Computer Organization and Architecture

Module 3

Amdahl's Law – Quantative Principles in Design

Prof. Indranil Sengupta

Dr. Sarani Bhattacharya

Department of Computer Science and Engineering

IIT Kharagpur

1

AMDAHL'S LAW

2

2

Introduction

Gene Amdahl

- Amdahl's law was established in 1967 by Gene Amdahl.
- Basically provides an understanding on scaling, limitations and economics of parallel computing.
- Forms the basis for quantitative principles in computer system design.
  - Can be applied to other application domains as well.

3

3

What is Amdahl's Law?

- It can be used to find the maximum expected improvement of an overall system when only *part of the system* is improved.
- It basically states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.
- Very useful to check whether any proposed improvement can provide expected return.
  - Used by computer designers to enhance only those architectural features that result in reasonable performance improvement.
  - Referred to as *quantitative principles in design*.

4

4

- Amdahl's law demonstrates the *law of diminishing returns*.
- An example:
  - Suppose we are improving a part of the computer system that affects only 25% of the overall task.
  - The improvement can be *very little* or *extremely large*.
  - With "*infinite*" speedup, the 25% of the task can be done in "*zero*" time.
  - Maximum possible speedup = XT_{orig} / XT_{new} = 1 / (1 – 0.25) = 1.33

We can never get a speedup of more than 1.33

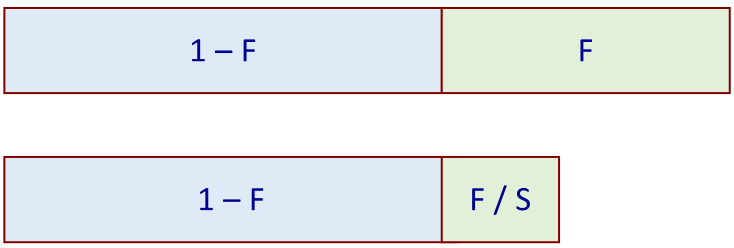
5

5

- Amdahl's law concerns the speedup achievable from an improvement in computation that affects a fraction *F* of the computation, where the improvement has a speedup of *S*.

Before improvement

After improvement



6

6

- Execution time before improvement: $(1 - F) + F = 1$
- Execution time after improvement: $(1 - F) + F / S$
- Speedup obtained:

$$\boxed{\text{Speedup} = \frac{1}{(1 - F) + F / S}}$$

- As $S \rightarrow \infty$, $Speedup \rightarrow 1 / (1 - F)$
  - The fraction F limits the maximum speedup that can be obtained.

7

7

- Illustration of law of diminishing returns: **1 / (1 – 0.25) = 1.33**
  - Let $F = 0.25$.
  - The table shows the speedup ($= 1 / (1 - F + F / S)$) for various values of S.

S	Speedup
1	1.00
2	1.14
5	1.25
10	1.29

S	Speedup
50	1.32
100	1.33
1000	1.33
100,000	1.33

8

8

- Illustration of law of diminishing returns:
  - Let *F = 0.75*.
  - The table shows the speedup for various values of *S*.

1 / (1 – 0.75) = 4.00

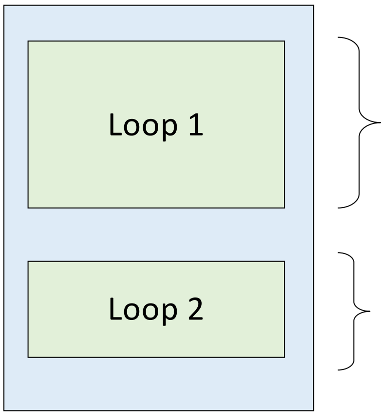
S	Speedup
1	1.00
2	1.60
5	2.50
10	3.08

S	Speedup
50	3.77
100	3.88
1000	3.99
100,000	4.00

9

9

Design Alternative using Amdahl's law



500 lines 10% of total execution time

20 lines 90% of total execution time

10

10

- Some examples:
  - We make 10% of a program 90X faster, speedup = 1 / (0.9 + 0.1 / 90) = 1.11
  - We make 90% of a program 10X faster, speedup = 1 / (0.1 + 0.9 / 10) = 5.26
  - We make 25% of a program 25X faster, speedup = 1 / (0.75 + 0.25 / 25) = 1.32
  - We make 50% of a program 20X faster, speedup = 1 / (0.5 + 0.5 / 20) = 1.90
  - We make 90% of a program 50X faster, speedup = 1 / (0.1 + 0.9 / 50) = 8.47

11

11

Example 1

- Suppose we are running a set of programs on a RISC processor, for which the following instruction mix is observed:

Operation	Frequency	CPI_i
Load	20 %	5
Store	8 %	3
ALU	60 %	1
Branch	12 %	2

W_i * CPI_i	% Time
1.00	0.48
0.24	0.12
0.60	0.29
0.24	0.11

CPI = 2.08

1 / 2.08

We carry out a design enhancement by which the CPI of Load instructions reduces from 5 to 2. What will be the overall performance improvement?

12

12

Operation	Frequency	CPI_i
Load	20 %	5
Store	8 %	3
ALU	60 %	1
Branch	12 %	2

$W_i * CPI_i$	% Time
1.00	0.48
0.24	0.12
0.60	0.29
0.24	0.11

CPI = 2.08

1 / 2.08

Fraction enhanced F = 0.48

Fraction unaffected 1 – F = 1 – 0.48 = 0.52

Enhancement factor S = 5 / 2 = 2.5

Therefore, speedup is

$$\frac{1}{(1-F)+F/S} = \frac{1}{0.52+0.48/2.5} = 1.40$$

13

13

Example 2

- The execution time of a program on a machine is found to be 50 seconds, out of which 42 seconds is consumed by multiply operations. It is required to make the program run 5 times faster. By how much must the speed of the multiplier be improved?

- Solution:
  - Here, F = 42 / 50 = 0.84
  - According to Amdahl's law,

    5 = 1 / (0.16 + 0.84 / S)

    or, 0.80 + 4.2 / S = 1

    or, S = 21

14

14

Example 2a

- The execution time of a program on a machine is found to be 50 seconds, out of which 42 seconds is consumed by multiply operations. It is required to make the program run **8** times faster. By how much must the speed of the multiplier be improved?

- Solution:
  - Here, $F = 42 / 50 = 0.84$
  - According to Amdahl's law,

    $8 = 1 / (0.16 + 0.84 / S)$

    or, $1.28 + 6.72 / S = 1$

    or, $S = -24$

No amount to speed improvement in the multiplier can achieve this.

Maximum speedup achievable:
$1 / (1 - F) = 6.25$

15

15

Example 3

- Suppose we plan to upgrade the processor of a web server. The CPU is 30 times faster on search queries than the old processor. The old processor is busy with search queries 80% of the time. Estimate the speedup obtained by the upgrade.
- Solution:
  - Here, $F = 0.80$ and $S = 30$
  - Thus, speedup $= 1 / (0.20 + 0.80 / 30) = 4.41$

16

16

Example 4

- The total execution time of a typical program is made up of 60% of CPU time and 40% of I/O time. Which of the following alternatives is better?
  - a) Increase the CPU speed by 50%
  - b) Reduce the I/O time by half

Assume that there is no overlap between CPU and I/O operations.

CPU	I/O	CPU	I/O	CPU	I/O

- Increase CPU speed by 50%
  - Here, F = 0.60 and S = 1.5
  - Speedup = 1 / (0.40 + 0.60 / 1.5) = 1.25
- Reduce the I/O time by half
  - Here, F = 0.40 and S = 2
  - Speedup = 1 / (0.60 + 0.40 / 2) = 1.25

Thus, both the alternatives result in the same speedup.

17

17

Example 5

- Suppose that a compute-intensive bioinformatics program is running on a given machine X, which takes 10 days to run. The program spends 25% of its time doing integer instructions, and 40% of time doing I/O. Which of the following two alternatives provides a better tradeoff?
  - a) Use an optimizing compiler that reduces the number of integer instructions by 30% (assume all integer instructions take the same time).
  - b) Optimizing the I/O subsystem that reduces the latency of I/O operations from 10 µsec to 5 µsec (that is, speedup of 2).

- Alternative (a):
  - Here, F = 0.25 and S = 100 / 70
  - Speedup = 1 / (0.75 + 0.25 * 70 / 100) = 1.08
- Alternative (b):
  - Here, F = 0.40 and S = 2
  - Speedup = 1 / (0.60 + 0.40 / 2) = 1.25

18

18

Extension to Multiple Enhancements

- Suppose we carry out multiple optimizations to a program:
  - Optimization 1 speeds up a fraction F_1 of the program by a factor S_1
  - Optimization 2 speeds up a fraction F_2 of the program by a factor S_2

$1 - F_1 - F_2$	F_1	F_2
$1 - F_1 - F_2$	F_1 / S_1	F_2 / S_2

Speedup

$$\frac{1}{(1 - F_1 - F_2) + F_1 / S_1 + F_2 / S_2}$$

19

19

- In the calculation as shown, it is assumed that F_1 and F_2 are disjoint.
  - S_1 and S_2 do not apply to the same portion of execution.
- If it is not so, we have to treat the overlap as a separate portion of execution and measure its speedup independently.
  - F_{1only} , F_{2only} , and $F_{1\&2}$ with speedups S_{1only} , S_{2only} , and $S_{1\&2}$

$1 - F_{1only} - F_{2only} - F_{1\&2}$	F_{1only}	$F_{1\&2}$	F_{2only}
$1 - F_{1only} - F_{2only} - F_{1\&2}$	F_{1only} / S_{1only}	$F_{1\&2} / S_{1\&2}$	F_{2only} / S_{2only}

$$\textbf{Speedup} = \frac{1}{(1 - F_{1only} - F_{2only} - F_{1\&2}) + F_{1only} / S_{1only} + F_{2only} / S_{2only} + F_{1\&2} / S_{1\&2}}$$

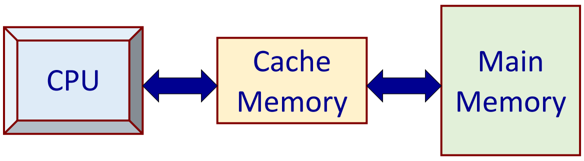
20

20

- General expression:
  - Assume *m* enhancements of fractions $F_1, F_2, \ldots, F_m$ by factors of $S_1, S_2, \ldots, S_m$ respectively.

$$\text{Speedup} = \frac{1}{(1 - \sum_{i=1}^{m} F_i) + \sum_{i=1}^{m} \frac{F_i}{S_i}}$$

Example 6



- Consider an example of memory system.
  - Main memory and a fast memory called cache memory.
  - Frequently used parts of program/data are kept in cache memory.
  - Use of the cache memory speeds up memory accesses by a factor of 8.
  - Without the cache, memory operations consume a fraction 0.40 of the total execution time.
  - Estimate the speedup.

Solution

$$\textbf{Speedup} = \frac{1}{(1 - F) + F / S} = \frac{1}{(1 - 0.4) + 0.4 / 8} = 1.54$$

Example 7

- Now we consider two levels of cache memory, L1-cache and L2-cache.

  Assumptions:
  - Without the cache, memory operations take 30% of execution time.
  - The L1-cache speeds up 80% of memory operations by a factor of 4.
  - The L2-cache speeds up 50% of the remaining 20% memory operations by a factor of 2.

  We want to find out the overall speedup.

- Solution:
  - Memory operations = 0.3
  - $F_{L1} = 0.3 * 0.8 = 0.24$
  - $S_{L1} = 4$
  - $F_{L2} = 0.3 * (1 - 0.8) * 0.5 = 0.03$
  - $S_{L2} = 2$

Speedup

$$\frac{1}{(1 - F_{L1} - F_{L2}) + F_{L1} / S_{L1} + F_{L2} / S_{L2}}$$

$$\frac{1}{(1 - 0.24 - 0.03) + 0.24 / 4 + 0.03 / 2}$$

$$= \mathbf{1.24}$$

23

23