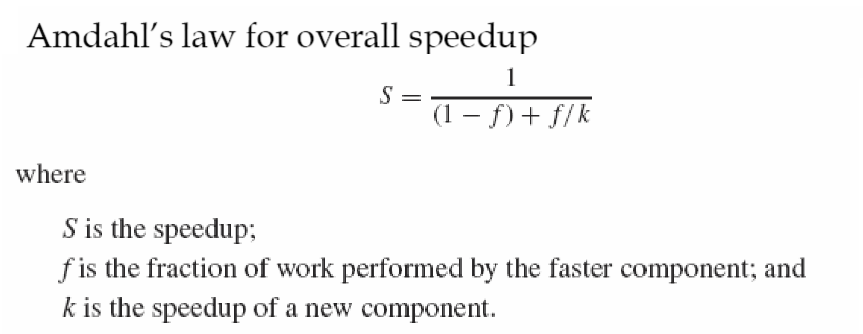
System Performance & Instruction Pipelining

Lecture 12

Speedup through Amdahl's law

Amdahl's Law states that

The overall *speedup* of a computer system depends on both the speedup in a particular component and how much that component is used by the system.



Amdahl's law for overall speedup

$$S = \frac{1}{(1 - f) + f/k}$$

where

S is the speedup;
f is the fraction of work performed by the faster component; and
k is the speedup of a new component.

Using Amdahl's law

Overall speedup if we make 90% of a program run 10 times faster.

F = 0.9 S = 10

$$\text{Overall Speedup} = \frac{1}{(1-0.9)+\frac{0.9}{10}} = \frac{1}{0.1+0.09} = 5.26$$

Overall speedup if we make 80% of a program run 20% faster.

F = 0.8 S = 1.2

$$\text{Overall Speedup} = \frac{1}{(1-0.8)+\frac{0.8}{1.2}} = \frac{1}{0.2+0.66} = 1.153$$

Exercise

- Suppose a program processing load consists of 60% CPU activity and 40% disk activity. You feel that the system is slow.
  - To upgrade your disks for Rs8,000 to make them 2.5 times as fast as they are currently.
  - To upgrade your CPU to make it 1.4 as fast for Rs5,000.
    - Which would you choose to yield the best performance improvement for the least amount of money?
    - Which option would you choose if you don't care about the money, but want a faster system?

Solution

a.

Fraction of work: 60% CPU, 40% disk.

$S_{CPU} = 1/((1-f)+(f/k)) = 1/((1-0.60)+(0.60/1.4)) = 1.2069$ or 20.69%

$S_{DISK} = 1/((1-f)+(f/k)) = 1/((1-0.40)+(0.40/2.5)) = 1.3158$ or 31.58%

•Choose the Disk upgrade

b.

CPU = Rs5000/20.69% = Rs241.66 per 1% increase in performance

Disk = Rs8000/31.58% = Rs253.32 per 1% increase in performance

– The CPU option gives a better performance improvement.

Cycles Per Instruction (CPI)

- Most computers run synchronously utilizing a CPU clock running at a constant clock rate:

  where: Clock rate = 1 / clock cycle

- A machine instruction is comprised of a number of micro operations which vary in number and complexity depending on the instruction and the exact CPU organization and implementation.
  - A micro operation is an elementary hardware operation that can be performed during one clock cycle.
  - This corresponds to one micro-instruction in microprogrammed CPUs.
    - Examples: register operations: shift, load, clear, increment, ALU operations: add , subtract, etc.

Cycles Per Instruction (CPI)

- A single machine instruction may take one or more cycles to complete termed as the Cycles Per Instruction (CPI).
  - Program execution time is one of the performance measure
- For a specific program compiled to run on a specific machine "A", the following parameters are provided:
  - The total instruction count of the program.
  - The average number of cycles per instruction (average CPI).
  - Clock cycle of machine "A"
- measure the performance of this machine

Program Execution Time

- How can we measure the performance of machine running a program?
  - The machine is said to be faster or has better performance running this program if the total execution time is shorter.
  - Thus the inverse of the total measured program execution time is a possible performance measure or metric:

$$\text{Performance}_A = 1 / \text{Execution Time}_A$$

How to compare performance of different machines?

What factors affect performance? How to improve performance?

Comparing Performance

- To compare the performance of two machines "A", "B" running a given program:

  $$\text{Performance}_A = 1 \;/\; \text{Execution Time}_A$$
  $$\text{Performance}_B = 1 \;/\; \text{Execution Time}_B$$

- Machine A is n times faster than machine B means: $n = \text{Performance}_A \;/\; \text{Performance}_B = \text{Execution Time}_B \;/\; \text{Execution Time}_A$

COA, BESE 15 B, Asst Prof Athar Mohsin 9

CPU Execution Time: The CPU Equation

- A program is comprised of a number of instructions, I
  - Measured in: instructions/program
- The average instruction takes a number of cycles per instruction (CPI) to be completed.
  - Measured in: cycles/instruction, CPI
- CPU has a fixed clock cycle time C = 1/clock rate
  - Measured in: seconds/cycle
- CPU execution time is the product of the above three parameters as follows:

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

$$T = I \times CPI \times C$$

COA, BESE 15 B, Asst Prof Athar Mohsin 10

CPU Execution Time

For a given program and machine:

CPI = Total program execution cycles / Instructions count

$\rightarrow$ CPU clock cycles = Instruction count x CPI

CPU execution time =

= CPU clock cycles x Clock cycle

= Instruction count x CPI x Clock cycle

= I x CPI x C

CPU Execution Time: Example

- A Program is running on a specific machine with the following parameters:
  - Total instruction count (I): 10,000,000 instructions
  - Average CPI for the program (CPI): 2.5 cycles/instruction.
  - CPU clock rate (C): 200 MHz.
- What is the execution time for this program:

$$\textbf{CPU time} = \frac{\textbf{Seconds}}{\textbf{Program}} = \frac{\textbf{Instructions}}{\textbf{Program}} \times \frac{\textbf{Cycles}}{\textbf{Instruction}} \times \frac{\textbf{Seconds}}{\textbf{Cycle}}$$

CPU time = Instruction count x CPI x Clock cycle

= 10,000,000 x 2.5 x 1 / clock rate

= 10,000,000 x 2.5 x 5×10^{-9}

= 0.125 seconds

Performance Comparison: Example

- From the previous example: A Program is running on a specific machine with the following parameters:
  - Total instruction count: 10,000,000 instructions
  - Average CPI for the program: 2.5 cycles/instruction.
  - CPU clock rate: 200 MHz.
- Using the same program with these changes:
  - A new compiler used: New instruction count 9,500,000
    New CPI: 3.0
  - Faster CPU implementation: New clock rate = 300 MHZ
- What is the speedup with the changes?

$$\textbf{Speedup} = \frac{\textbf{Old Execution Time}}{\textbf{New Execution Time}} = \frac{I_{old} \times CPI_{old} \times \text{Clock cycle}_{old}}{I_{new} \times CPI_{new} \times \text{Clock Cycle}_{new}}$$

Speedup = $(10{,}000{,}000 \times 2.5 \times 5\text{x}10^{-9}) / (9{,}500{,}000 \times 3 \times 3.33\text{x}10^{-9})$

= .125 / .095 = 1.32

or 32 % faster after changes.

Comparing Performance

- Example:

For a given program:

Execution time on machine A: Execution_A = 1 second

Execution time on machine B: Execution_B = 10 seconds

Performance_A / Performance_B = Execution Time_B / Execution Time_A

= 10 / 1 = 10

The performance of machine A is 10 times the performance of machine B when running this program, or:
Machine A is said to be 10 times faster than machine B when running this program.

Pipeline

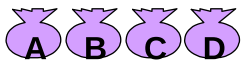
- It is observed that organization enhancements to the CPU can improve performance.
  - We have already seen that use of multiple registers rather than a single a accumulator, and various performance measures to improves the performance considerably.
- Another organizational approach, which is quite common, is instruction pipelining.
  - Pipelining is a particularly effective way of organizing parallel activity in a computer system.
- The basic idea is very simple
  - It is frequently encountered in manufacturing plants, where pipelining is commonly known as an assembly line operation
    - By laying the production process out in an assembly line, product at various stages can be worked on simultaneously.
  - This process is also referred to as pipelining, because, as in a pipeline, new inputs are accepted at one end before previously accepted inputs appear as outputs at the other end.

Real life example

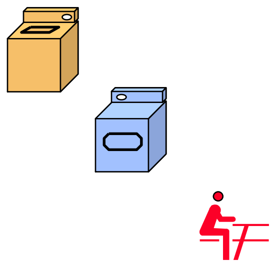
- A common example for a pipeline is a factory assembly line. Assume that there are three stages:
  1. Welding
  2. Painting
  3. Polishing
- For simplicity, assume that each task takes one hour.
  - If a single person were to work on the product it would take three hours to produce one product.
    - If we had three people, one person could work on each stage, upon completing their stage they could pass their product on to the next person (since each stage takes one hour there will be no waiting).
  - We could then produce one product per hour assuming the assembly line has been filled.

Traditional Pipeline Concept

- Laundry Example
- A, B, C, D each have one load of clothes to wash, dry, and fold

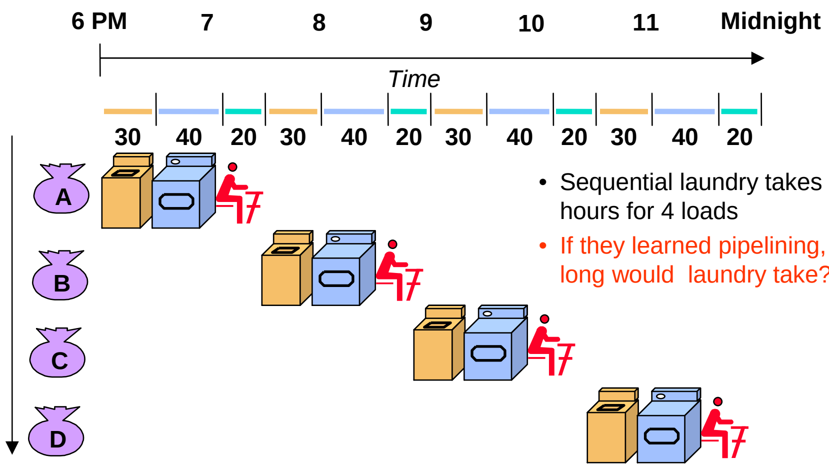


- Washer takes 30 minutes
- Dryer takes 40 minutes
- "Folder" takes 20 minutes



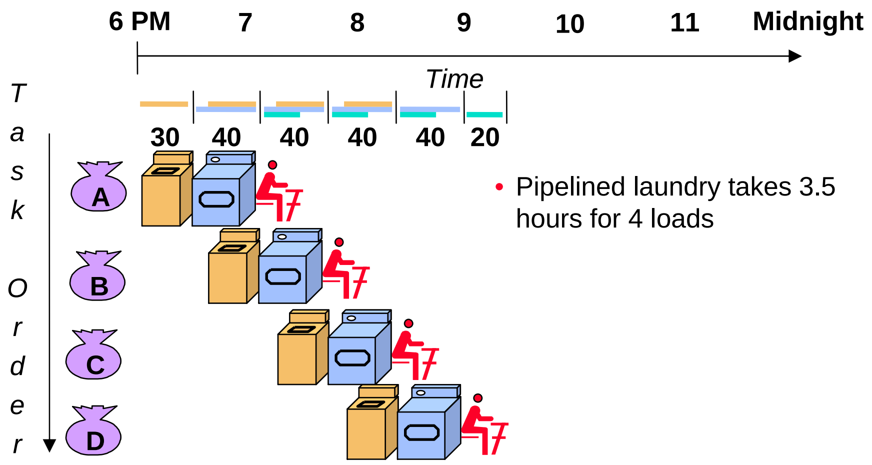
- Total Load 4
- Time to complete one load 1 hr 30 min

Traditional Pipeline Concept



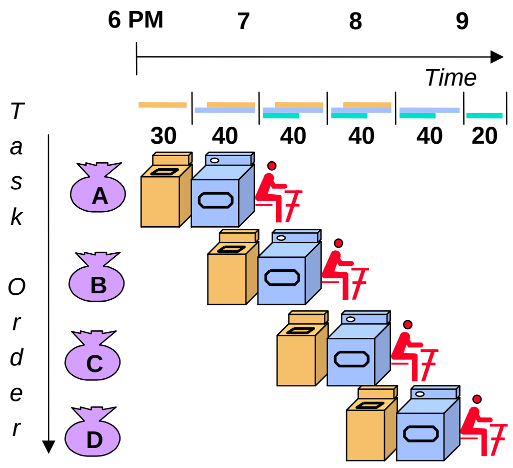
- Sequential laundry takes 6 hours for 4 loads
- If they learned pipelining, how long would laundry take?

Traditional Pipeline Concept



- Pipelined laundry takes 3.5 hours for 4 loads

Traditional Pipeline Concept



- Pipelining doesn’t help latency of single task, it helps throughput of entire workload
- Pipeline rate limited by slowest pipeline stage
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number pipe stages
- Unbalanced lengths of pipe stages reduces speedup
- Time to “fill” pipeline and time to “drain” it reduces speedup

Pipeline Latency & Throughput

- Pipeline throughput:
  - Instructions completed per second.
    - the number of instructions that the processor completes *each clock cycle*
- Pipeline latency:
  - How long does it take to execute a single instruction in the pipeline.
    - The amount of time that a single instruction takes to pass through the pipeline.
- While pipelining can reduce the a processor's cycle time and increase the instruction throughput, it increases the latency of the processor

Instruction level Pipelining

- Pipelining**:**
  - An implementation technique where multiple instructions are overlapped in execution.
  - The computer pipeline is divided in **stages**.
    - Each stage completes a part of an instruction in parallel.
    - The stages are connected one to the next to form a pipe - instructions enter at one end, progress through the stages, and exit at the other end
  - Pipelining does not decrease the time for individual instruction execution.
    - Instead, it increases instruction throughput.
  - The **throughput** of the instruction pipeline is determined by how often an instruction exits the pipeline.

The speedup from pipelining

- The speedup in completion rate versus a single-cycle implementation that's gained from pipelining is ideally equal to the number of pipeline stages.
  - A four-stage pipeline yields a four-fold speedup in the completion rate versus single-cycle,
  - A five-stage pipeline yields a five-fold speedup, so on
- This speedup is possible because:
  - The more pipeline stages in a processor, the more instructions the processor can work on simultaneously and the more instructions it can complete in a given period of time.

Characteristics of Pipelining

- If the stages of a pipeline are not balanced and one stage is slower than another, the entire throughput of the pipeline is affected.
- In terms of a pipeline within a CPU:
  - Each instruction is broken up into different stages
  - Ideally if each stage is balanced (all stages are ready to start at the same time and take an equal amount of time to execute.) the time taken per instruction (pipelined) is defined as:

    Time per instruction (unpipelined) / Number of stages

Characteristics Of Pipelining

- In terms of a CPU:
  - The implementation of pipelining has the effect of reducing the average instruction time, therefore reducing the average CPI.
    - EX: If each instruction in a microprocessor takes 5 clock cycles (unpipelined) and we have a 4 stage pipeline, the ideal average CPI with the pipeline will be 1.25 .