Parallel Computing: Basic Concepts

Tuesday, August 24, 2021 11:02 AM

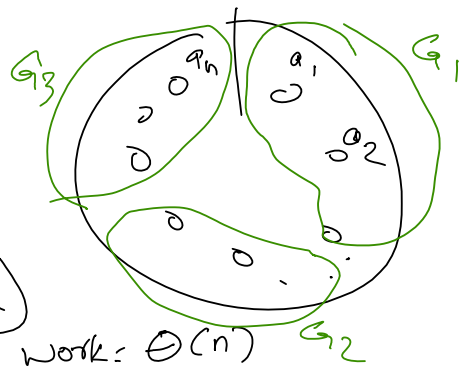
"work" ≡ volume of computation to perform a task

"time" ≡ wall clock time to perform a task

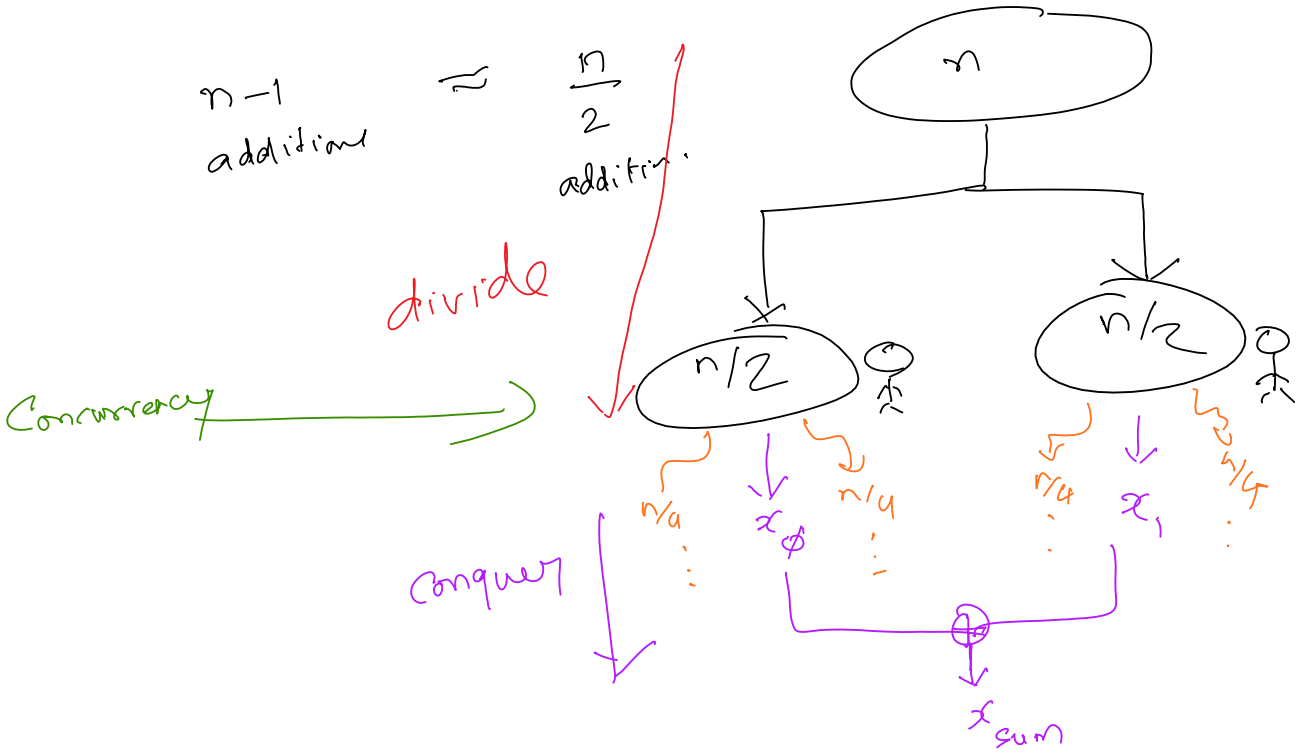
time ∝ work

Groups ≈ k

time ∝ $\frac{\Theta(n)}{k}$ (ideal)



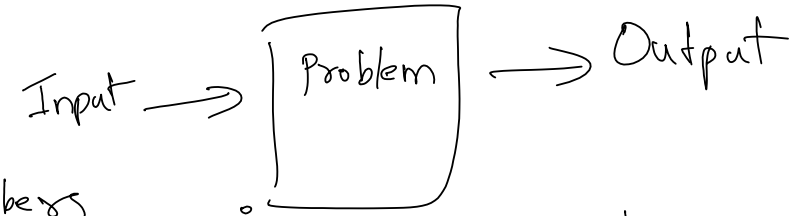
$n-1$ additions $\approx$ $\frac{n}{2}$ additions



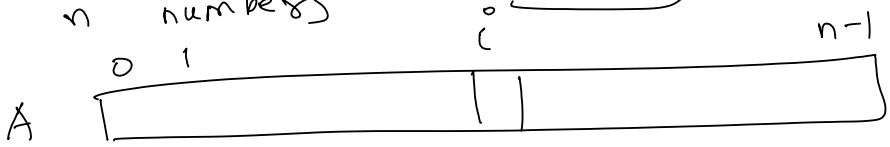
Parallel Sum

Tuesday, August 24, 2021 11:03 AM

Sum: "Problem"



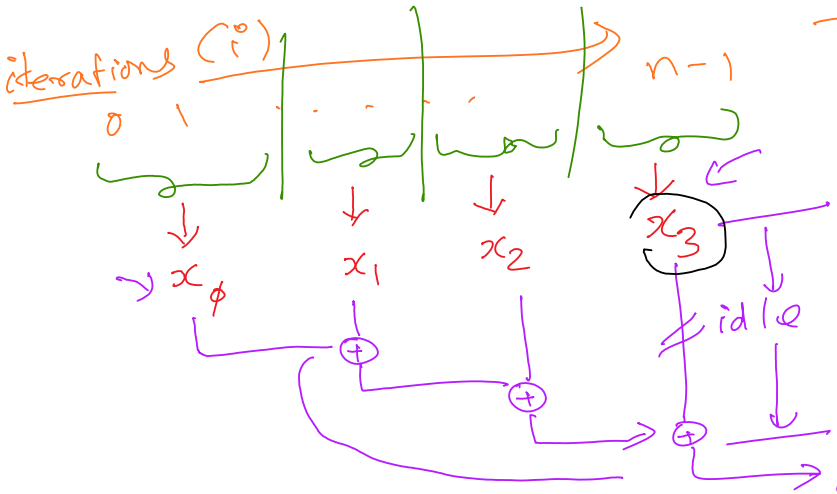
Input: n numbers



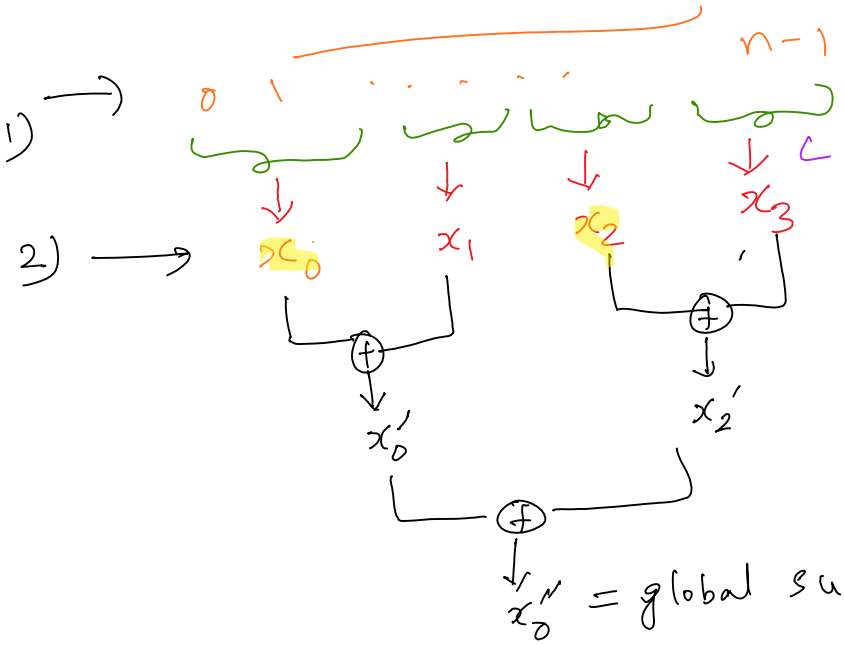
Output: Global sum = $\sum_{i=0}^{n-1} A[i]$

Serial code

```
gSum = 0
for(i=0; i<n; i++)
    gSum += A[i];
return gSum;
```




```
for(j=0;
    j<4;
    j++)
gSum += x_j
```



n = 4

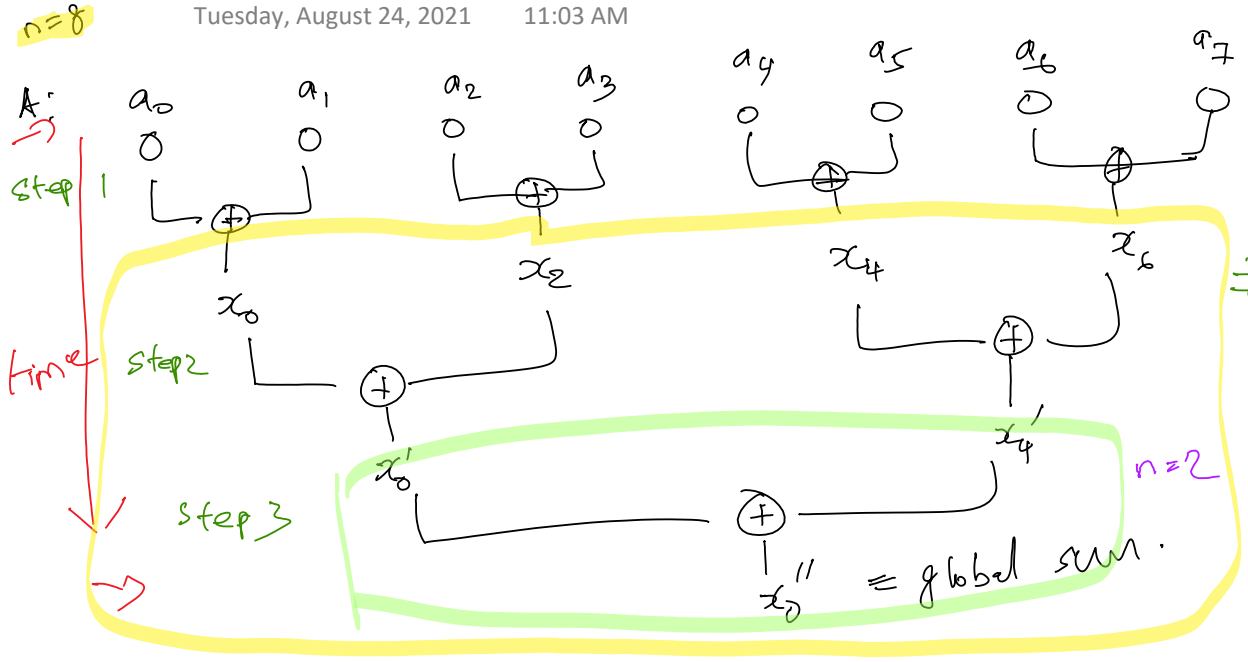
Parallel algorithm: work ≡ n−1 additions

Serial baseline = n−1 additions = work

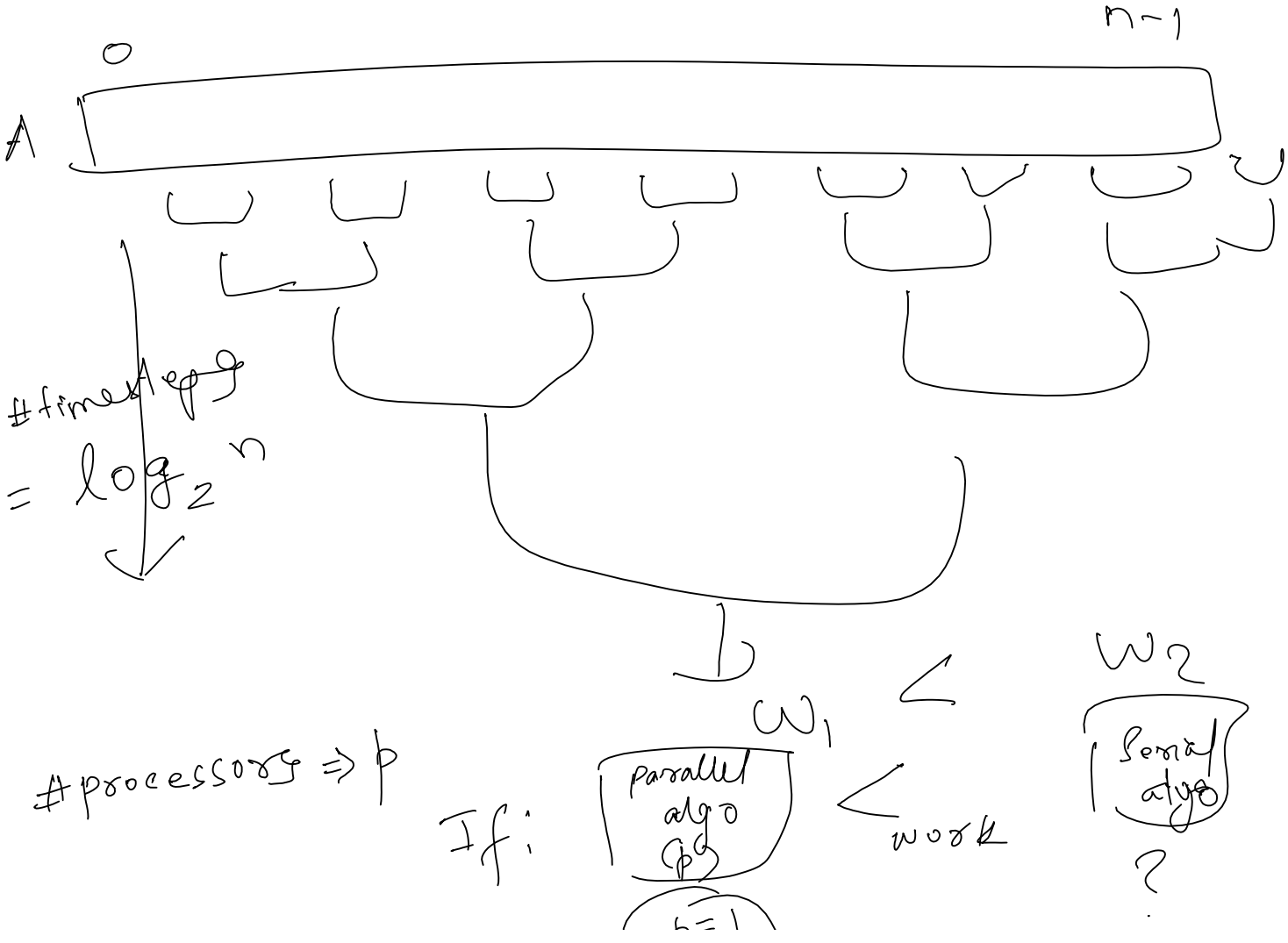
n = 8

Tuesday, August 24, 2021 11:03 AM

p = 8



timesteps = 3 << 7

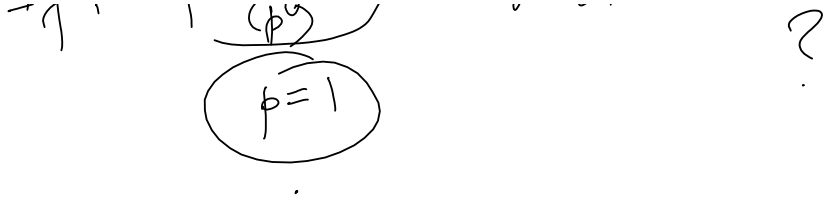


processors ⇒ p

If: W_1 (parallel algo on p) < work

W_2 (serial algo) ?

p = 1


ρ=1

Parallel Performance

Tuesday, August 24, 2021 11:03 AM

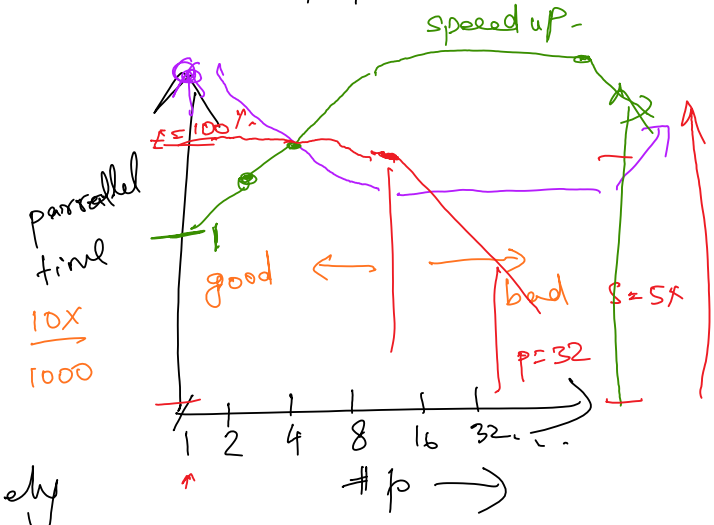
Parallel Performance:

work / time — serial (aka. "sequential") / parallel — } wall clock time (start ↓ end)

Speedup

$S(n, p) = \frac{\text{serial time}}{\text{parallel time on } p \text{ procs.}}$

n: input size; p: #processes



Efficiency (E):

$E(n, p) = \frac{S(n, p)}{p}$

10x / 10 — 10x / 1000

≈ % processes effectively utilized to complete a task.

E = 10% on (p = 1000)

100 procs were doing useful work

900 procs are not

$n >> p$

Speedup example

Tuesday, August 24, 2021 11:03 AM

Ex: Sum of n numbers ($n = p$)

Serial time $T_s(n) = \Theta(n)$

parallel time

└ naive approach: $T_p^{naive}(n,p) = \Theta(n)$

Speedup $S(n,p) = \frac{\Theta(n)}{\Theta(n)} \approx 1$

Eff: $E(n,p) = \frac{\Theta(1)}{p} \approx 1/p$

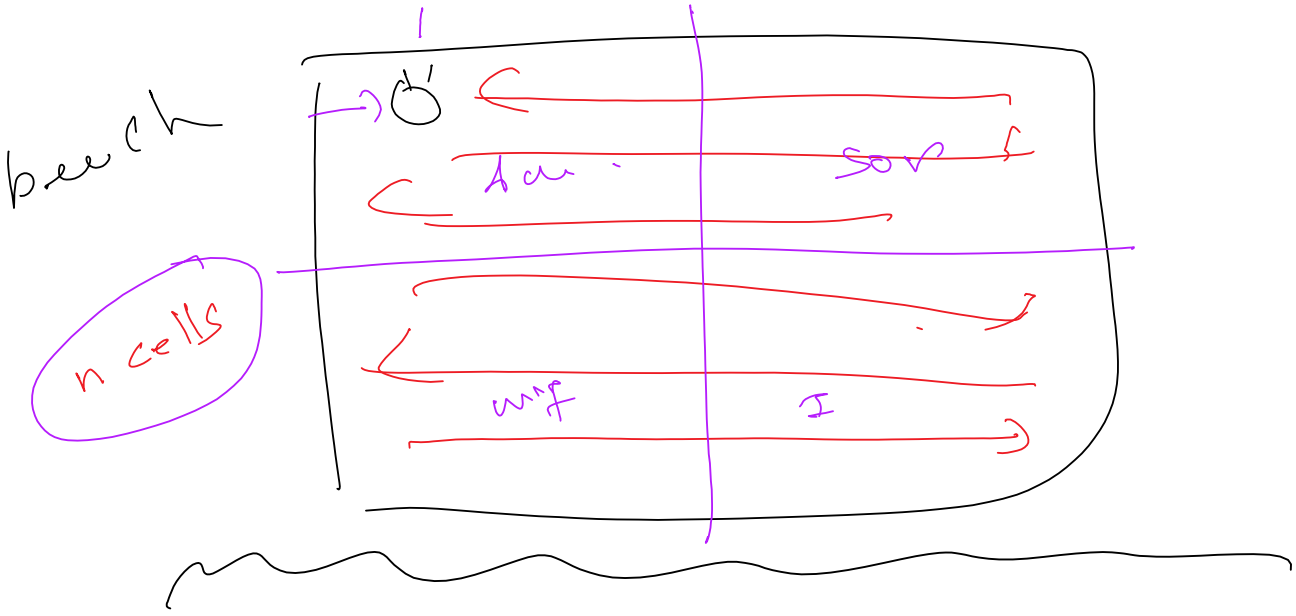
" " Tree approach: $T_p^{tree}(n,p) = \Theta(\log_2 n)$

Speedup $S(n,p) = \frac{\Theta(n)}{\Theta(\log_2 n)}$

Eff $E(n,p) = \frac{n}{(\lg n)\, n} \approx \frac{1}{\lg n}$

Superlinear speedup example

Tuesday, August 24, 2021 11:03 AM



Efficiency

Thursday, August 26, 2021 12:47 PM

Time t ≈ time

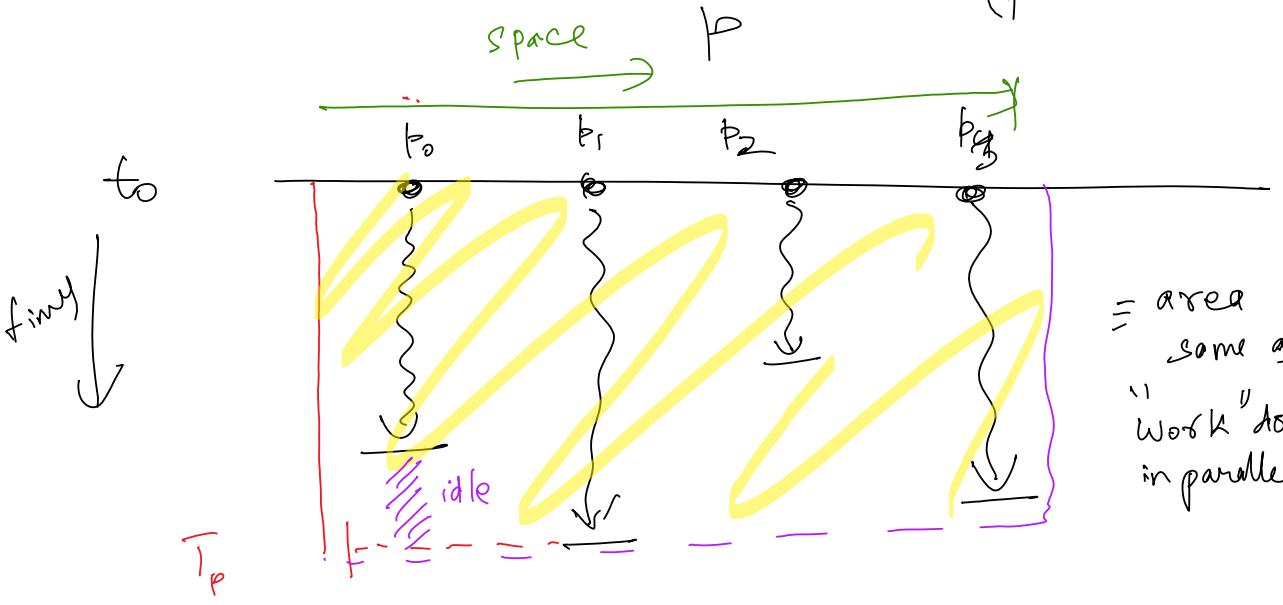
Speedup ≈ velocity

Fff. ≈ acceleration

Effieiency $E = \frac{S(n,p)}{p}$

$= \frac{\left(\frac{\text{serial time}}{\text{parallel time}}\right)}{p} = \frac{\text{serial time}}{(\text{parallel time}) \times p} = \frac{\text{serial work}}{\text{parallel work}}$

$= \frac{\text{serial work}}{\text{serial work} + \text{extra work}}$ ← overhead



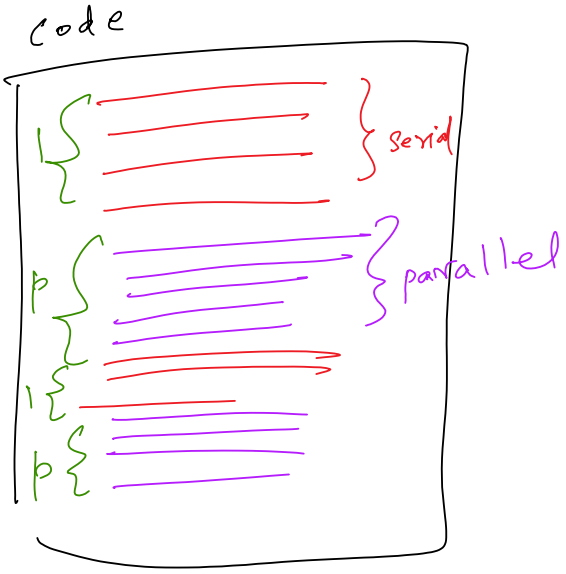
= area is same as "work" done in parallel

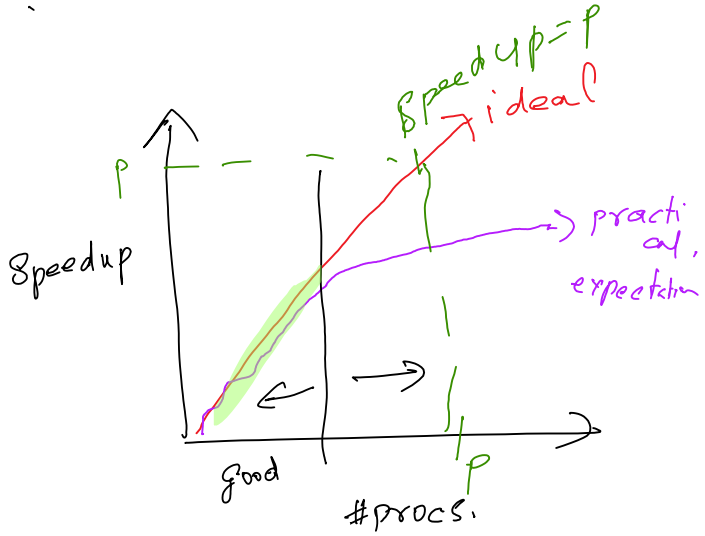
parallel time = max { all running times across procs }

#tasks per minute.

Throughput & Amdahl's Law

Tuesday, August 31, 2021 11:33 AM





Total work = 100%

Let t_s ← fraction of code/work that is serial

t_p ← fraction of the code/work that is parallel

(p procs. in the parallel segments)

$t_s + t_p = 1$

Speedup S = serial time / parallel time

t_s t_p

$$= \frac{1}{t_s + \frac{t_p}{p}} \approx \frac{1}{t_s}$$

bottleneck ←

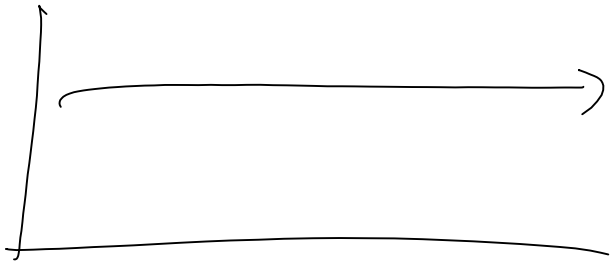
Gustafson's Law

$$S = \frac{1}{t_s + \frac{t_p}{p}}$$

$$\text{Speedup} = \frac{p}{p\,t_s + t_p}$$

#p: 10 procs ↓ 20 procs ⇓

#n work: $n = 10^3$ ↓ $2n$



n work: $n = 10^3$ $2n$

Parallel Computation: Basics

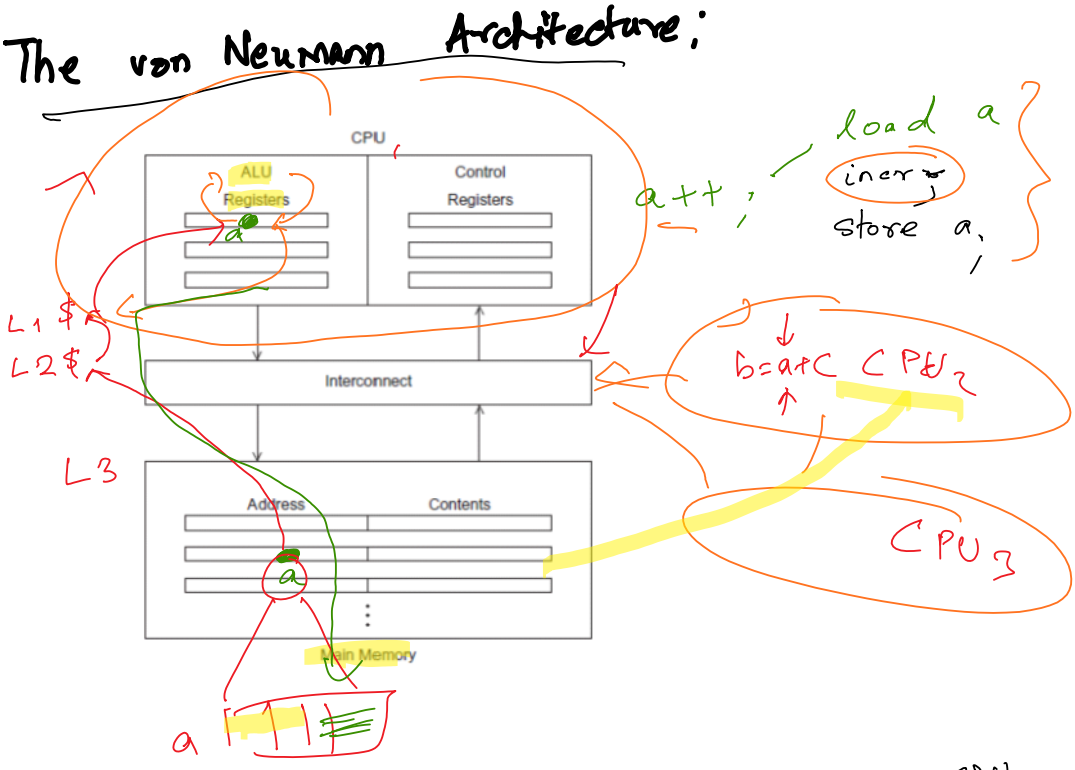
Tuesday, September 1, 2020 11:33 AM

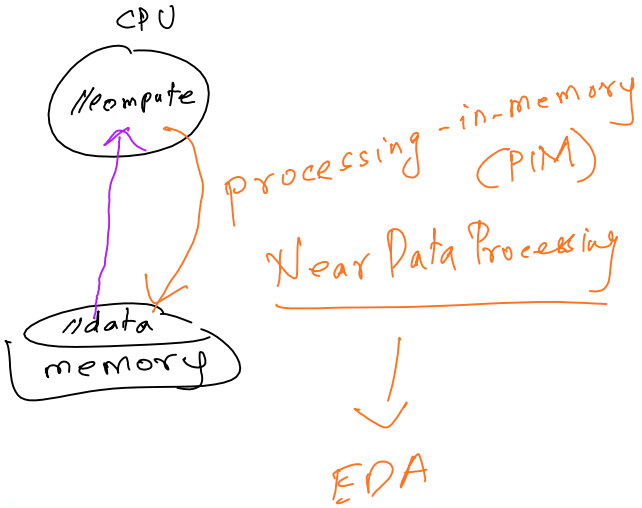
Useful reference:

1. Pacheco book slides on chapter 2 (parallel h/w and parallel s/w)

Topics for today:

1. The von Neumann Architecture
2. Shared vs. Distributed Memory models
3. Parallel computing models
4. Process***es*** vs. Process***ors***
5. Processes vs. Threads
6. Message Passing





Parallel Programming Models

Tuesday, August 31, 2021 11:34 AM

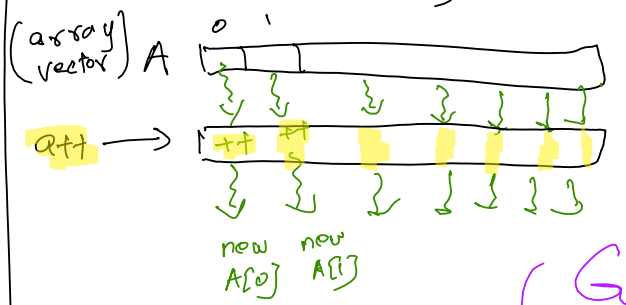
Parallel Computing Models.

Single Instruction Single Data (SISD)

data → compute

(Serial)

Single Instruction Multiple Data (SIMD)



≡ vector processing

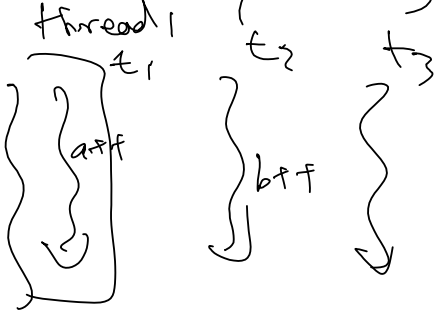
(GPUs Vector Proc)

Multiple Instruction Single Data (MISD)

program

call graph

Multiple Instruction Multiple Data (MIMD)

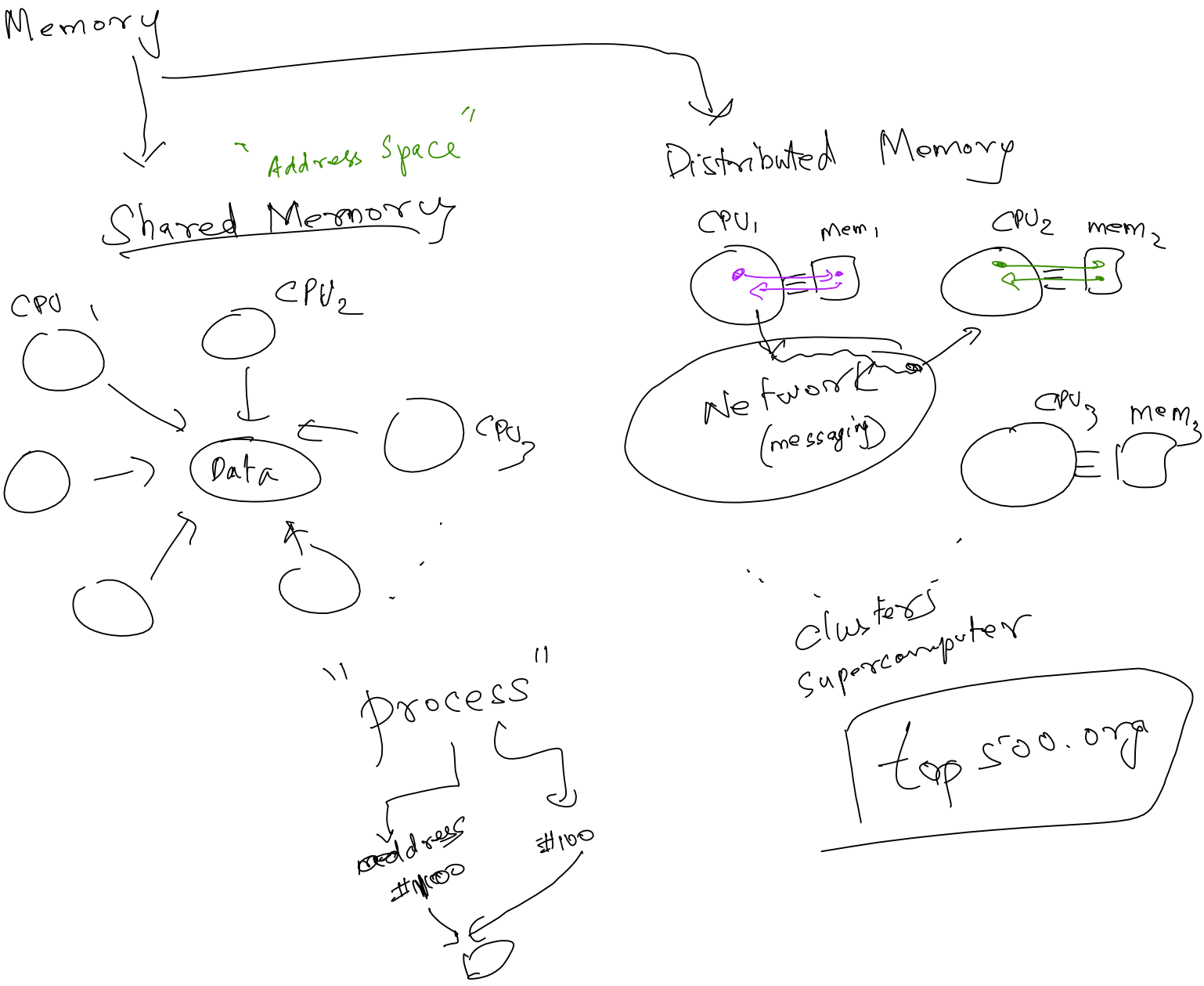


multithreading

Message Passing.

Memory-based models

Tuesday, August 31, 2021 11:34 AM



DRAM

Processes

Tuesday, August 31, 2021 1:11 PM