CSE 213
Computer Architecture

Lecture 2: Computer Performance

Military Institute of Science and Technology

Performance

- *Performance is the key to understanding underlying motivation for the hardware and its organization*

- *Why is some hardware better than others for different programs?*

- *What factors of system performance are hardware related? (e.g., do we need a new machine, or a new operating system?)*

Performance

- Why is performance important?
  - For purchasers: to choose between computers
  - For designers: to make the sales pitch
- Defining performance is not straightforward!
  - An analogy with airplanes shows the difficulty

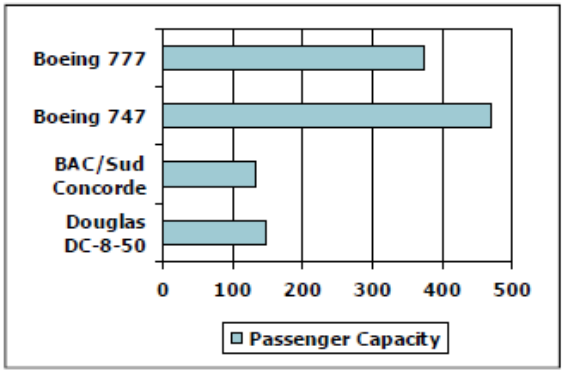
What do we measure? Define performance....

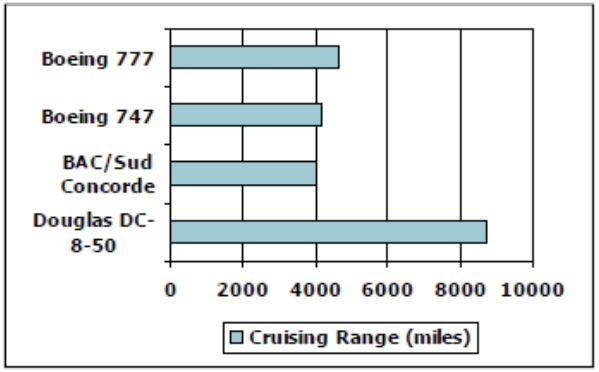
Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,424

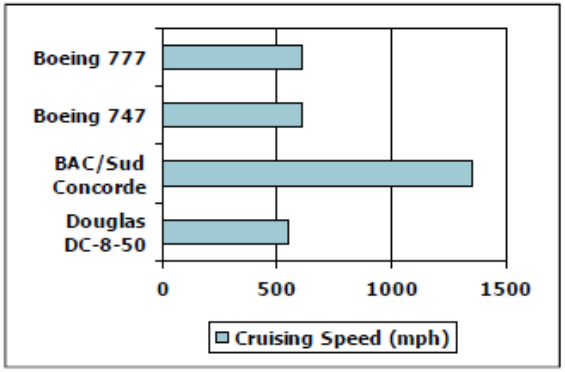
- How much faster is the Concorde compared to the 747?
- How much bigger is the Boeing 747 than the Douglas DC-8?

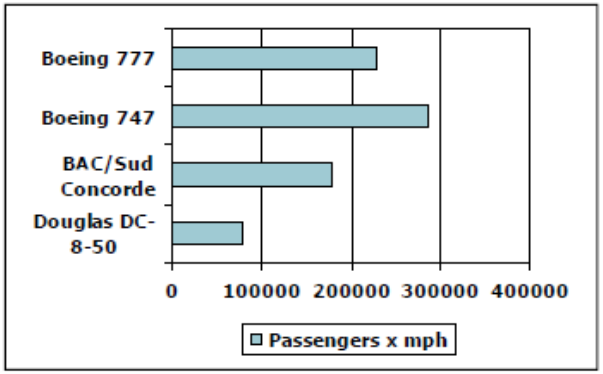
Defining **Performance**

- Which airplane has the best performance?









Computer Performance: TIME, TIME, TIME!!!

- *Response Time* (*elapsed time*, *latency*):
  - How long does it take to complete (start to finish) *a task*?
  - Eg: how long must *I* wait for the database query?

Individual user concerns…

Individual is more interested in response time. As a user of a smart phone/laptop, the one that responds faster is better!

Response time (computer): the total time required by computer to complete a task including :

Disk access *Memory access* *I/O activities* *OS overheads* *CPU exec. time* *etc*

Computer Performance: TIME, TIME, TIME!!!

- *Throughput*:
  - Total work done per unit time……(per hr,day etc)
  - how *many* jobs can the machine run at once?
  - what is the *average* execution rate?
  - how *much* work is getting done?

Systems manager concerns…

Response Time and Throughput

- *If we upgrade a machine with a new processor what do we increase?*

- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?

Relative Performance

- Define Performance = $1/\text{Execution Time}$
- $\text{Performance}_x > \text{Performance}_y$
- $1/\text{Execution Time}_x > 1/\text{Execution Time}_y$
- $\text{Execution Time}_y > \text{Execution Time}_x$

$$\text{Performance}_X/\text{Performance}_Y = \text{Execution time}_Y/\text{Execution time}_X = n$$

Relative Performance

- Define Performance = 1/Execution Time
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$$

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Execution Time

- ***Elapsed Time***
  - counts everything (*disk and memory accesses, waiting for I/O, running other programs, etc.*) from start to finish
  - **a useful number, but often not good for comparison purposes**

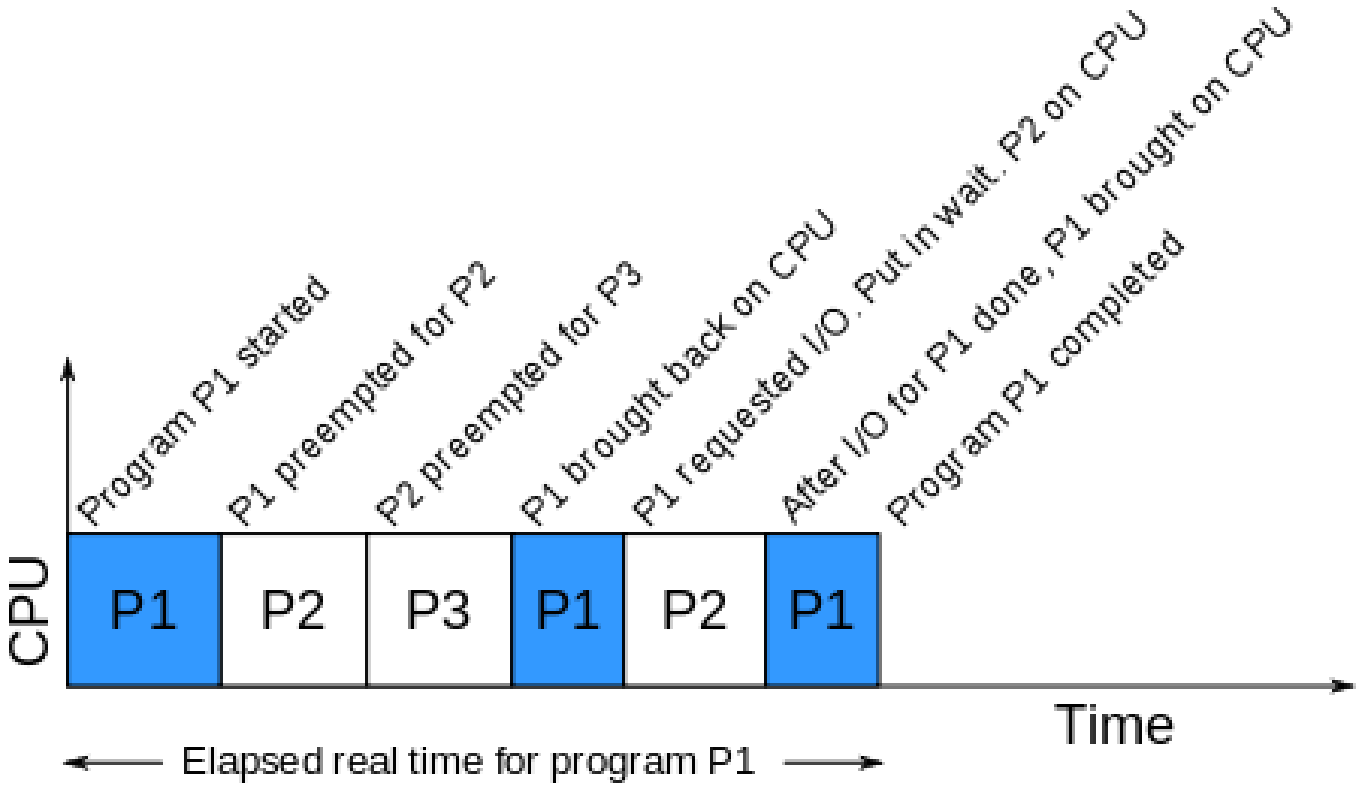
    Elapsed time = CPU time + wait time (I/O, other programs, etc.)

- ***CPU time***
  - **doesn't count waiting for I/O or time spent running other programs**
  - can be divided into *user CPU time* and *system CPU time* (OS calls)

    CPU time = user CPU time + system CPU time

- Our focus:
- *user CPU time* (*CPU execution time* or, simply, *execution time*)
  - **time spent executing the lines of code that are *in our program***
  - *For easier writing,* ***user CPU time*** *has been termed simply as* ***CPU time*** *in rest of the studies.*

Execution Time



CPU Clocking

Clock cycle. The speed of a computer processor, or CPU, is determined by the clock cycle, which is the amount of time between two pulses of an oscillator.

- Operation of digital hardware governed by a constant-rate clock
  - Clock period: duration of a clock cycle
    - e.g., 250ps = 0.25ns = 250×10^{-12}s
  - Clock frequency (rate): cycles per second
    - e.g., 4.0 GHz = 4000 MHz = 4.0×10^{9}Hz

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

Performance Equation - I

Example

- Our favorite program runs in 10 seconds on computer A, which has a 2Ghz. clock.
- We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.

- *What clock rate should we tell the designer to target?*

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles compared to A
- How fast must Computer B clock be i.e., what is the clock rate for Computer B?

$$\text{CPU Time}_B = \frac{\text{CPU Clock Cycles}_B}{\text{Clock Rate}_B}$$

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles compared to A
- How fast must Computer B clock be?

$$\text{CPU Time}_B = \frac{\text{CPU Clock Cycles}_B}{\text{Clock Rate}_B}$$

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

$$\text{CPU Time}_B = \frac{\text{CPU Clock Cycles}_B}{\text{Clock Rate}_B}$$

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

Performance Equation - II

Factors Influencing Performance

- **Execution time = clock cycle time x number of instrs x avg CPI**

- **Clock cycle time:** manufacturing process (how fast is each transistor), how much work gets done in each pipeline stage (more on this later)
- **Number of instructions:** the quality of the compiler and the instruction set architecture
- **CPI:** the nature of each instruction and the quality of the architecture implementation

CPU Time Example

Suppose we have two implementations of the same instruction set architecture. Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program, and computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program. Which computer is faster for this program and by how much?

CPU Time Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$

CPU Time Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster...

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

...by this much

Self Help

- Suppose we have two implementations of the same instruction set architecture (ISA). For some program:
  - machine A has a clock cycle time of 10 ns. and a CPI of 2.0
  - machine B has a clock cycle time of 20 ns. and a CPI of 1.2

- *Which machine is faster for this program, and by how much?*
- *If two machines have the same ISA, which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?*

Average cycles per instruction (CPI):

- CPI_i be the number of cycles required for instruction type ***i***, and I_i be the number of executed instructions of type ***i***

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_c}$$

The processor time *T* needed to execute a given program,

$$T = I_c \times CPI \times \tau$$

- τ = A constant cycle time
- I_c = Instruction count

CPI Example

- A compiler designer is trying to decide between two code sequences for a particular machine.
- Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C,

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

Code sequence	Instruction counts for each instruction class		
	A	B	C
1	2	1	2
2	4	1	1

- *Which code sequence has the most instructions? Which sequence will be faster? How much? What is the CPI for each sequence?*

CPI Example

Which code sequence has the most instructions?

Sequence 1 executes $2 + 1 + 2 = 5$ instructions. Sequence 2 executes $4 + 1 + 1 = 6$ instructions. Therefore, sequence 1 executes fewer instructions.

We can use the equation for CPU clock cycles based on instruction count and CPI to find the total number of clock cycles for each sequence:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{C}_i)$$

Which sequence will be faster?

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$

$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

What is the CPI for each sequence

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

MIPS Rate:

A common measure of performance for a processor is the rate at which instructions are executed, expressed as millions of instructions per second (MIPS) aka the **MIPS rate**

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

MIPS Example

EXAMPLE 2.2 Consider the execution of a program that results in the execution of 2 million instructions on a 400-MHz processor. The program consists of four major types of instructions. The instruction mix and the *CPI* for each instruction type are given below, based on the result of a program trace experiment:

Instruction Type	*CPI*	Instruction Mix (%)
Arithmetic and logic	1	60
Load/store with cache hit	2	18
Branch	4	12
Memory reference with cache miss	8	10

$$CPI = 0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) = 2.24$$

MIPS rate is $(400 \times 10^6)/(2.24 \times 10^6) \approx 178$.

Self Help

- Two different compilers are being tested for a 500 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require 1, 2 and 3 cycles (respectively). Both compilers are used to produce code for a large piece of software.
- Compiler 1 generates code with 5 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- Compiler 2 generates code with 10 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

- *Which sequence will be faster according to MIPS?*
- *Which sequence will be faster according to execution time?*

Another Example

A given application written in Java runs 15 seconds on a desktop processor. A new Java compiler is released that requires only 0.6 as many instructions as the old compiler. Unfortunately, it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler? Pick the right answer from the three choices below

a. $\frac{15 \times 0.6}{1.1} = 8.2 \text{ sec}$

b. $15 \times 0.6 \times 1.1 = 9.9 \text{ sec}$

c. $\frac{15 \times 1.1}{0.6} = 27.5 \text{ sec}$

MFLOPS

MFLOPS: Millions of floating-point operations per second

$$\text{MFLOPS rate} = \frac{\text{Number of executed floating} - \text{point operations in a program}}{\text{Execution time} \times 10^6}$$

List Performance Measure

- **Basic Measurement:**
  - Clock Speed
  - Instruction Execution Rate
    - MIPS
    - MFLOPS
- Calculating the Mean: calculating the mean value of a set of data points related to execution time.
  - Arithmetic Mean (AM)
  - Geometric mean (GM)
  - Harmonic mean (HM)

AM, GM, HM Formula

Arithmetic mean

$$AM = \frac{x_1 + \cdots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Geometric mean

$$GM = \sqrt[n]{x_1 \times \cdots \times x_n} = \left(\prod_{i=1}^{n} x_i\right)^{1/n} = e^{\left(\frac{1}{n}\sum_{i=1}^{n} \ln(x_i)\right)}$$

Harmonic mean

$$HM = \frac{n}{\left(\frac{1}{x_1}\right) + \cdots + \left(\frac{1}{x_n}\right)} = \frac{n}{\sum_{i=1}^{n}\left(\frac{1}{x_i}\right)} \qquad x_i > 0$$

Basic example: How AM Performs?

- Suppose we have a set of n benchmark programs and record the execution times of each program on a given system as **t1, t2, …, tn.**
- For simplicity, let us assume that each program executes the same number of operations **Z**
- The execution rate for each individual program is Ri = Z/ti.
- We use the AM to calculate the average execution rate.

$$AM = \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{1}{n}\sum_{i=1}^{n} \frac{Z}{t_i} = \frac{Z}{n}\sum_{i=1}^{n} \frac{1}{t_i}$$

- AM execution rate is proportional to the sum of the inverse execution times.

Basic example: How AM Performs?

The HM yields the following result:

$$HM = \frac{n}{\sum_{i=1}^{n}\left(\frac{1}{R_i}\right)} = \frac{n}{\sum_{i=1}^{n}\left(\frac{1}{Z/t_i}\right)} = \frac{nZ}{\sum_{i=1}^{n} t_i}$$

The HM is inversely proportional to the total execution time, which is the desired property.

Benchmarks and Spec

A = B + C

- In a complex instruction set computer (CISC), this instruction can be compiled into one processor instruction:

```
add mem(B), mem(C), mem(A)
```

- On a RISC machine, the compilation would look something like this:

```
load mem(B), reg(1);
load mem(C), reg(2);
add reg(1), reg(2), reg(3);
store reg(3), mem (A)
```

Both machines may execute the original high-level language instruction in about the same time.

- CISC machine is rated at 1 MIPS,
- RISC machine is rated at 4 MIPS.

Benchmarks and Spec

- Best Performance determined by running a real application
  - use programs typical of expected workload
  - or, typical of expected class of applications
    e.g., compilers/editors, scientific applications, graphics, etc.

- Benchmark suites
  - Each vendor announces a SPEC rating for their system
  - a measure of execution time for a fixed collection of programs
  - is a function of a specific CPU, memory system, IO system, operating system, compiler enables easy comparison of different systems

SPEC (System Performance Evaluation Corporation)

- Sponsored by industry but independent and self-managed – trusted by code developers and machine vendors
- Clear guides for testing, see www.spec.org
- Regular updates (benchmarks are dropped and new ones added periodically according to relevance)
- Specialized benchmarks for particular classes of applications

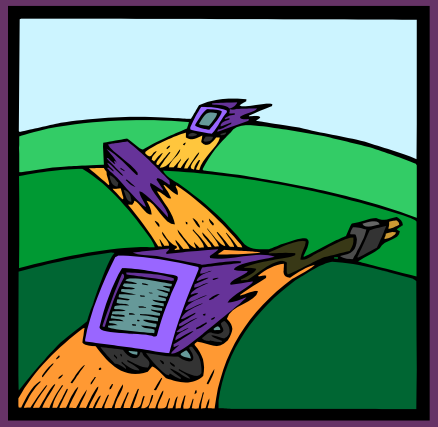
SPEC CPU

•The best known of the SPEC benchmark suites is SPEC CPU2006.

•The 2006 version includes 12 integer and 17 floating-point applications

• The SPEC rating specifies how much faster a system is, compared to a baseline machine – a system with SPEC rating 600 is 1.5 times faster than a system with SPEC rating 400

Amdahl's Law



- Gene Amdahl [AMDA67]
- Deals with the potential speedup of a program using multiple processors compared to a single processor
- Illustrates the problems facing industry in the development of multi-core machines
  - Software must be adapted to a highly parallel execution environment to exploit the power of parallel processing
- Can be generalized to evaluate and design technical improvement in a computer system

Amdahl's law

Amdahl's law states that in parallelization

- **f** is the proportion of a system or program that can be made parallel
- **1-f** is the proportion that remains serial

$$\text{Speedup} = \frac{\text{Time to execute program on a single processor}}{\text{Time to execute program on } N \text{ parallel processors}}$$

$$= \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$$

Two important conclusions can be drawn:

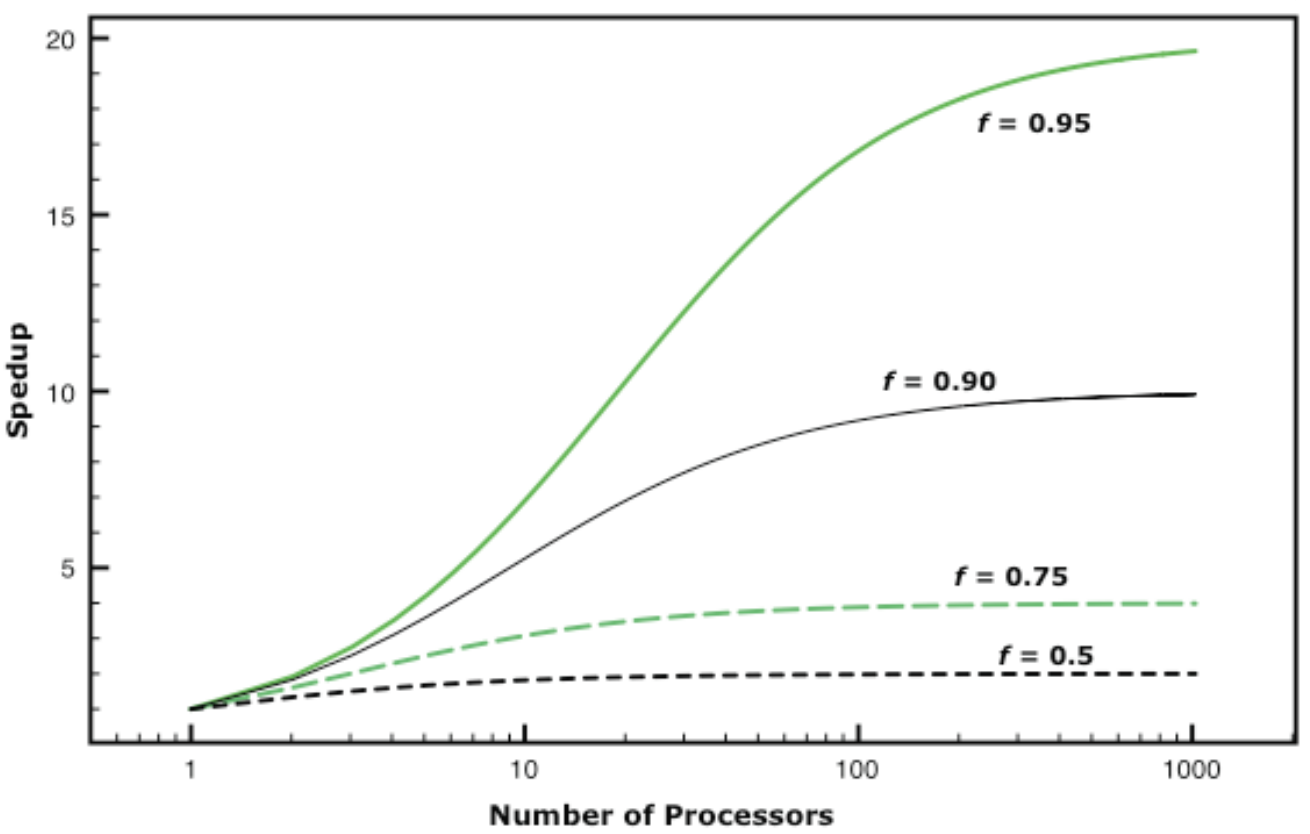
1. When f is small, the use of parallel processors has little effect.
2. As N approaches infinity, speedup is bound by 1/(1 - f), so that there are diminishing returns for using more processors.

Amdahl's law

If a program needs 20 hours using a single processor core, and a particular part of the program which takes one hour to execute cannot be parallelized, If there are no limitations of using processors, then calculate the **minimum execution time** of the program.

Answer:
while the remaining 19 hours (f = 0.95) of execution time can be parallelized, then regardless of how many processors are devoted to a parallelized execution of this program, the minimum execution time cannot be less than that critical one hour. Hence, the theoretical speedup is limited to at most 20 times ($1/(1 - f) = 20$).

Amdahl's Law


20
15
10
5
Spedup
f = 0.95
f = 0.90
f = 0.75
f = 0.5
1
10
100
1000
Number of Processors

Little's Law

- Fundamental and simple relation with broad applications
- Can be applied to almost any system that is statistically in steady state, and in which there is no leakage
- Queuing system
  - If server is idle an item is served immediately, otherwise an arriving item joins a queue
  - There can be a single queue for a single server or for multiple servers, or multiples queues with one being for each of multiple servers

Little's Law

- Average number of items in a queuing system equals the average rate at which items arrive multiplied by the time that an item spends in the system
  - Relationship requires very few assumptions
  - Because of its simplicity and generality it is extremely useful

Number of items in the system (L)= **the rate items enter and leave the system** (A / arrival rate / departure rate / throughput / λ) x **the average amount of time items spend in the system** (w/ lead time)

$L = A \times W$

Little's Law

You are estimating number of threads required by your server to execute clients requests efficiently and initially you starts 4 threads on the server. Request arrival rate on your server is 4 request/sec and each request takes fixed amount of time to complete with following time descriptions. The arrival rate is fixed and all new request arrivals have fixed service time.

Request_1= 0.2 sec; Request_2 =0.9 sec; Request_3=0.6 sec Request_4=0.5 sec

What improvement factor should you think for the maximization of your thread uses?

Little's Law

Since each request would be assigned to each thread so no waits are there so Average Response time is (0.2+0.9+0.6+0.5)/4=0.55

Request arrival rate is 4.

Applying Little's law to estimate required threads on server to serve requests is now as follows:

Required threads on server =Request arrival rate $*$ average response time=4$*$(0.55)=2.2

So, there should be 2 threads only for request arrivals of 4 req/sec with given service times,

and in this case any 2 requests must wait on each cycle of arrivals because we have only 2 resources (threads) and arrival rate is 4 requests/sec. So any 2 requests must wait and this results in increased response time. And two 2 threads on the server will always remain idle.

Summary

- Performance is specific to a particular program
  - total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
  - increases in clock rate (without adverse CPI affects)
  - improvements in processor organization that lower CPI
  - compiler enhancements that lower CPI and/or instruction count

Thank you ☺