CS3210
Parallel Computing

Changes from Monday in Green

Tut 2
Mon (4pm)
Tues (2pm)

Admin

Updates

- Assignment 1 part 1 due yesterday morning, 11am
  - Late penalty: 10% per day, up to a week
  - If you have submitted, download your submission and check its contents are correct

- Lab 1 comments released in LumiNUS gradebook
  - Full credit awarded if you followed submission instructions
  - For clarifications: email me or drop me a message

- End-tutorial Quiz 2 later

Q1

Task Parallelism

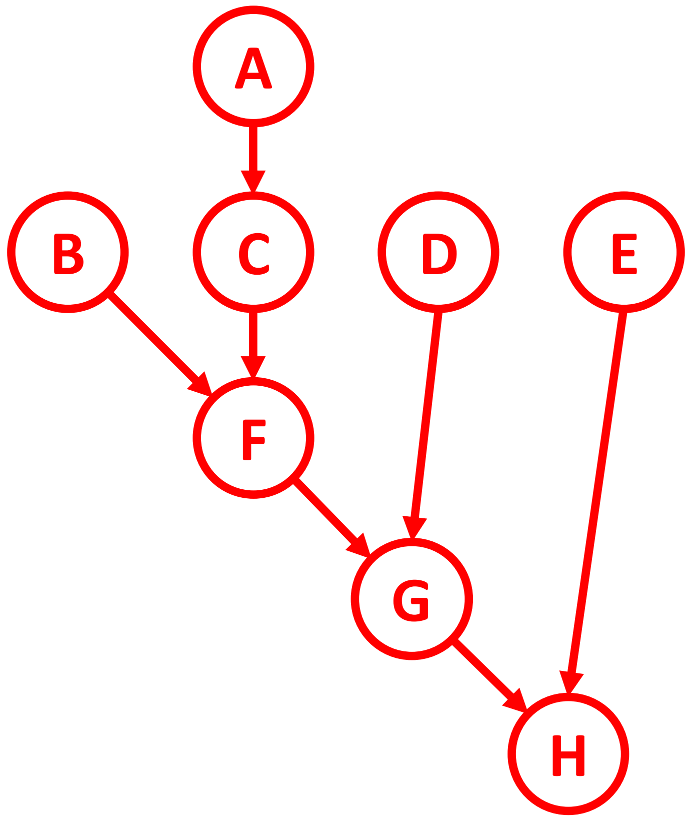
- Parbegin-parend pattern
  - Demarcates a parallel region in code
  - A set of threads are spawned at **`parbegin`**
  - Threads execute subsequent statements until **`parend`**, where they are joined/destroyed
  - Statements only executed in parallel with another statement if specified with **`parallel`** keyword, e.g. **`A parallel B`**
  - A sub-region is demarcated with **`do (…) end`**

```
parbegin
  do
    parbegin
      do
        parbegin
          do
            A
            C
          end
          parallel
          B
        parend
        F
      end
      parallel
      D
    parend
    G
  end
  parallel
  E
parend
H
```

Q1(a)

Task Parallelism

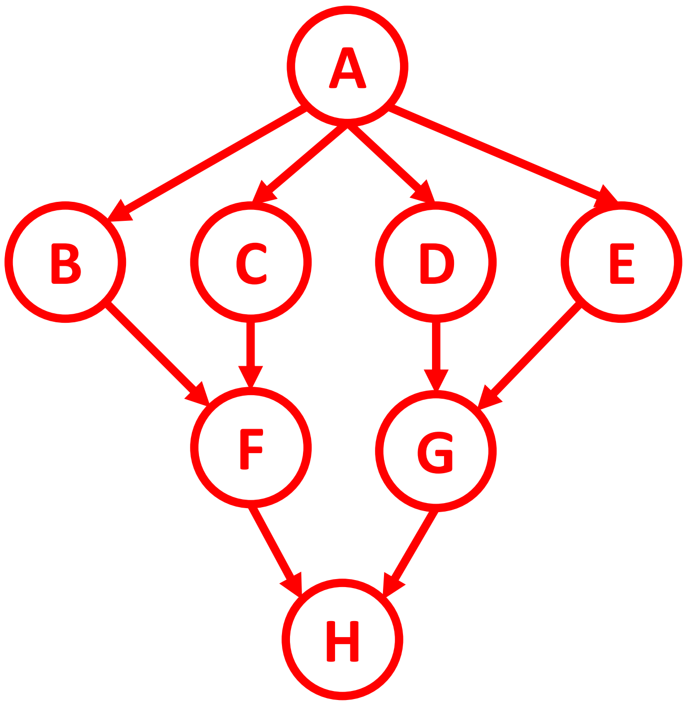
- Sketch out the task dependence graph for code fragment 1



```
A
parbegin
  do
    parbegin
      B
      parallel
      C
    parend
    F
  end
  parallel
  do
    parbegin
      D
      parallel
      E
    parend
    G
  end
parend
H
```

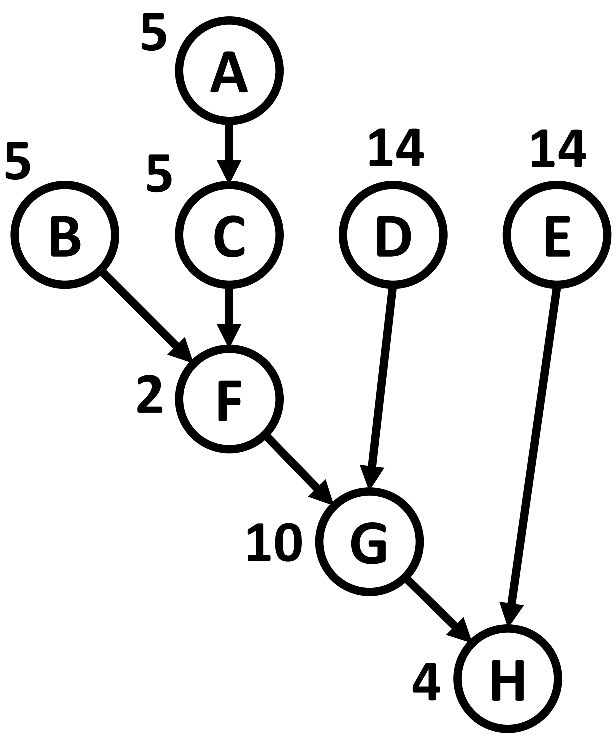
Q1(a)
Task Parallelism

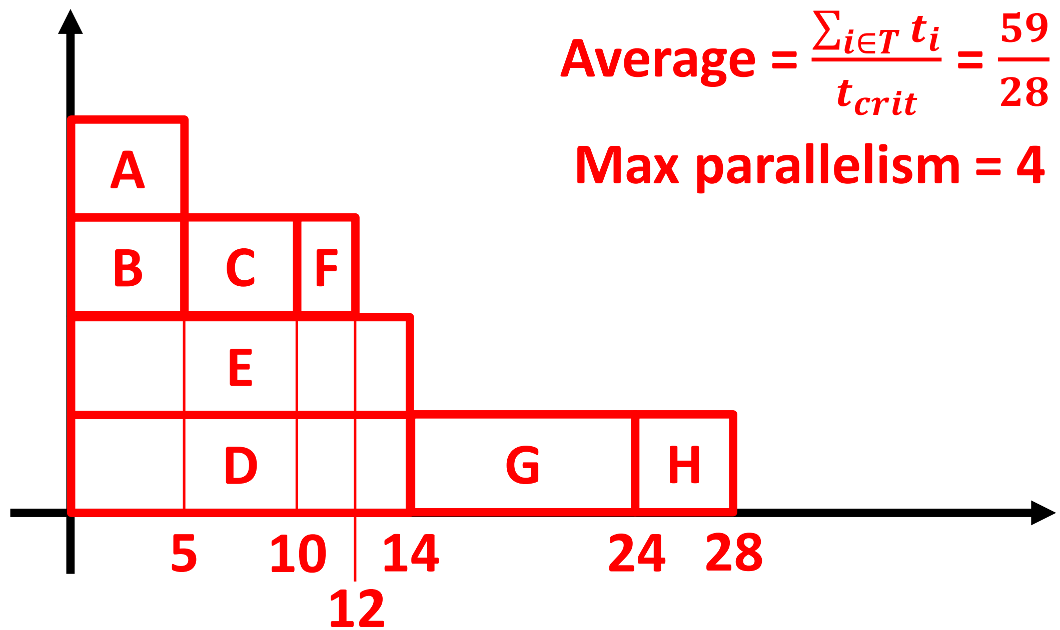
- Sketch out the task dependence graph for code fragment 2



Q1(b)
Task Parallelism

- Plot the degree of task parallelism over time for fragment 1, and derive its average and max parallelism





Critical path t_{crit} = 28

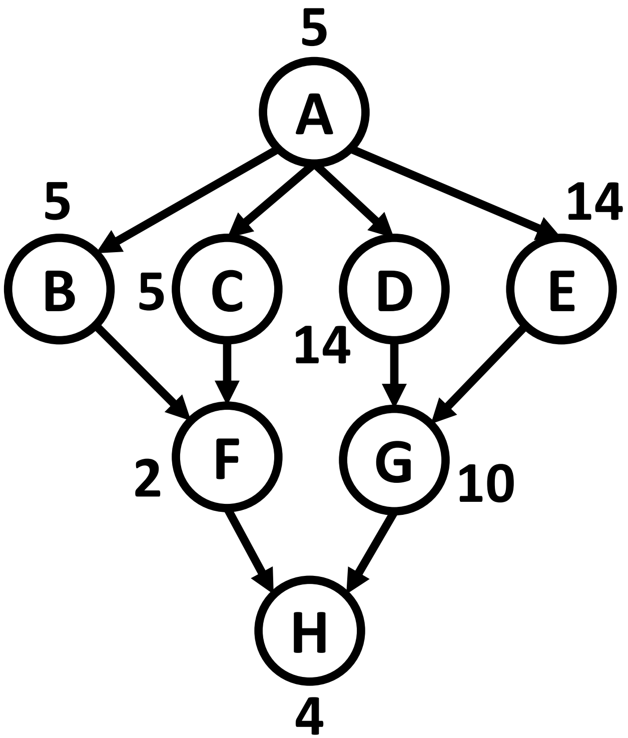
Average = $\frac{\sum_{i \in T} t_i}{t_{crit}} = \frac{59}{28}$

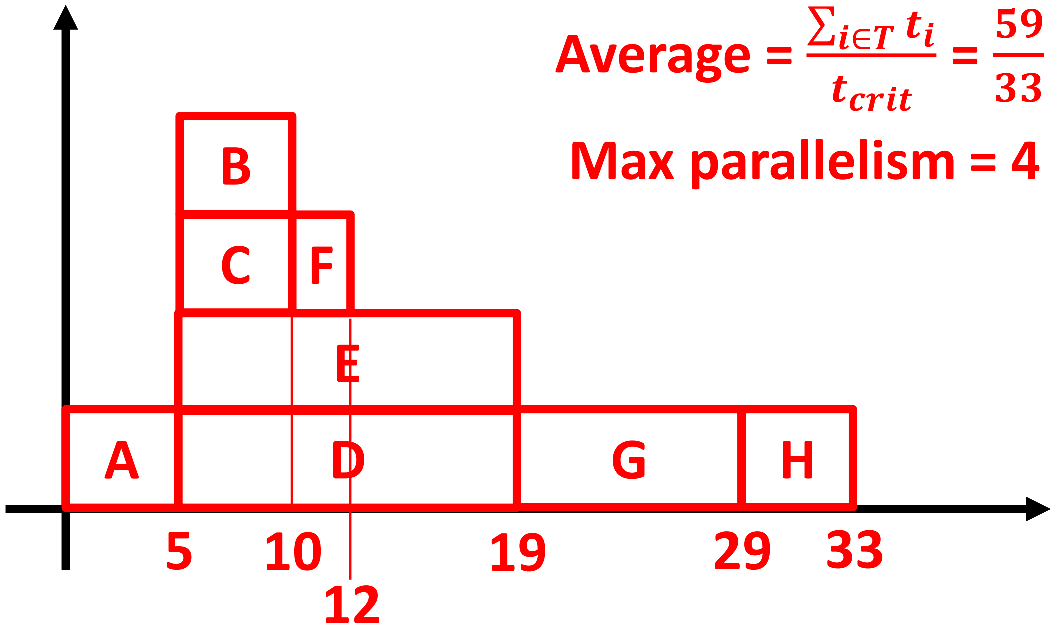
Max parallelism = 4

Q1(b)

Task Parallelism

- Plot the degree of task parallelism over time for fragment 2, and derive its average and max parallelism





Critical path t_{crit} = 33

Average = $\frac{\sum_{i \in T} t_i}{t_{crit}} = \frac{59}{33}$

Max parallelism = 4

Q1(c)

Task Parallelism

- What is the maximum achievable speedup for both fragments 1 and 2, given *infinite resources*?
  - The execution at each point cannot utilize more resources than the number of parallel tasks
  - Therefore, this is equal to average degree of task parallelism

Code fragment 1

Critical path t_{crit} = 28

Average = $\frac{\sum_{i \in T} t_i}{t_{crit}} = \frac{59}{28}$

Max parallelism = 4

Code fragment 2

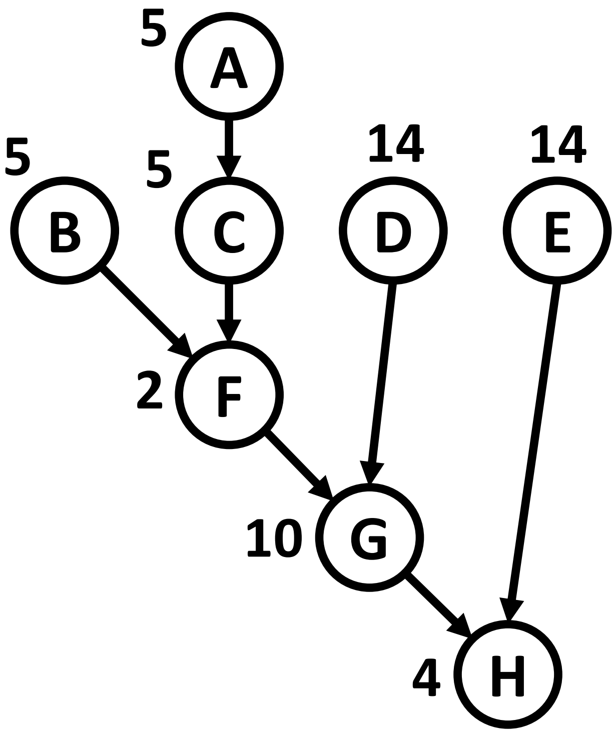
Critical path t_{crit} = 33

Average = $\frac{\sum_{i \in T} t_i}{t_{crit}} = \frac{59}{33}$

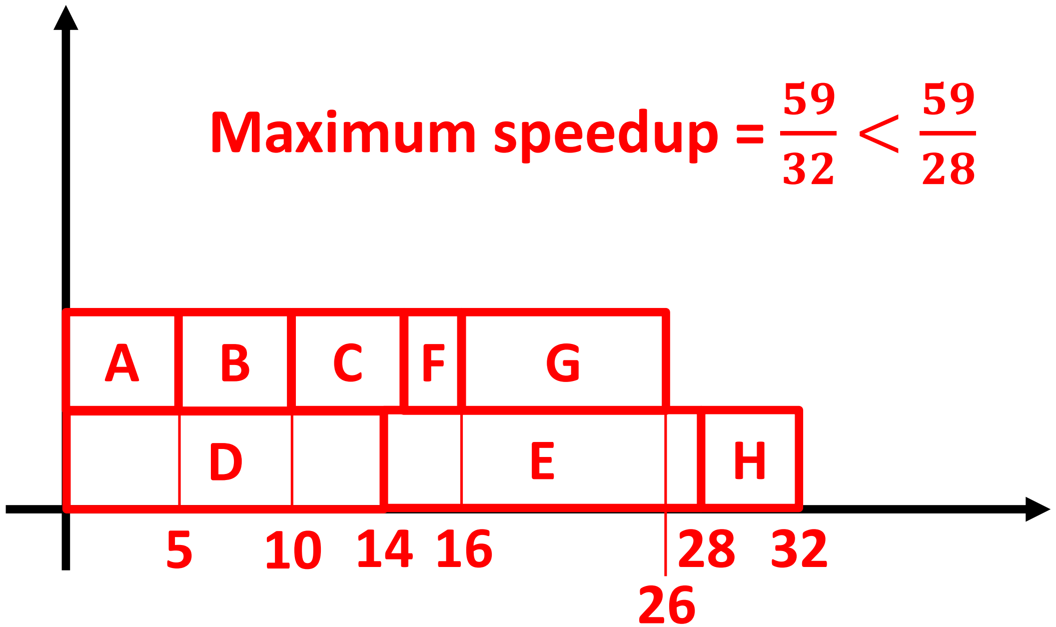
Max parallelism = 4

Q1(d)
Task Parallelism

- If we only have 2 processing elements, what then is the maximum achievable speedup for fragment 1?



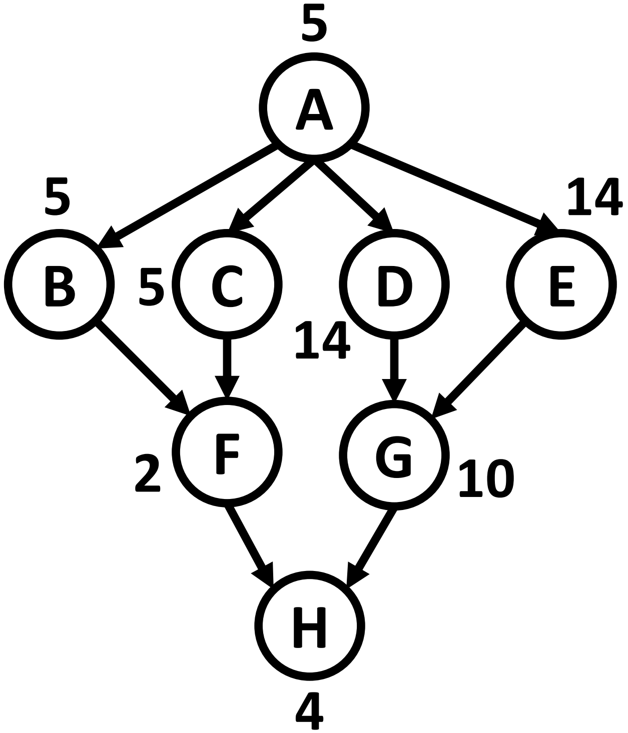
Maximum speedup = $\frac{59}{32} < \frac{59}{28}$



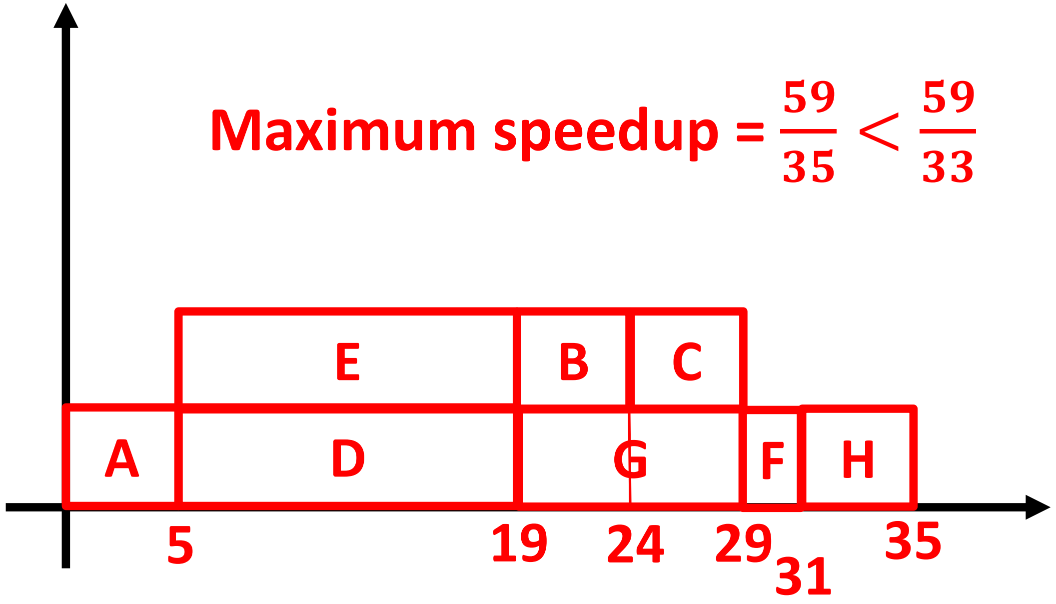
Q1(d)

Task Parallelism

- If we only have 2 processing elements, what then is the maximum achievable speedup for fragment 2?

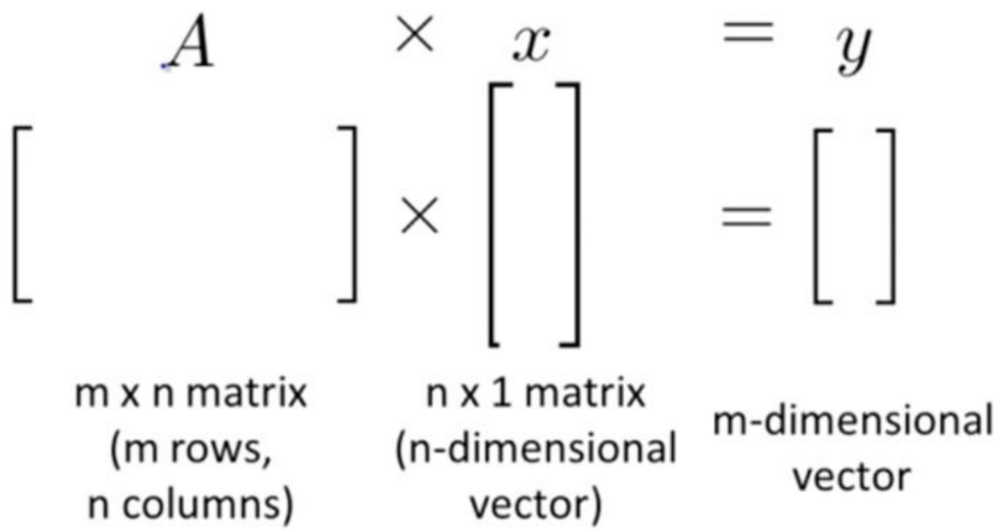


Maximum speedup $= \frac{59}{35} < \frac{59}{33}$



Q2(a)
Parallel Programming Models

- Consider the matrix-vector multiplication problem
  - What type of parallelism does this problem entail (data or task)?
  - Data parallelism



Q2(b)

Parallel Programming Models

- What type of architecture from Flynn's taxonomy would you choose to solve it with? Explain.
  - SIMD (single-instruction multiple-data)

- What parallel programming pattern would you employ to solve the matrix-vector multiplication problem?
  - Patterns: fork-join, parbegin-parend, SPMD/SIMD, master-worker, client-server, pipelining, task pool, producer-consumer
  - Master-worker pattern/SIMD

Q3

Performance of Parallel Systems

- Consider two processors P1 and P2 with the same ISA
  - P1 has a clock rate of 6 GHz, P2 has a clock rate of 2 GHz
  - Three types of instructions, with different CPI (cycles per instruction) for each on each processor
  - Three compilers C1, C2, C3 that generate programs with same total instructions (but different proportions)

Instructn Type	CPI (P1)	CPI (P2)	C1	C2	C3
FP arithmetic	4	2	30%	30%	50%
Int ADD/SUB	6	4	50%	20%	20%
Int MUL	8	3	20%	50%	30%

Q3(a)
Performance of Parallel Systems

- If C1 is used for both processors, how much faster is P1 than P2?

Instructn Type	CPI (P1)	CPI (P2)	C1	C2	C3
FP arithmetic	4	2	30%	30%	50%
Int ADD/SUB	6	4	50%	20%	20%
Int MUL	8	3	20%	50%	30%

- Average CPI $\overline{CPI} = \sum_{i=1}^{n} P_i \times CPI_i$
- Time taken $T = I \times \overline{CPI} \times T_{cyc} = \frac{N_{cyc}}{f}$
- % difference = $\frac{T_2 - T_1}{T_2} = 39.6\%$

Q3(b)

Performance of Parallel Systems

- If C2 is used for both processors, how much faster is P1 than P2?

Instructn Type	CPI (P1)	CPI (P2)	C1	C2	C3
FP arithmetic	4	2	30%	30%	50%
Int ADD/SUB	6	4	50%	20%	20%
Int MUL	8	3	20%	50%	30%

- Average CPI $\overline{CPI} = \sum_{i=1}^{n} P_i \times CPI_i$
- Time taken $T = I \times \overline{CPI} \times T_{cyc} = \frac{N_{cyc}}{f}$
- % difference = $\frac{T_2 - T_1}{T_2} = 26.4\%$

Q3(c,d)

Performance of Parallel Systems

- Which of the three compilers is best for each of P1 and P2? **(Hint: do we need to consider the clock frequency?)**

Instructn Type	CPI (P1)	CPI (P2)	C1	C2	C3
FP arithmetic	4	2	30%	30%	50%
Int ADD/SUB	6	4	50%	20%	20%
Int MUL	8	3	20%	50%	30%

Compiler	Weighted CPI (P1)	Weighted CPI (P2)
C1		
C2		
C3		

Q4(a)

Parallel Performance Scaling

- Consider a program that for problem size N has a:
  - Sequential portion: with N instructions
  - Parallel portion: with N^2 instructions

- Suppose $N = 100$; what is the speedup with 10 and 100 processors respectively? What is the upper limit?
  - Use Amdahl's law for $\boldsymbol{p}$ processors and sequential fraction $\boldsymbol{f}$

$$\boldsymbol{S_p(N) = \frac{1}{f + \frac{1-f}{p}}}$$

Q4(a)

Parallel Performance Scaling

- Suppose $N = 100$; what is the speedup with 10 and 100 processors respectively? What is the upper limit?

$$f = \frac{I_{seq}}{I_{total}} = \frac{N}{N + N^2} = \frac{1}{1 + N} = \frac{1}{101}$$

$$S_{10}(100) = \frac{1}{\frac{1}{101} + \frac{100}{101(10)}} = 9.181$$

$$S_{100}(100) = \frac{1}{\frac{1}{101} + \frac{100}{101(100)}} = 50.5$$

$$S_{\infty}(100) = \frac{1}{\frac{1}{101} + 0} = \frac{1}{f} = 101$$

Q4(b)

Parallel Performance Scaling

- Assume each instruction can be executed in 2 cycles on a 1 GHz processor
  - ➢ Calculate the time needed to run the program with $N = 100$ on a single core

$$I_{N=100} = N + N^2 = 100(101) = 10100$$

$$t_{N=100} = \frac{2 \times I_{N=100}}{f} = \frac{20200}{10^9} = 0.0000202s$$

Q4(b)

Parallel Performance Scaling

- Given the same amount of time, what is the problem size we can solve with 10 and 100 processors?

$$0.0000202 = t(N,p) = \boxed{\left(N + \frac{N^2}{p}\right)} \times \frac{2}{10^9}$$

CPI

f

I_{worst} **(on processor that executes the sequential part)**

$$\frac{N^2}{p \times 10^9} + \frac{N}{10^9} - 0.0000101 = 0$$

$$p = 10 \Rightarrow N \approx 312$$

$$p = 100 \Rightarrow N \approx 956$$

➢ What is the speedup for these problem sizes and processor counts? ☺

Admin

Tutorial Quiz

- Go to Luminus > Quiz
  - 1% of your grade
  - 8 minutes for 4 MCQ questions
  - **Don't be stressed - full credit for $\geq$ 60% correct (you should be able to do this after the tutorial, well, hopefully)**

- 10 minutes now to attempt it

CS3210

Parallel Computing

Thank you! Any questions?

Tut 2

Mon (4pm)
Tues (2pm)