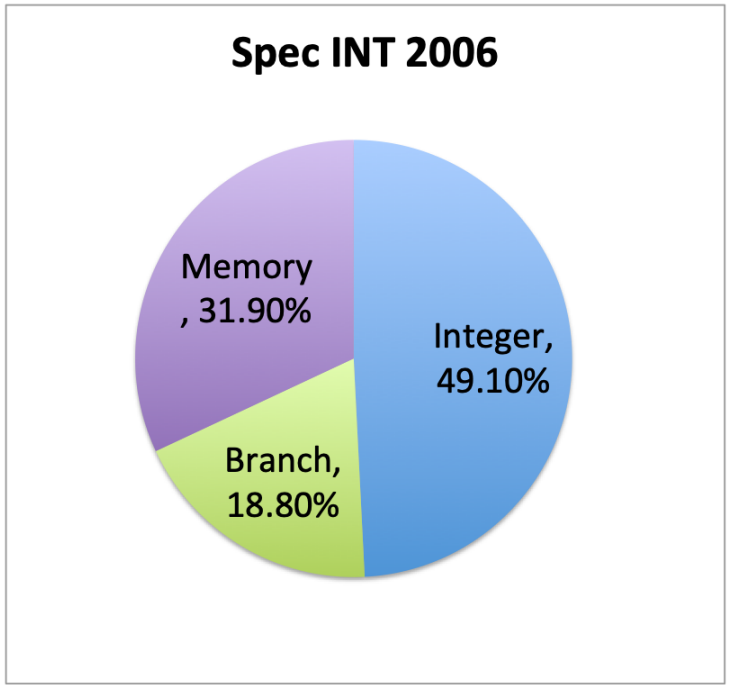
# Wrap-up Exercise

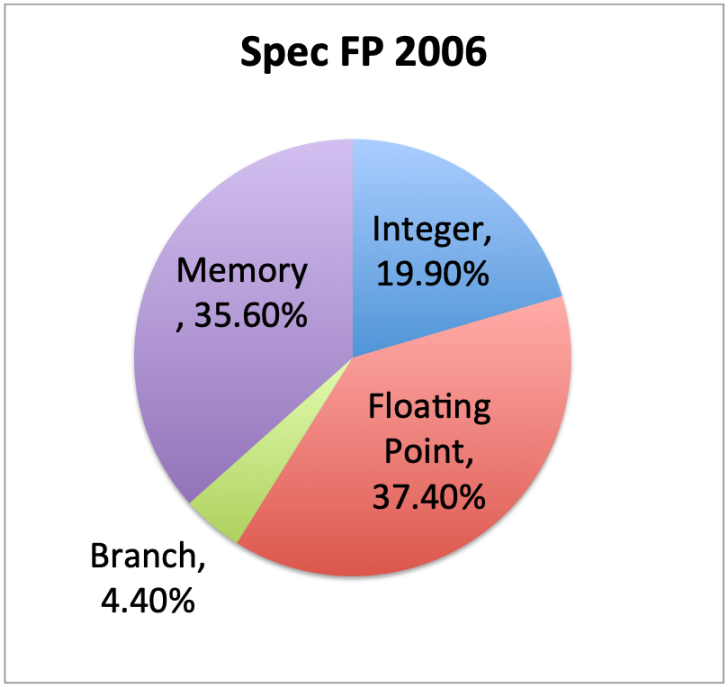
1. Compute the CPI for the following program and machine:
   - Program: 10% floating-point operations, 20% memory access instructions, 40% branch/jump instructions, and the rest are ALU operations.
   - Machine: cycles required for each of the following operations:
     - floating-point operations: 11 cycles
     - memory access instructions: 50 cycles*
     - branch/jump instructions: 5 cycles*
     - ALU operations: 2 cycles

$$\begin{aligned} \text{CPI} &= 0.10 \times 11 + 0.20 \times 50 + 0.40 \times 5 + (1 - 0.10 - 0.20 - 0.40) \times 2 \\ &= \boxed{13.7} \end{aligned}$$

# Instruction Mix and CPI

- Different programs need different kinds of instructions
  - e.g., “Integer apps” don’t do much floating point math.
- The compiler also has some flexibility in which instructions it uses.
- As a result the combination and ratio of instruction types that programs execute (their *instruction mix*) varies.




Spec INT and Spec FP are popular benchmark suites

# Instruction Mix and CPI

- Instruction selections (and, therefore, instruction selection) impacts CPI because some instructions require extra cycles to execute
- All theses values depend on the particular implementation, not the ISA.

| Instruction Type | Cycles |
|---|---|
| Integer +, -, \|, &, branches | 1 |
| Integer multiply | 3-5 |
| integer divide | 11-100 |
| Floating point +, -, *, etc. | 3-5 |
| Floating point /, sqrt | 7-27 |
| Loads and Stores | 1-100s |

These values are for Intel's Nehalem processor

# Program Inputs and CPI

- Different inputs make programs behave differently
  - They execute different functions
  - They branches will go in different directions
  - These all affect the instruction mix (and instruction count) of the program.

# Comparing Similar Systems

Latency = Instructions * Cycles/Instruction * Seconds/Cycle

- Often, we will comparing systems that are partly the same
  - e.g., Two CPUs running the same program
  - e.g., One CPU running two programs
- In these cases, many terms of the equation are not relevant
  - e.g., If the CPU doesn't change, neither does CT, so performance can measured in cycles:

    Instructions * Cycles/Instruction == Cycles.
  - e.g., If the workload is fixed, IC doesn't change, so performance can be measured in Instructions/Second:

    1/(Cycles/Instruction * Seconds/Cycle)
  - e.g., If the workload *and* clock rate are fixed, the latency is equivalent to CPI (smaller-is-better). Alternately, performance is equivalent to Instructions per Cycle (IPC; bigger-is-better).

You can only ignore terms in the PE, if they are *identical* across the two systems

# Treating PE Terms Differently

- The PE also allows us to apply “rules of thumb” and/or make projections.
- Example: “CPI is modern processors is between 1 and 2”
  - $L = IC * CPI_{guess} * CT$
  - In this case, IC corresponds to a particular application, but $CPI_{guess}$ is an estimate.
- Example: This new processor will reduce CPI by 50% and reduce CT by 50%.
  - $L = IC * 0.5CPI * CT/2$
  - Now CPI and CT are both estimates, and the resulting L is also an estimate. IC may not be an estimate.

# Abusing the PE

- Be ware of Guaranteed Not To Exceed (GTNE) metrics
- Example: “Processor X has a speed of 10 GOPS (giga insts/sec)”
  - This is equivalent to saying that the average instruction latency is 0.1ns.
  - No workload is given!
  - Does this means that L = IC * 0.1ns? Probably not!
- The above claim (probably) means that the processor is capable of 10 GOPS under perfect conditions
  - The vendor promises it will never go faster.
  - That’s very different that saying how fast it will go in practice.
- It may also mean they get 10 GOPS on an industry standard benchmark
  - All the hazards of benchmarks apply.
  - Does your workload behave the same as the industry standard benchmark?

# The top500 list

- What's the fastest computer in the world?
  - www.top500.org will tell you
  - A list of the fastest 500 computers
- They report floating point operations per second (FLOPS)
  - They use the linpack benchmark suite (dense matrix computation)
- The fastest machine is now hosted in the US
- Is it meaningful or fair?
  - a new list www.graph500.org