Performance

Hung-Wei Tseng

Outline

- What is performance?
- What is the performance equation?
- What affects performance
- Amdahl's Law

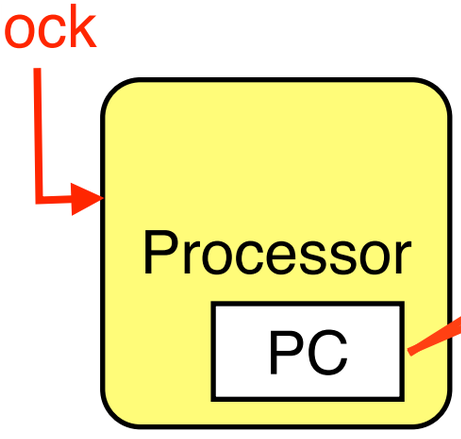
Performance!

What do you want in a computer?

- Frame rate
- Responsiveness
- Real-time
- Throughput
- Cost
- Volume
- Weight
- Battery life
- Low power/low temperature
- Reliability
- Latency/Execution time

Execution Time

- The simplest kind of performance
- Shorter execution time means better performance
- Usually measured in seconds



How many of these?

Instruction Count!

How long is it take to execution each of these?

Cycles per instruction * cycle time

instruction memory

```
120007a30:   0f00bb27   ldah   gp,15(t12)
120007a34:   509cbd23   lda    gp,-25520(gp)
120007a38:   00005d24   ldah   t1,0(gp)
120007a3c:   0000bd24   ldah   t4,0(gp)
120007a40:   2ca422a0   ldl    t0,-23508(t1)
120007a44:   130020e4   beq    t0,120007a94
120007a48:   00003d24   ldah   t0,0(gp)
120007a4c:   2ca4e2b3   stl    zero,-23508(t1)
120007a50:   0004ff47   clr    v0
120007a54:   28a4e5b3   stl    zero,-23512(t4)
120007a58:   20a421a4   ldq    t0,-23520(t0)
120007a5c:   0e0020e4   beq    t0,120007a98
120007a60:   0204e147   mov    t0,t1
120007a64:   0304ff47   clr    t2
120007a68:   0500e0c3   br     120007a80
```

Performance equation!

Performance Equation

$$\text{Execution Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

How many instruction executed?

How long is it to execute each instruction

- ET = IC * CPI * CT
- IC (Instruction Count)
- CPI (Cycles Per Instruction)
- CT (Seconds Per Cycle)
  - 1 Hz = 1 second per cycle; 1 GHz = 1 ns per cycle

Speedup

- Compare the relative performance of the baseline system and the improved system
- Definition

$$\text{Speedup} = \frac{\text{Execution time}_{\text{baseline}}}{\text{Execution time}_{\text{improved system}}}$$

What affects performance

Demo: programmer & performance

- Row-major, column major (we have seen this in the first class)
- Sorted array

Demo: compiler & performance

- Compiler optimization can help reducing the instruction count
- Compiler optimization can improve CPI
  - Wise selection of instruction combinations
  - Use registers to eliminate loads and stores

Recap: Performance Equation

$$\text{Execution Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

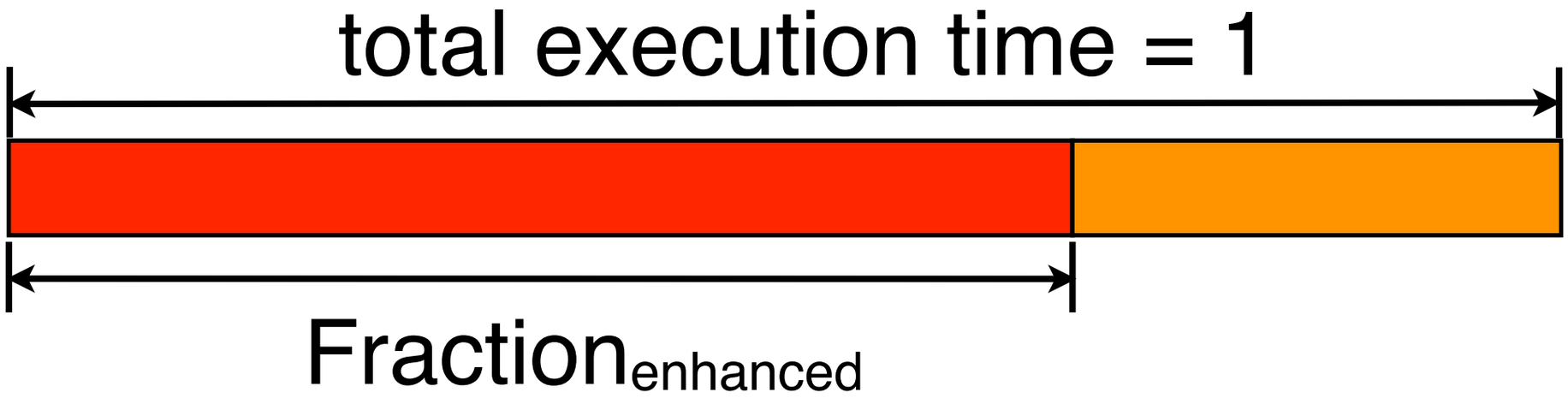
- ET = IC * CPI * Cycle Time
- IC (Instruction Count)
  - ISA, Compiler, algorithm, programming language
- CPI (Cycles Per Instruction)
  - Machine Implementation, microarchitecture, compiler, application, algorithm, programming language
- Cycle Time (Seconds Per Cycle)
  - Process Technology, microarchitecture

Amdahl's Law

Amdahl's Law

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

- Amdahl's Law can be used anywhere!
- The Fraction means the fraction of "time"



Amdahl's Corollary #1

- Maximum possible speedup S_{max}, if we are targeting *x* of the program.

$$S = \text{infinity}$$

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

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

Maximum of speedup

- Call of Duty Black Ops II loads a zombie map for 10 minutes on my current machine, and spends 20% of this time in integer instructions
- How much faster must you make the integer unit to make the map loading 5 minutes faster?

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

$$1.25 = \frac{1}{(1-20\%)}$$

2x is not possible.

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 can change based on inputs, compiler options, optimizations you've applied, etc.

If we repeatedly optimizing our design based on Amdahl's law...

- With optimization, the common becomes uncommon.
- An uncommon case will (hopefully) become the new common case.
- Now you have a new target for optimization.

Amdahl's Corollary #3

- Assume that we have an application, in which x of the execution time in this application can be fully parallelized with S processors. What's the speedup if we use a S-core processor instead of a single-core processor?

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

Amdahl's Corollary #4

- Amdahl's law for latency (L)
- By definition
  - Speedup = oldLatency/newLatency
  - newLatency = oldLatency * 1/Speedup
- By Amdahl's law:
  - newLatency = old Latency * (x/S + (1-x))
  - newLatency = x*oldLatency/S + oldLatency*(1-x)
- Amdahl's law for latency
  - newLatency = x*oldLatency/S + oldLatency*(1-x)

Amdahl's Non-Corollary

- Amdahl's law does not bound slowdown
  - newLatency = x*oldLatency/S + oldLatency*(1-x)
  - newLatency is linear in 1/S
- Example: x = 0.01 of execution, oldLat = 1
  - S = 0.001;
    Newlat = 1000*Oldlat *0.01 + Oldlat *(0.99) = ~ 10*Oldlat
  - S = 0.00001;
    Newlat = 100000*Oldlat *0.01 + Oldlat *(0.99) = ~ 1000*Oldlat
- Things can only get so fast, but they can get arbitrarily slow.
  - Do not hurt the non-common case too much!

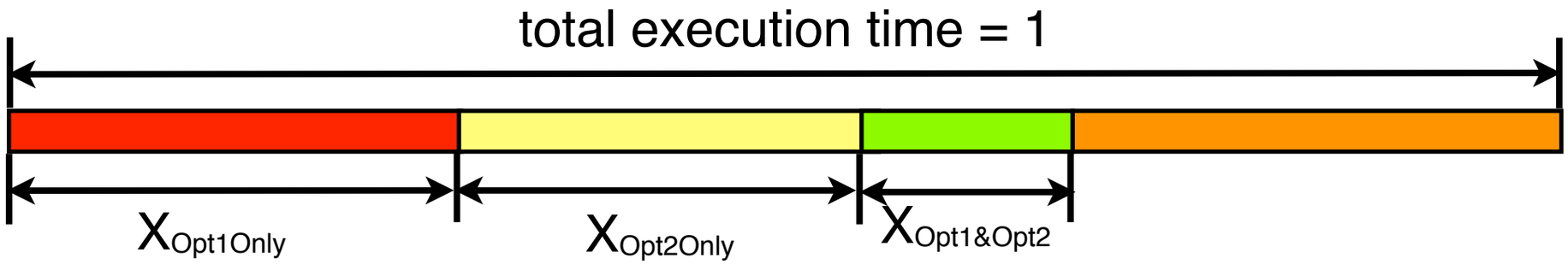
Multiple optimizations

- We can apply Amdahl's law for multiple optimizations
- These optimizations must be dis-joint!
  - If optimization #1 and optimization #2 are dis-joint:

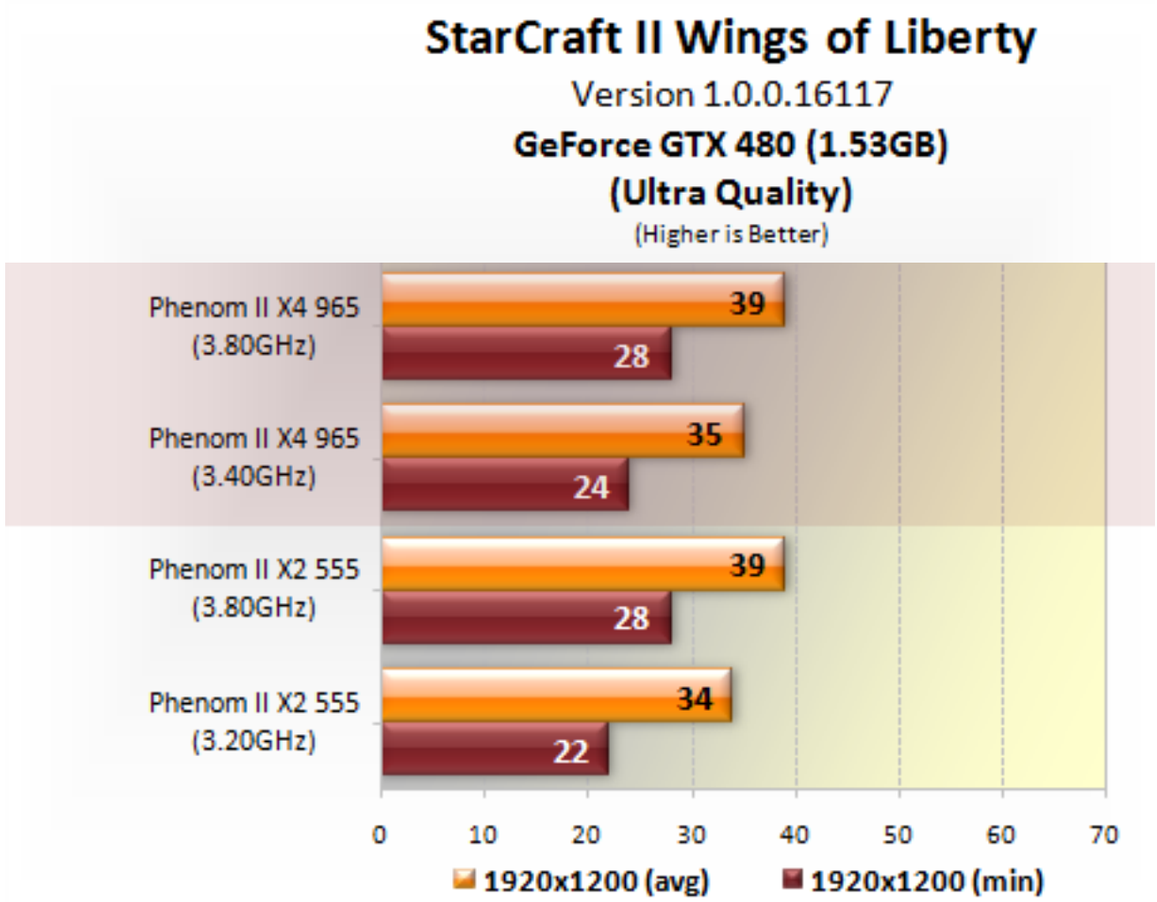
$$\text{Speedup} = \frac{1}{(1 - X_{Opt1} - X_{Opt2}) + \frac{X_{Opt1}}{S_{Opt1}} + \frac{X_{Opt2}}{S_{Opt2}}}$$

  - If optimization #1 and optimization #2 are not dis-joint:

$$S = \frac{1}{(1 - X_{Opt1Only} - X_{Opt2Only} - X_{Opt1\&Opt2}) + \frac{X_{Opt1}}{S_{Opt1Only}} + \frac{X_{Opt2}}{S_{Opt2Only}} + \frac{X_{Opt1\&Opt2}}{S_{Opt1\&Opt2}}}$$

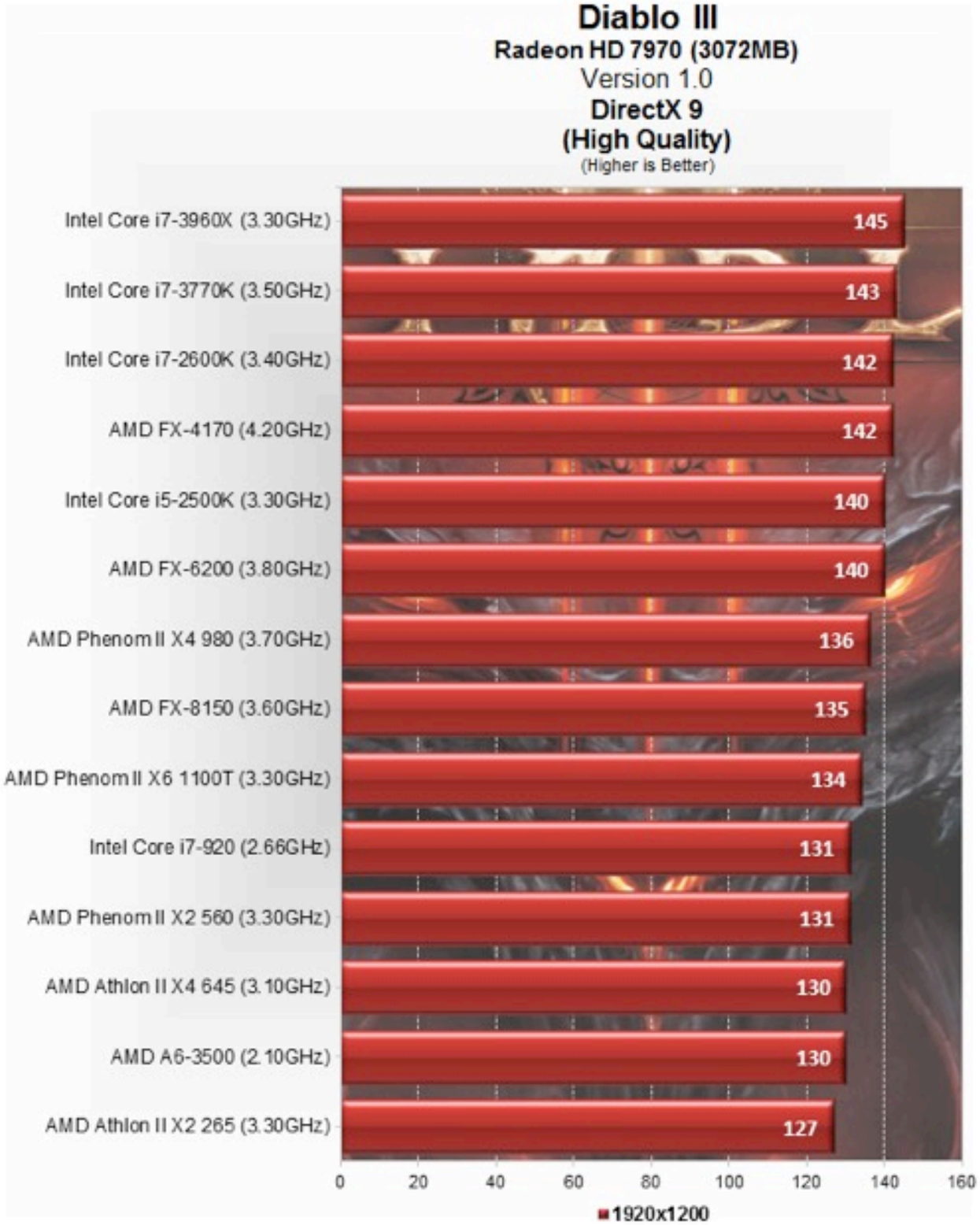


Case study: StarCraft II



- Corollary #3
- Adding cores does not always work
  - The application does not scale with the number of cores very well.
- Still help improving overall system performance if you have multiple tasks in the background (like web browsers, IMs...)

Case study: Diablo III



- Corollary #2
- The CPU is not the main performance bottleneck
- You might consider
  - GPU
  - network
  - storage (loading maps)

Other important metrics

Bandwidth

- The amount of work (or data) during a period of time
  - Network/Disks: MB/sec, GB/sec, Gbps, Mbps
  - Game/Video: Frames per second
- Also called “throughput”
- “Work done” / “execution time”

Response time and BW trade-off

- Increase bandwidth can hurt the execution time of a single task
- If you want to transfer 2 Peta-Byte of data from UCLA
  - 125 miles (201.25 km) from UCSD
  - You can use an Internet 2 network with 100Gbps speed
  - 2 Peta-byte over 167772 seconds = 1.94 Days
  - 22.5TB in 30 minutes
  - Bandwidth: 100 Gbps

Or …

- Use a Toyota Prius!
  - 125 miles (201.25 km) from UCSD
  - 75 MPH on highway!
  - 50 MPG
  - Max load: 374 kg = 2,770 hard drives (1TB per drive)
  - 4 hours round-trip
  - Get nothing in first 30 minutes...
  - Bandwidth: 145 GB/sec

- Internet 2 network with 100Gbps speed
  - 2 Peta-byte over 167772 seconds = 1.94 Days
  - 22.5TB in 30 minutes
  - Bandwidth: 100 Gbps = 12.5 GB/sec

Reliability

- Mean time to failure (MTTF)
  - Average time before a system stops working
  - Very complicated to calculate for complex systems
- Hardware can fail because of
  - Electromigration
  - Temperature
  - High-energy particle strikes

Energy

- Energy == joules
  - You buy electricity in joules.
  - Battery capacity is in joules
  - To minimizes operating costs, minimize energy
  - You can also think of this as the amount of work that computer must actually do
- Power == Watt == joules/sec
  - Power is how fast your machine uses joules
  - It determines battery life
  - It is also determines how much cooling you need. Big systems need 0.3-1 Watt of cooling for every watt of compute.

You may also heard...

GFLOPS (Giga FLoating-point Operations Per Second)

	GFLOPS	clock rate
XBOX One	1228.8	1.6 GHz
PS4	2900.0	1.6 GHz
Core i7 EE 3970X + AMD Radeon 6990	5099.0	3.5 GHz

Is GFLOPS (Giga FLoating-point Operations Per Second) a good metric?

$$\text{GFLOPS} = \frac{\text{\# of floating point instructions} / 10^9}{\text{Execution Time}}$$

$$= \frac{\text{IC} \times \text{\% of floating point instructions}}{\text{IC} \times \text{CPI} \times \text{CycleTime} \times 10^9} = \frac{\text{Clock Rate} \times \text{\% FP ins.}}{\text{CPI} \times 10^9}$$

- Cannot compare different ISA/compiler
  - What if the compiler can generate code with fewer instructions?
  - What if new architecture has more IC but also lower CPI?
- Does not make sense if the application is not floating point intensive

Q & A