Performance (II)

Hung-Wei Tseng
Recap: Performance Equation

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

- \( ET = \text{IC} \times \text{CPI} \times \text{Cycle Time} \)
- \( \text{IC} \) (Instruction Count)
  - ISA, compiler, algorithm, programming language, \textit{programmer}
- \( \text{CPI} \) (Cycles Per Instruction)
  - Machine implementation, microarchitecture, compiler, application, algorithm, programming language, \textit{programmer}
- \( \text{Cycle Time} \) (Seconds Per Cycle)
  - Process technology, microarchitecture, ISA, \textit{programmer}
Today’s CSE141

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

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

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

Total execution time = 1

Just another form to represent speedup!
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

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

\[
\begin{align*}
\text{Speedup} &= \frac{10}{10-1} = 1.111 \\
1.111 &= \frac{1}{\frac{20\%}{S} + (1-20\%)} \\
S &= 2
\end{align*}
\]
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

\[ \text{Speedup} = \frac{10}{10-5} = 2 \]

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

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

\[ S = -0.66 \]

Is this possible?
Amdahl’s Corollary #1

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

$$S = \text{infinity}$$

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

$$S_{\text{max}} = \frac{1}{1 - x}$$
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?

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

\[ 1.25 = \frac{1}{(1-0.20)} \]

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 = $\frac{1}{\frac{x}{S} + (1-x)}$</td>
<td>Speedup = $\frac{1}{\frac{x}{S} + (1-x)}$</td>
</tr>
<tr>
<td>$1.11 = \frac{1}{\frac{20%}{2} + (1-20%)}$</td>
<td>$1.16 = \frac{1}{\frac{35%}{20/12} + (1-35%)}$</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...

Common case

- $7x \Rightarrow 1.4x$
- $4x \Rightarrow 1.3x$
- $1.3x \Rightarrow 1.1x$

Total $= \frac{20}{10} = 2x$

- 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
  • 10.3 seconds in reading the input file
  • 104.6 seconds in sorting
  • sorting accounts for \( \frac{104.6}{10.3 + 104.6} = 91\% \) of the execution time

• GPU gives you 31x speed up!
  • Only takes 3.3 seconds in sorting now
  • sorting accounts for \( \frac{3.3}{10.3 + 3.3} = 24\% \) of the execution time

• New bottleneck emerges!
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?

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

- Recent advances in process technology have quadruple 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.

\[
S_{\text{quad-core}} = \frac{1}{\frac{x}{S} + (1-x) + \frac{1}{40\%} + (1-40\%)} \\
1.43 = \frac{1}{\frac{40\%}{4} + (1-40\%)}
\]

\[
S_{\text{dual-core}} = \frac{1}{\frac{x}{S} + (1-x) + \frac{1}{80\%} + (1-80\%)} \\
1.67 = \frac{1}{\frac{80\%}{2} + (1-80\%)}
\]
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:
    \[
    S = \frac{1}{(1 - X_{\text{Opt1Only}} - X_{\text{Opt2Only}} - X_{\text{Opt1&Opt2}}) + \frac{X_{\text{Opt1}}}{S_{\text{Opt1Only}}} + \frac{X_{\text{Opt2}}}{S_{\text{Opt2Only}}} + \frac{X_{\text{Opt1&Opt2}}}{S_{\text{Opt1&Opt2}}}}
    \]

<table>
<thead>
<tr>
<th>X_{\text{Opt1Only}}</th>
<th>X_{\text{Opt2Only}}</th>
<th>X_{\text{Opt1&amp;Opt2}}</th>
</tr>
</thead>
</table>

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

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 performance chart](chart.png)
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
Or ...

<table>
<thead>
<tr>
<th></th>
<th>Toyota Prius</th>
<th>10Gb Ethernet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bandwidth</strong></td>
<td>290GB/sec</td>
<td>100 Gb/s or 12.5GB/sec</td>
</tr>
<tr>
<td><strong>latency</strong></td>
<td>4 hours</td>
<td>2 Peta-byte over 167772 seconds = 1.94 Days</td>
</tr>
<tr>
<td><strong>response time</strong></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>
</tbody>
</table>

- Toyota Prius:
  - 125 miles (201.25 km) from UCSD
  - 75 MPH on highway!
  - 50 MPG
  - Max load: 374 kg = 2,770 hard drives (2TB per drive)

- 10Gb Ethernet:
  - Response time: 2 Peta-byte over 167772 seconds = 1.94 Days
  - You can start watching the movie as soon as you get a frame!
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
### GFLOPS (Giga FLoating-point Operations Per Second)

<table>
<thead>
<tr>
<th></th>
<th>GFLOPS</th>
<th>Clock Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XBOX One</strong></td>
<td>1310</td>
<td>1.75 GHz</td>
</tr>
<tr>
<td><strong>PS4</strong></td>
<td>1843</td>
<td>1.6 GHz</td>
</tr>
<tr>
<td><strong>Core i7 EE 3970X + AMD Radeon 6990</strong></td>
<td>5099.0</td>
<td>3.5 GHz</td>
</tr>
</tbody>
</table>
Is GFLOPS (Giga FLoating-point Operations Per Second) a good metric?

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

- 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 necessarily mean better battery life if the processor slows 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?

\[
\text{Speedup}_{2\text{-core}} = \frac{1}{(1-0.6)+\frac{0.6}{2}} = 1.43
\]
\[
\text{Power}_{2\text{-core}} = 2x
\]
\[
\text{Energy}_{2\text{-core}} = 2 \times \left[\frac{1}{1/(1.43)}\right] = 1.39
\]
\[
\text{Speedup}_{2\times\text{Clock}} = 2
\]
\[
\text{Power}_{2\times\text{Clock}} = 8x
\]
\[
\text{Energy}_{2\times\text{Clock}} = \frac{8}{2} = 4
\]
Announcement

• Reading quiz due tomorrow