

COL380

Introduction to
Parallel & Distributed Programming

Parallel Speedup, $S_p = \frac{\text{Exec time using 1 processor system } (t_1)}{\text{Exec time using } p \text{ processors } (t_p)}$

Parallel Efficiency, $\mathcal{E}_p = \frac{S_p}{p}$

Parallel Cost, $C_p = p \times t_p$

Cost Optimal if $C_p = t_1$

Look out for inefficiency:

$$t_1 = n^3$$

$$t_p = n^2, \text{ for } p = n^2$$

$$C_p = n^4$$

Parallelization Overhead

$$\bar{o}_p = p \times t_p - t_1$$

Parameterize with “p”

- Algorithms scale up to a processor count, p
 - A larger value of p raises the expectation that the algorithm scales well

- Just count the number of computations?

- One unit of work done per processor at step i
- (Common clock: steps proceed at the same rate)

$$\text{Work} = \sum_{i=0}^{t(n,p)} p_i$$

#steps

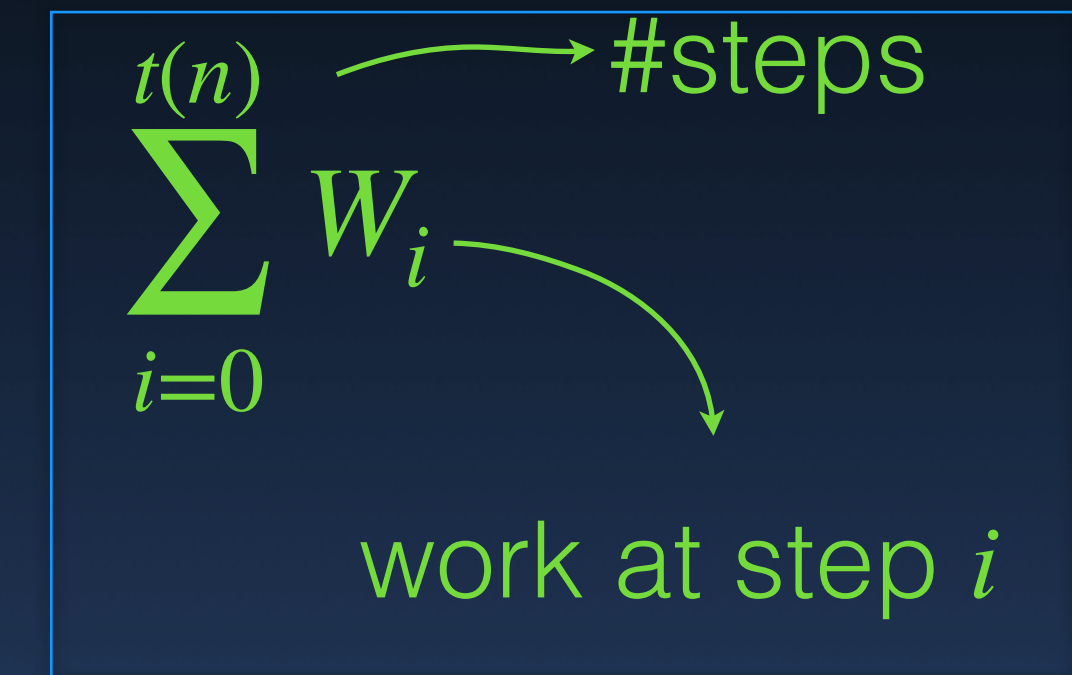
#processors active at step i

- To execute using fewer processors $p^\# < p$, where $p = \max(p_i)$

- Simulate each step (that uses p_i processors) thus:

Work Time Scheduling Principle

- Design algorithm in terms of
 - Total work done per 'time step': $W_i(n)$
 - $t(n)$ steps
- Total work done $W(n) = \sum W_i(n)$
- Given p processors, at step i :
 - divide the work $W_i(n)$ among p processors
 - ▶ Time $\leq \sum \lceil (W_i(n)/p) \rceil \leq \lfloor W(n)/p \rfloor + t(n)$
- Cost = $t(n,p) * p$



- **Work = Cost if:**
 - $p * t(n,p) = O(W(n))$
 - Or, $p = O(W(n)/t(n,p))$

Work \leq Cost:
Cost optimality is more stringent.

- f = fraction of the problem that is sequential
 - ➔ $\Rightarrow (1 - f)$ = fraction that is parallel

- Best parallel time
$$t_{par} = t_{seq} \left(f + \frac{1-f}{p} \right)$$
 - ➔ Fraction $(1-f)$ equally shared by p processors

- Speedup with p processors:
$$S_p = \frac{1}{f + \frac{1-f}{p}}$$

Amdahl's Law

$$S_p = \frac{1}{f + \frac{1-f}{p}} \rightarrow 0$$

- Speed-up due to p processors
- Upper bound on speedup at $p \rightarrow \infty$

$$S_\infty = \frac{1}{f}$$

Amdahl's Law

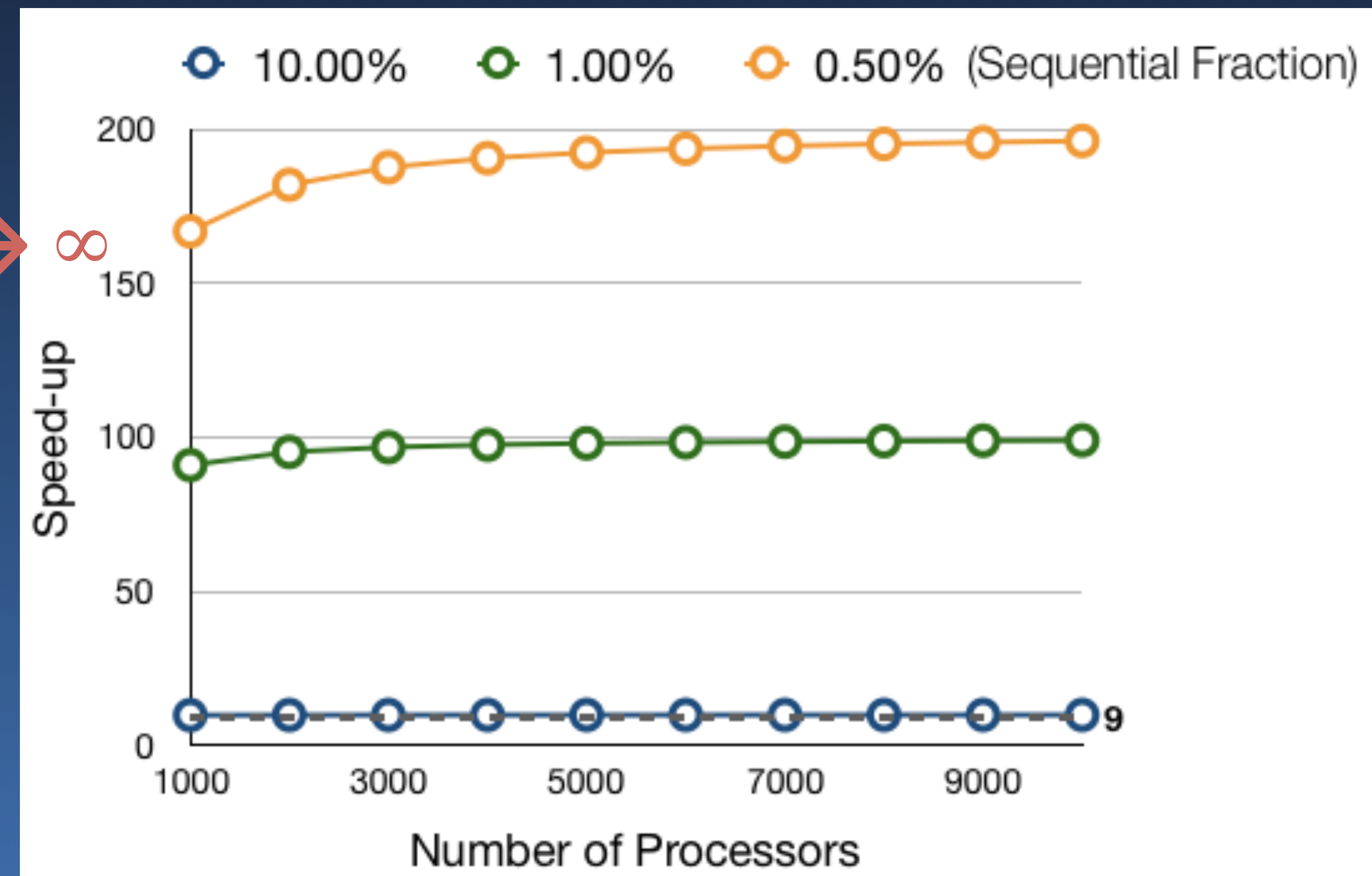
$$S_p = \frac{1}{f + \frac{1-f}{p}} \rightarrow 0$$

- Speed-up due to p processors

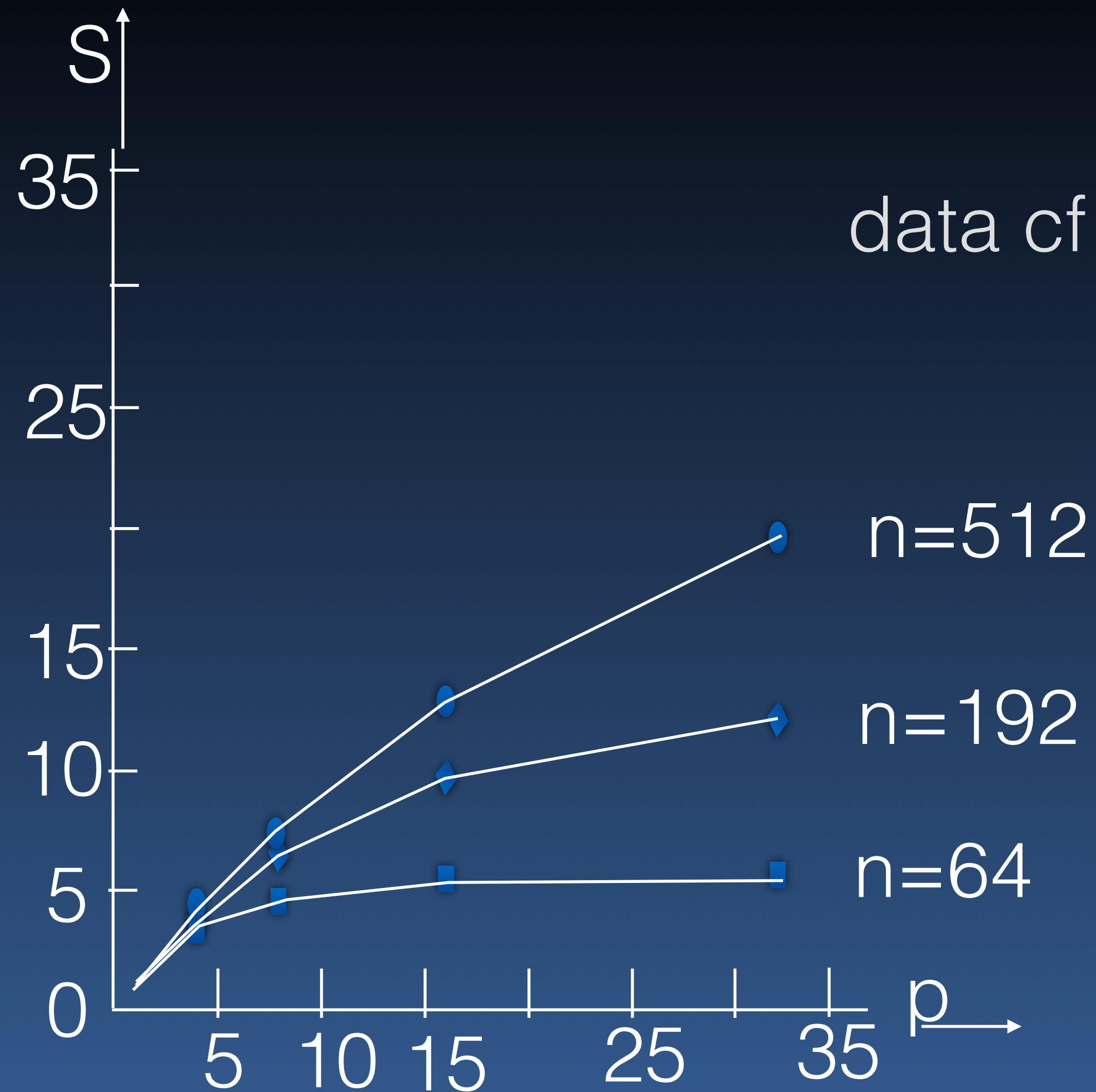
- Upper bound on speedup at $p \rightarrow \infty$

$$S_\infty = \frac{1}{f}$$

$$f = 10\%, S_\infty \rightarrow 1 / 0.1 = 10$$



Example Scaling



Speedup saturates and efficiency drops
(Amdahl's law)

But the limit depends on n:
size of the problem

Speedup versus the number of processing elements for adding a list of numbers

Gustaffson's Law

- Observation: applications seem to exceed Amdahl's speed-up
 - (i.e., assumption are too restrictive)
- As p increases, the opportunity for parallelization can also increase

Time taken for a problem on p processors = $t_{seq} + t_{par} = t(n,p)$

f = fraction of
time spent in
sequential work

Gustaffson's Law

- Observation: applications seem to exceed Amdahl's speed-up
 - (i.e., assumption are too restrictive)
- As p increases, the opportunity for parallelization can also increase

Time taken for a problem on p processors = $\frac{t_{seq}}{p} + t_{par} = t(n,p)$

$f t(n,p)$ $(1-f) t(n,p)$

$$t(n,1) = f t(n,p) + p (1-f) t(n,p)$$

f = fraction of time spent in sequential work

- Observation: applications seem to exceed Amdahl's speed-up
 - (i.e., assumption are too restrictive)
- As p increases, the opportunity for parallelization can also increase

Time taken for a problem on p processors = $\frac{t_{seq}}{p} + t_{par} = t(n,p)$

$f t(n,p)$ $(1-f) t(n,p)$

$$t(n,1) = f t(n,p) + p (1-f) t(n,p)$$

f = fraction of time spent in sequential work

$$\Rightarrow S_p = \frac{t(n,1)}{t(n,p)} = f + p (1-f)$$

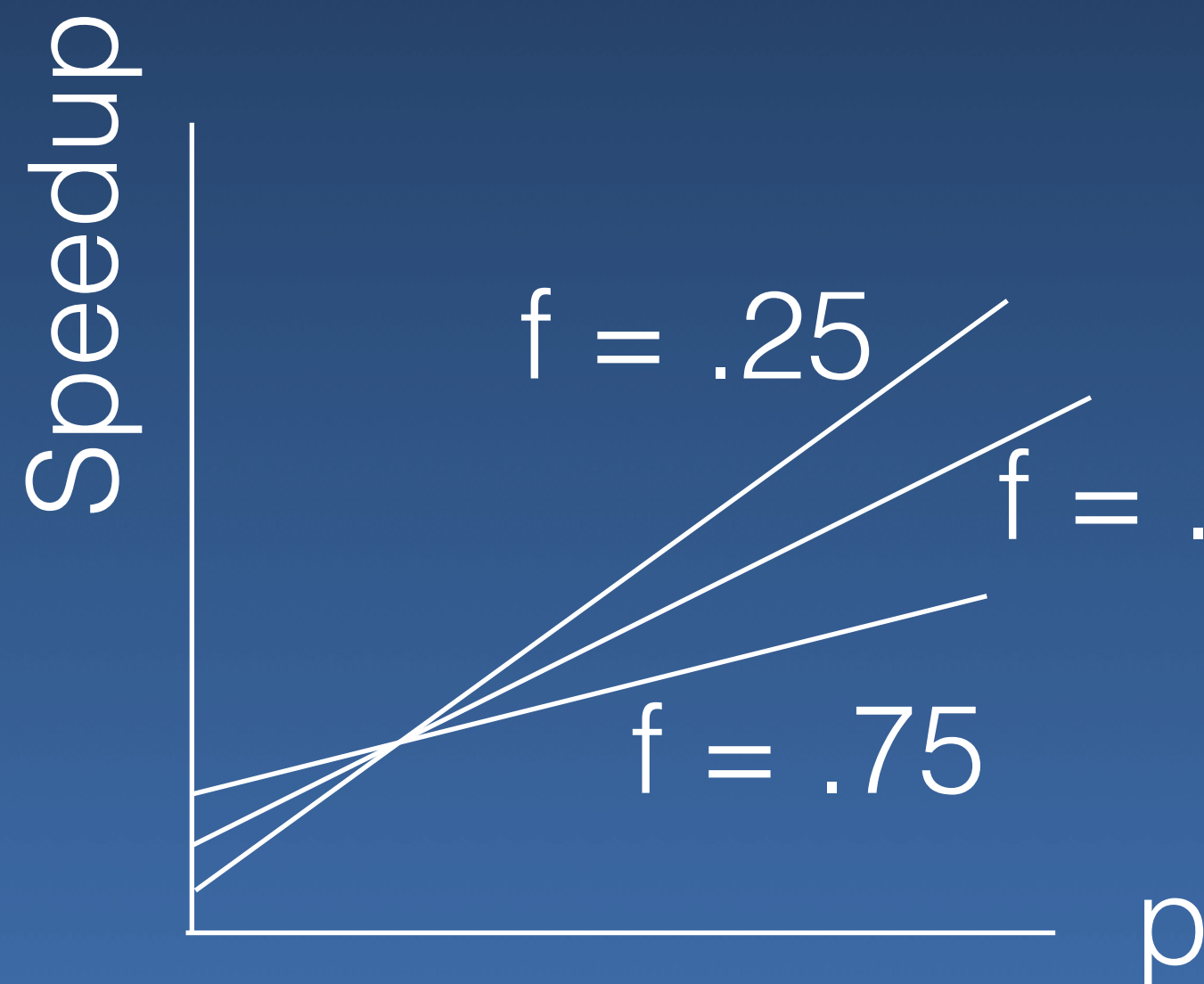
- Observation: applications seem to exceed Amdahl's speed-up
 - (i.e., assumption are too restrictive)
- As p increases, the opportunity for parallelization can also increase

Time taken for a problem on p processors = $\underbrace{t_{seq}}_{f t(n,p)} + \underbrace{t_{par}}_{(1-f) t(n,p)} = t(n,p)$

f = fraction of time spent in sequential work

$$t(n,1) = f t(n,p) + p (1-f) t(n,p)$$

$$\Rightarrow S_p = \frac{t(n,1)}{t(n,p)} = f + p(1-f) \neq \frac{1}{f + \frac{1-f}{p}} \quad (\text{Amdahl's law})$$



Karp Flatt metric

f
(Amdahl's law)

$$S_p = \frac{1}{f + \frac{1-f}{p}}$$

$$\Rightarrow S_p = \frac{p}{pf + 1 - f}$$

- Estimate the fractional part

$$\Rightarrow \frac{p}{S_p} - 1 = (p - 1)f$$

$$\Rightarrow f = \frac{\frac{1}{S_p} - \frac{1}{p}}{1 - \frac{1}{p}}$$

Measure speedup,
estimate the sequential fraction

Isoefficiency: Measure of Scalability

- Rate at which *problem size* must grow (as a function of p) to maintain constant *efficiency*

→ If n need not grow with $p \Rightarrow$ **Strongly scalable**

▶ **Weakly scalable**, otherwise

▶ **Lower** rate of growth needed \Rightarrow **More Scalable**

$$\mathcal{E}(n, p) = \frac{t(n, 1)}{p t(n, p)}$$

Isoefficiency: Measure of Scalability

- Rate at which *problem size* must grow (as a function of p) to maintain constant *efficiency*

→ If n need not grow with $p \Rightarrow$ **Strongly scalable**

▶ **Weakly scalable**, otherwise

▶ **Lower** rate of growth needed \Rightarrow **More Scalable**

$$\mathcal{E}(n, p) = \frac{t(n, 1)}{p t(n, p)} = k?$$

Isoefficiency: Measure of Scalability

- Rate at which *problem size* must grow (as a function of p) to maintain constant *efficiency*

→ If n need not grow with $p \Rightarrow$ **Strongly scalable**

▶ **Weakly scalable**, otherwise

▶ **Lower** rate of growth needed \Rightarrow **More Scalable**

$$\mathcal{E}(n, p) = \frac{t(n, 1)}{p \cdot t(n, p)} = k?$$

Isoefficiency: Measure of Scalability

- Rate at which *problem size* must grow (as a function of p) to maintain constant *efficiency*

→ If n need not grow with $p \Rightarrow$ **Strongly scalable**

▶ **Weakly scalable**, otherwise

▶ **Lower** rate of growth needed \Rightarrow **More Scalable**

$$\mathcal{E}(n, p) = \frac{t(n, 1) \uparrow}{p \downarrow t(n, p)} = k?$$

Isoefficiency: Measure of Scalability

- Rate at which *problem size* must grow (as a function of p) to maintain constant *efficiency*

→ If n need not grow with $p \Rightarrow$ **Strongly scalable**

▶ **Weakly scalable**, otherwise

▶ **Lower** rate of growth needed \Rightarrow **More Scalable**

$$t(n,1) = \mathcal{F}(n) \text{ "problem size"}$$

$$\mathcal{E}(n,p) = \frac{t(n,1) \uparrow}{p \downarrow t(n,p)} = k?$$

Isoefficiency: Measure of Scalability

- Rate at which *problem size* must grow (as a function of p) to maintain constant *efficiency*

→ If n need not grow with $p \Rightarrow$ **Strongly scalable**

▶ **Weakly scalable**, otherwise

▶ **Lower** rate of growth needed \Rightarrow **More Scalable**

$$t(n,1) = \mathcal{F}(n) \text{ "problem size"}$$

$$\mathcal{E}(n,p) = \frac{t(n,1) \uparrow}{p \downarrow t(n,p)} = k?$$

Measure of problem size $n(p)$ s.t. $\mathcal{E} = k$
 $\rightarrow \mathcal{F}(p)$

Isoefficiency: Measure of Scalability

- Rate at which *problem size* must grow (as a function of p) to maintain constant *efficiency*

→ If n need not grow with $p \Rightarrow$ **Strongly scalable**

▶ **Weakly scalable**, otherwise

▶ **Lower** rate of growth needed \Rightarrow **More Scalable**

$$t(n,1) = \mathcal{F}(n) \text{ "problem size"}$$

$$\mathcal{E}(n,p) = \frac{t(n,1) \uparrow}{p \downarrow t(n,p)} = k?$$

Example:

$$t(n,p) = \Theta\left(\frac{n}{p} + \log p\right)$$

$$t(n,1) = \Theta(n)$$

$$pt(n,p) = \Theta(n + p \log p) = \Theta(t(n,1)) = \Theta(n) \text{ if } \mathcal{J}(p) = \Omega(p \log p)$$

Measure of problem size $n(p)$ s.t. $\mathcal{E} = k$
 $\rightarrow \mathcal{J}(p)$

Isoefficiency: Measure of Scalability

- Rate at which *problem size* must grow (as a function of p) to maintain constant *efficiency*

$$t(n,1) = \mathcal{F}(n) \text{ "problem size"}$$

→ If n need not grow with $p \Rightarrow$ **Strongly scalable**

▶ **Weakly scalable**, otherwise

▶ **Lower** rate of growth needed \Rightarrow **More Scalable**

$$\mathcal{E}(n,p) = \frac{t(n,1) \uparrow}{p \downarrow t(n,p)} = k?$$

Example:

$$t(n,p) = \Theta\left(\frac{n}{p} + \log p\right)$$

$$t(n,1) = \Theta(n)$$

Measure of problem size $n(p)$ s.t. $\mathcal{E} = k$
 $\rightarrow \mathcal{J}(p)$

Isoefficiency Function $\mathcal{J}(p) = \Omega(\bar{o}(p))$

$$pt(n,p) = \Theta(n + p \log p) = \Theta(t(n,1)) = \Theta(n) \text{ if } \mathcal{J}(p) = \Omega(p \log p)$$