Performance (II)

Hung-Wei Tseng
Recap: Performance Equation

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, programmer
- CPI (Cycles Per Instruction)
  - Machine implementation, microarchitecture, compiler, application, algorithm, programming language, programmer
- Cycle Time (Seconds Per Cycle)
  - Process technology, microarchitecture, ISA, programmer
Announcement

• Reading quizzes due Thursday
• Reading quizzes due next Tuesday
• Hung-Wei’s office hour tomorrow 2p-3p
  • No office hour this Friday
• Runping Wang’s tutor hours this Friday 3p-5p
• CSE141/L in summer session I
Today’s CSE141

• Amdahl’s Law
• Other performance metrics
• Power and energy
Amdahl’s Law
Amdahl’s Law

Speedup = \frac{1}{\left(\frac{x}{S}\right) + (1-x)}

- x: the fraction of “execution time” that we can speed up in the target application
- S: by how many times we can speed up x

\text{total execution time} = 1

\text{total execution time} = \left(\frac{x}{S}\right) + (1-x)
Example of Amdahl’s Law

• 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 1 minute faster?

A. 1.11  
B. 1.25  
C. 1.31  
D. 2.00  
E. 2.51

\[ \text{Speedup} = \frac{1}{\frac{x}{S} + (1-x)} \]

\[ \text{Speedup} = \frac{10}{10-1} = 1.111 \]

\[ 1.111 = \frac{1}{\frac{20\%}{S} + (1-20\%)} \]

\[ S = 2 \]
Example of Amdahl’s Law

- 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?

A. 0.66x  
B. 16.6x  
C. 66.6x  
D. 100x  
E. None of the above

Speedup = \frac{10}{10-5} = 2

\text{Speedup} = \frac{1}{\frac{x}{S} + (1-x)}

2 = \frac{1}{\frac{20\%}{S} + (1-20\%)}

S = -0.66

Is this possible?
Amdahl’s Corollary #1

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

\[ S = \text{infinity} \]

\[ S_{\text{max}} = \frac{1}{(1-x)} \]

\[ S_{\text{max}} = \frac{1}{(\frac{x}{\inf}+(1-x))} \]
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?

\[
S_{\text{max}} = \frac{1}{1-x} \\
1.25 = \frac{1}{1-0.20}
\]

2x is not possible.
What to optimize?

• Call of Duty Black Ops II loads a zombie map for 10 minutes on my current machine.
• It spends 20% of loading map time in integer instructions.
• It spends 35% of loading map time in accessing SSD.
• If I have $200 to upgrade the system, should I:
  A. Upgrading my CPU to speed up the integer instruction processing by 2x
  B. Replacing my SSD with a high-end model that reduces the access time from 20us to 12us
What to optimize?

- Call of Duty Black Ops II loads a zombie map for **10 minutes** on my current machine.
- It spends **20%** of loading map time in **integer instructions**
- It spends **35%** of loading map time in accessing SSD
- If I have $200 to upgrade the system, should I:

<table>
<thead>
<tr>
<th>Replacing CPU</th>
<th>Replacing SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup = (\frac{x}{s} + (1-x)) (\frac{1}{1-20%})</td>
<td>Speedup = (\frac{x}{s} + (1-x)) (\frac{1}{1-30%})</td>
</tr>
<tr>
<td>1.11 = (\frac{20%}{2} + (1-20%))</td>
<td>1.16 = (\frac{35%}{20/12} + (1-35%))</td>
</tr>
</tbody>
</table>
What to optimize?

• Call of Duty Black Ops II loads a zombie map for **10 minutes** on my current machine.
• It spends **20%** of loading map time in integer instructions.
• It spends **35%** of loading map time in accessing SSD.
• If I have $200 to upgrade the system, should I:
  
  A. Upgrading my CPU to speed up the integer instruction processing by 2x
  
  B. Replacing my SSD with a high-end model that reduces the access time from 20us to 12us
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.
Demo

- Quicksort takes a lot of time if we want to sort a 300M array
- GPU gives you 10x speed up!
- New bottleneck emerges!
Don’t hurt non-common part too mach

- If the program spend 90% in A, 10% in B. Assume that an optimization can accelerate A by 9x, by hurts B by 10x...
- Assume the original execution time is $T$. The new execution time

$$T_{\text{new}} = \frac{T \times 0.9}{9} + T \times 0.1 \times 10$$

$$T_{\text{new}} = 1.1T$$

Speedup = \frac{T}{1.1T} = 0.91
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?

\[ S_{par} = \frac{1}{\frac{x}{S} + (1-x)} \]
Add cores or features?

- Recent advances in process technology have quadrupled the number transistors you can fit on your die.
- Currently, your key customer can use up to 4 processors for 40% of their application.
- Which will you choose?

A. Increase the number of processors from 1 to 4

B. Use 2 cores, but add features that will allow the application to use two cores for 80% of execution.

\[ S_{\text{quad-core}} = \frac{1}{\frac{x}{S} + (1-x)} \]

\[ 1.43 = \frac{1}{\frac{40\%}{4} + (1-40\%)} \]

\[ S_{\text{dual-core}} = \frac{1}{\frac{x}{S} + (1-x)} \]

\[ 1.67 = \frac{1}{\frac{80\%}{2} + (1-80\%)} \]
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:
    \[
    \text{Speedup} = \frac{1}{(1 - X_{\text{Opt1}} - X_{\text{Opt2}}) + \frac{X_{\text{Opt1}}}{S_{\text{Opt1}}} + \frac{X_{\text{Opt2}}}{S_{\text{Opt2}}}}
    \]
  - If optimization #1 and optimization #2 are not dis-joint:
    \[
    S = \frac{1}{(1 - X_{\text{Opt1\Only}} - X_{\text{Opt2\Only}} - X_{\text{Opt1\&Opt2}}) + \frac{X_{\text{Opt1}}}{S_{\text{Opt1\Only}}} + \frac{X_{\text{Opt2}}}{S_{\text{Opt2\Only}}} + \frac{X_{\text{Opt1\&Opt2}}}{S_{\text{Opt1\&Opt2}}}}
    \]

\[
\text{total execution time} = 1
\]
Amdahl’s Law for multicore processors

• Assume that we have an application, in which 50% of the application can be fully parallelized with 2 processors. Assuming 80% of the parallelized part can be further parallelized with 4 processors, what’s the speed up of the application running on a 4-core processor?

Code can be optimized for 2-core = 50%*(1-80%) = 10%

Code can be optimized for 4-core = 50%*80% = 40%

\[
\text{Speedup}_{\text{quad}} = \frac{1}{(1-0.5) + \frac{0.10}{2} + \frac{0.40}{4}} = 1.54
\]
Amdahl’s Law for multiple optimizations

• Assume that memory access takes 30% of execution time.
  • Cache can speedup 80% of memory operation by a factor of 4
  • L2 cache can speedup 50% of the remaining 20% by a factor of 2

What’s the total speedup?

A. 1.22
B. 1.23
C. 1.24
D. 2.63
E. 2.86

Execution time can be optimized by L1 only = 30%*80% = 24%
Execution time can be optimized by L2 only = 30%*50%*20% = 3%

Speedup = \( \frac{1}{(1-0.27) + \frac{0.24}{4} + \frac{0.03}{2}} = 1.24 \)
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...)

![StarCraft II Wings of Liberty benchmark](image)
Case study: LOL

- Corollary #2
- The CPU is not the main performance bottleneck
- CPU parallelism doesn’t help, either
- 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 response time of a single task
- If you want to transfer a 2 Peta-Byte video from UCLA
  - 125 miles (201.25 km) from UCSD
  - Assume that you have a 100Gbps ethernet
    - 2 Peta-byte over 167772 seconds = 1.94 Days
    - 22.5TB in 30 minutes
    - Bandwidth: 100 Gbps
<table>
<thead>
<tr>
<th></th>
<th>Toyota Prius</th>
<th>10Gb Ethernet</th>
</tr>
</thead>
<tbody>
<tr>
<td>bandwidth</td>
<td>290GB/sec</td>
<td>100 Gb/s or 12.5GB/sec</td>
</tr>
<tr>
<td>latency</td>
<td>4 hours</td>
<td>2 Peta-byte over 167772 seconds = 1.94 Days</td>
</tr>
<tr>
<td>response time</td>
<td>You see nothing in the first 4 hours</td>
<td>You can start watching the movie as soon as you get a frame!</td>
</tr>
<tr>
<td></td>
<td>• 125 miles (201.25 km) from UCSD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 75 MPH on highway!</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 50 MPG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Max load: 374 kg = 2,770 hard drives (2TB per drive)</td>
<td></td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th></th>
<th>GFLOPS</th>
<th>clock rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>XBOX One</td>
<td>1310</td>
<td>1.75 GHz</td>
</tr>
<tr>
<td>PS4</td>
<td>1843</td>
<td>1.6 GHz</td>
</tr>
<tr>
<td>Core i7 EE 3970X + AMD Radeon 6990</td>
<td>5099.0</td>
<td>3.5 GHz</td>
</tr>
</tbody>
</table>
Is GFLOPS (Giga FLoating-point Operations Per Second) a good metric?

\[
\text{GFLOPS} = \frac{\# \text{ of floating point instructions}}{10^9} \frac{1}{\text{Execution Time}}
\]

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

- 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
Power & Energy
Power

- **Dynamic power:** $P = aCV^2f$
  - $a$: switches per cycle
  - $C$: capacitance
  - $V$: voltage
  - $f$: frequency, usually linear with $V$
  - Doubling the clock rate consumes more power than a quad-core processor!

- **Static/Leakage power** becomes the dominant factor in the most advanced process technologies.

- **Power is the direct contributor of “heat”**
  - Packaging of the chip
  - Heat dissipation cost
Energy

- Energy = P * ET
- The electricity bill and battery life is related to energy!
- Lower power does not necessary means better battery life if the processor slow down the application too much
Double Clock Rate or Double the Processors?

• Assume 60% of the application can be fully parallelized with 2-core or speedup linearly with clock rate. Should we double the clock rate or duplicate a core?

\[
\text{Speedup}_{2\text{-core}} = \frac{1}{(1-0.6)+\frac{0.6}{2}} = 1.43
\]

\[
\text{Power}_{2\text{-core}} = 2x
\]

\[
\text{Energy}_{2\text{-core}} = 2 \times \left[ \frac{1}{1/(1.43)} \right] = 1.39
\]

\[
\text{Speedup}_{2\times\text{Clock}} = 2
\]

\[
\text{Power}_{2\times\text{Clock}} = 8x
\]

\[
\text{Energy}_{2\times\text{Clock}} = \frac{8}{2} = 4
\]