Performance Example

- Assume that we have an application composed with a total of 500000 instructions, in which 20% of them are the load/store instructions with an average CPI of 6 cycles, and the rest instructions are integer instructions with average CPI of 1 cycle.

- If we double the CPU clock rate to 4GHz but keep using the same memory module, the average CPI for load/store instruction will become 12 cycles. What’s the performance improvement after this change?

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

\[ ET_{\text{new}} = 500000 \times (0.8 \times 1 + 0.2 \times 12) \times 0.25\text{ ns} = 400000\text{ ns} \]

Speedup = \[ \frac{ET_{\text{old}}}{ET_{\text{new}}} = \frac{500000}{400000} = 1.25 \]
Assume that we have an application composed with a total of 500000 instructions, in which 20% of them are the load/store instructions with an average CPI of 6 cycles, and the rest instructions are integer instructions with average CPI of 1 cycle.

If we double the CPU clock rate to 4GHz but keep using the same memory module, the average CPI for load/store instruction will become 12 cycles. What’s the performance improvement after this change?

A. No change
B. 1.25
C. 1.5
D. 2
E. None of the above
Execution Time = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}
What affects performance
Identify the performance bottleneck

Why does an Intel Core i7 @ 3.5 GHz usually perform better than an Intel Core i5 @ 3.5 GHz or AMD FX-8350@4GHz?

A. Because the instruction count of the program are different
B. Because the clock rate of AMD FX is higher
C. Because the CPI of Core i7 is better
D. Because the clock rate of AMD FX is higher and CPI of Core i7 is better
E. None of the above

Sysbench 2014 from http://www.anandtech.com/
Identify the performance bottleneck

Why does an Intel Core i7 @ 3.5 GHz usually perform better than an Intel Core i5 @ 3.5 GHz or AMD FX-8350@4GHz?

ET = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}

ET = \frac{\text{IC}}{\text{CPI}} \times \text{Cycle Time}

- Every time when the question asks you about “performance”, thinking about the performance equation first!
How programmer affects performance?

- \( ET = IC \times CPI \times CT \)
- What can a programmer affect?
  - A. IC
  - B. IC & CPI
  - C. IC, CPI & CT
  - D. IC & CT
Demo: programmer & performance

• Row-major, column major
  • How do you know this?
• Let’s identify where the performance gain is from!
  • Using “performance counters”
  • You may use “perf stat” in linux
  • You can also create your own functions to obtain counter values
Which of the following programming language needs to highest instruction count to print “Hello, world!” on screen?

A. C
B. C++
C. Java
D. Perl
E. Python
Programming languages

• How many instructions are there in “Hello, world!”

<table>
<thead>
<tr>
<th></th>
<th>Instruction count</th>
<th>LOC</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>480k</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>C++</td>
<td>2.8M</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Java</td>
<td>166M</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Perl</td>
<td>9M</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Python</td>
<td>30M</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
**Applications**

- Different applications can have different CPIs on the same machine
Compiler

- Compiler can change the combination of instructions and lead to different CPIs, instruction counts.
**Summary: 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, programmer
Amdahl’s Law
Amdahl’s Law

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

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

- Assume that we have an application composed with a total of \(500000\) instructions, in which 20% of them are the load/store instructions with an average CPI of 6 cycles, and the rest instructions are integer instructions with average CPI of 1 cycle.
- If we double the CPU clock rate to 4GHz but keep using the same memory module, the average CPI for load/store instruction will become 12 cycles. What’s the performance improvement after this change?

How much time in load/store?
\[
500000 \times (0.2 \times 6) \times 0.5 \text{ ns} = 300000 \text{ ns}
\]
60%

How much time in the rest?
\[
500000 \times (0.8 \times 1) \times 0.5 \text{ ns} = 200000 \text{ ns}
\]
40%
• Assume that we have an application composed with a total of 500000 instructions, in which 20% of them are the load/store instructions with an average CPI of 6 cycles, and the rest instructions are integer instructions with average CPI of 1 cycle.

• If we double the CPU clock rate to 4GHz but keep using the same memory module, the average CPI for load/store instruction will become 12 cycles. What’s the performance improvement after this change?

\[
\text{Speedup} = \frac{1}{\frac{0.4}{2} + (1-0.4)} = 1.25
\]

\[
\text{Speedup} = \frac{1}{0.8} = 1.25
\]
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

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

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

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

\[
S = -0.66
\]

Is this possible?
Amdahl’s Corollary #1

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

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

$$S = \text{infinity}$$
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?

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

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

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 ALU operations.
- 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 ALU operations
• 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))</td>
<td>Speedup = (\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 ALU operations**
- 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 the **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.
Identify the most time consuming part

- Compile your program with -pg flag
- Run the program
  - It will generate a gmon.out
  - gprof your_program gmon.out > your_program.prof
- It will give you the profiled result in your_program.prof
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.
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?

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

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

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

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

\[ 1.67 = \frac{1}{\frac{80\%}{2} + (1-80\%)} \]