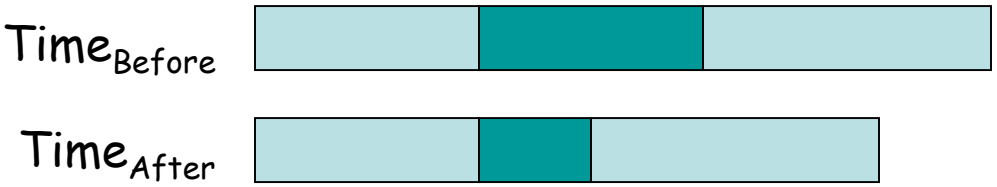
Amdahl's Law and Evaluating Performance

Based on slides created by Mark Hill and others

Slides adapted by Dr Sparsh Mittal

Compute Speedup – Amdahl's Law

Speedup is due to enhancement(E):

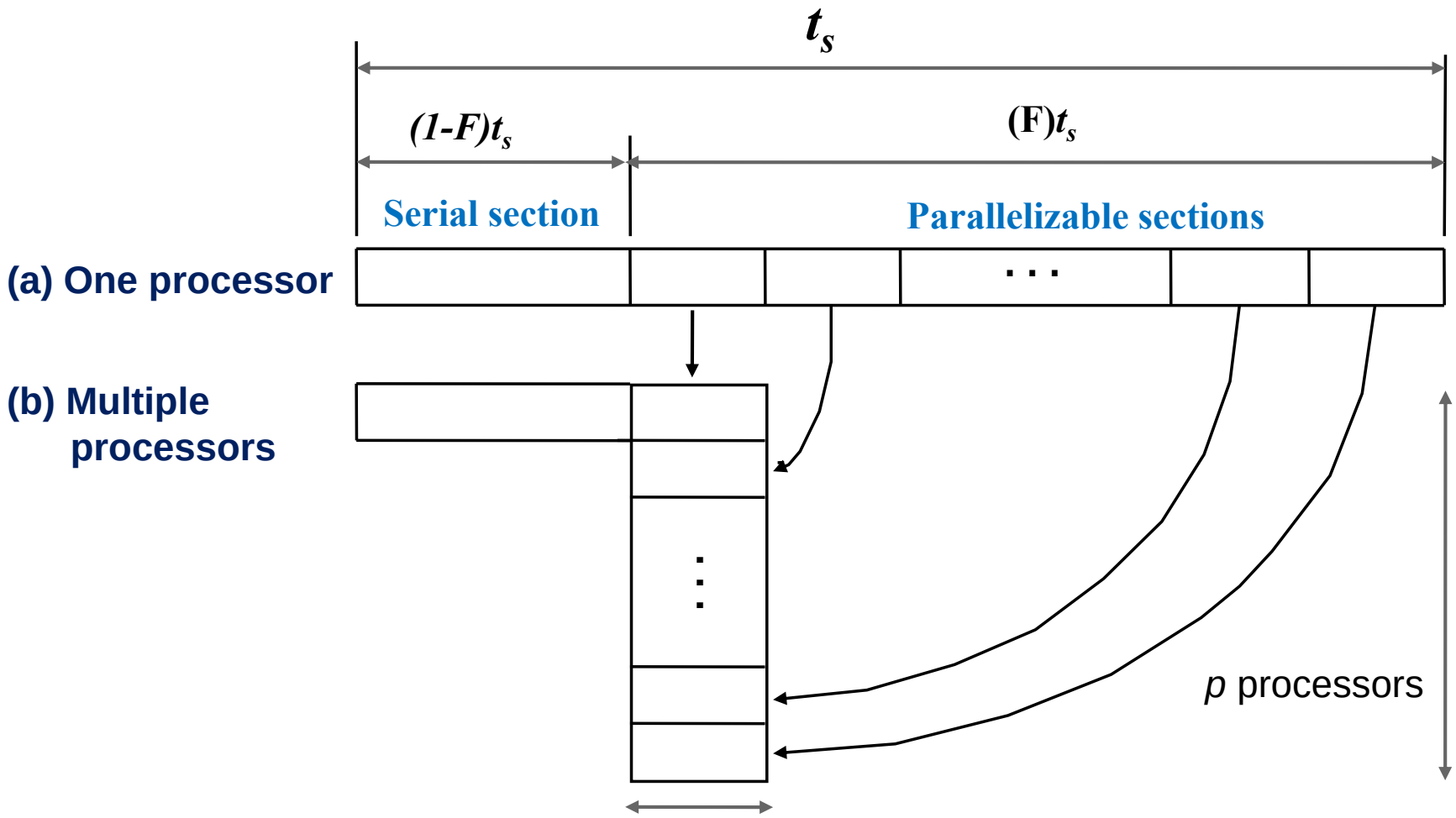


Let F be the fraction where enhancement is applied => Also, called parallel fraction and (1-F) as the serial fraction

$$\textit{Execution time}_{after} = \text{ExTime}_{before} \times \left[(1-F) + \frac{F}{S}\right]$$

$$\textit{Speedup(E)} = \frac{\text{ExTime}_{before}}{\text{ExTime}_{after}} = \frac{1}{\left[(1-F) + \frac{F}{S}\right]}$$

Even with infinite number of processors, maximum speedup is limited to 1/(1-F).


t_s
$(1-F)t_s$
$(F)t_s$
Serial section
Parallelizable sections
(a) One processor
(b) Multiple processors
p processors

Performance

- Which computer is fastest?
- Not so simple

Response Time vs. Throughput

- Is throughput = 1/av. response time?

  - Only if NO overlap
  - Otherwise, throughput > 1/av. response time

Principles of Computer Design

$$\text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time}$$

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time}$$

Principles of Computer Design

- Different instruction types having different CPIs

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

$$\text{CPU time} = \left(\sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$

Measure of Performance

$$\text{Processor Performance} = \frac{\text{Time}}{\text{Program}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycle}}$$

(code size) (CPI) (cycle time)

Architecture --> Implementation --> Realization

Compiler Designer **Processor Designer** **Chip Designer**

Our Goal

- Minimize time which is the product, NOT isolated terms
- Common error to miss terms while devising optimizations
  - E.g. ISA change to decrease instruction count
  - BUT leads to CPU organization which makes clock slower
- Bottom line: terms are inter-related

ISA = instruction set architecture

Other Metrics

- MIPS and MFLOPS
- MIPS = instruction count/(execution time x 10^6)

  = clock rate/(CPI x 10^6)
- But MIPS has serious shortcomings

Problems with MIPS

- E.g. without FP hardware, an FP op may take 50 single-cycle instructions
- With FP hardware, only one 2-cycle instruction

  - Thus, adding FP hardware:
    - Total execution time decreases
  - BUT, MIPS gets worse!

Problems with MIPS

- It ignores program
- When is MIPS ok?
  - Same compiler, same ISA
  - E.g. same binary running on Pentium-III, IV
  - Why? Instr/program is constant and can be ignored

Other Metrics

- MFLOPS = FP ops in program/(execution time x 10^6)
- Assuming FP ops independent of compiler and ISA
  - Often safe for numeric codes: matrix size determines # of FP ops/program
  - However, not always safe:
    - Missing instructions (e.g. FP divide, sqrt/sin/cos)
    - Optimizing compilers

Solved Question

- Suppose that when Program A is run, the user CPU time is 3 seconds, the elapsed wallclock time is 4 seconds, and the system performance is 10 MFLOP/sec.
- Assume that there are no other processes taking any significant amount of time, and the computer is either doing calculations in the CPU, or doing I/O, but it can't do both at the same time.
- We now replace the processor with one that runs six times faster, but doesn't affect the I/O speed. What will the user CPU time, the wallclock time, and the MFLOP/sec performance be now?

Solution

- CPU $perf_B$/CPU $perf_A$ = CPU $time_A$/CPU $time_B$
- 6 = 3/CPU $time_B$
- User CPU Time = .5 seconds
- I/O time is unaffected by the performance increase, it still takes 1 second to do I/O.
- => it takes 1 + .5 = 1.5 seconds to run Program A on the faster CPU => Wallclock Time = 1.5 seconds
- System Performance in MFLOPS = Number of Floating Point Operations * 10^6/Wallclock Time
- Old System Performance (10) = #FLOP * 10^6/4
- #FLOP = 40 * 10^6
- New System Performance = 40 * 10^6/1.5
- MFLOP/sec = 26.667

Example

- Machine A: clock 1ns, CPI 2.0, for program x
- Machine B: clock 2ns, CPI 1.2, for program x
- Which is faster and how much?

  Time/Program = instr/program x cycles/instr x sec/cycle

  Time(A) = N x 2.0 x 1 = 2N

  Time(B) = N x 1.2 x 2 = 2.4N

  Compare: Time(B)/Time(A) = 2.4N/2N = 1.2

- So, Machine A is 20% faster than Machine B for this program

Solved question

Instruction Type	Fraction	Cycles
Loads & Stores	30%	6 cycles
Arithmetic Instructions	50%	4 cycles
All Others	20%	3 cycles

- Consider a 200 MHz processor. The split of instructions for an application is given above. Compute its CPI.
- If we say that there are 100 instructions, then:
- 30 of them will be loads and stores.
- 50 of them will be arithmetic instructions.
- 20 of them will be all others.
- (30 * 6) + (50 * 4) + (20 * 3) = 440 cycles/100 instructions
- Therefore, there are 4.4 Cycles per instruction.

Solved question (continued)

- Given: CPU execution time on the application is 11 seconds. What is the ``native MIPS'' processor speed for the application in millions of instructions per second?

The formula for calculating MIPS is:
MIPS = Clock rate/(CPI * 10^6)
The clock rate is 200MHz so...
MIPS = (200 * 10^6)/(4.4 * 10^6) = 45.454545

Solved question (continued)

- If the cycle time must be increased by 20%, find new clock speed (in MHz)?
- Clock time = 1/Cycle Time
- Cycle Time = $1/(200 * 10^6) = 5 * 10^{-9}$
- The cycle time is then increased by 20%:
- $(5 * 10^{-9}) * 1.2 = 6 * 10^{-9}$
- New clock rate: $1/(6 * 10^{-9}) = 166.667 * 10^6$ or 166.667 MHz
- Note that 20% increase in cycle time did not bring 20% decrease in clock speed

Solved question (continued)

- Assume a new compiler generates code that requires only half the number of Loads & Stores. Whats new CPI?
- There were 100 instructions, so we will reduce the number of loads and stores by half, and this will reduce the total number of instructions. So the new instruction mix will be:
- 15 Loads and Stores
- 50 Arithmetic Instructions
- 20 All Others
- The total number of instructions is now 85, so the answer is:
- ((15 * 6) + (50 * 4) + (20 * 3)) / 85 = 350 cycles/ 85 instructions = 4.12 CPI

Solved question (continued)

- If both above optimizations were to applied, how many CPU seconds will the program take?
- CPU seconds = (Number of instructions * Number of Clocks per instructions)/Clock Rate
- First thing we need to do, is calculate the number of instructions which execute in 11 seconds on the new application - the one with half the number of loads and stores.
- To do this, we will need to figure out how many instructions execute on the original application in 11 seconds. Since we know the MIPS or how many *Millions of Instructions Per Second* for the original application, we say:
- $(45.45 * 10^6) * 11 = 500 * 10^6$ instructions in 11 seconds

Solved question (continued)

- Now we need to figure out how many of those are Loads and Stores so:
- $(500 * 10^6) * .3 = 150 * 10^6$ are Load and Store instructions in original app because the chart says that 30% of all instructions are Loads and Stores.
- Now we need to cut this number in half, because the new benchmark says that we have half the number of loads and stores , but the cycle time increases by 20%. Therefore there are only $75 * 10^6$ loads and stores. This also means that there are now less total instructions, $425 * 10^6$ total instructions.
- The final solution is:
- $((425 * 10^6) * 4.12)/(166.667 * 10^6) = 10.548$ seconds