COL380

Introduction to
Parallel & Distributed Programming

Measure Performance

- A function of: input size, number of processors
  - ➡ communication, dependencies
- Scaling
- How fast does a job complete?
  - ➡ Elapsed wall time (Latency)
    - compute + communicate + synchronize
- How many jobs complete in a given time?
  - ➡ Throughput

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

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

Cost, $C_p = p \times t_p$

Cost Optimal if $C_p = t_1$

Look out for inefficiency:

$t_1 = n^3$

$t_p = n^2$, 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
- To execute using fewer processors $p^\# < p$:
  - ➡ Simulate each step (that uses up to p processors)
    - Group into p sets, one for each processor
    - Each set takes $\lfloor p/p^\# \rfloor$ or $\lceil p/p^\# \rceil$ steps
    - The time taken for original step = $O(\lceil p/p^\# \rceil)$
- The total time taken $t(n,p^\#) = O(t(n,p) * \lceil p/p^\# \rceil)$

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

processors executing step i

Subodh Kumar

Amdahl's Law

- 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

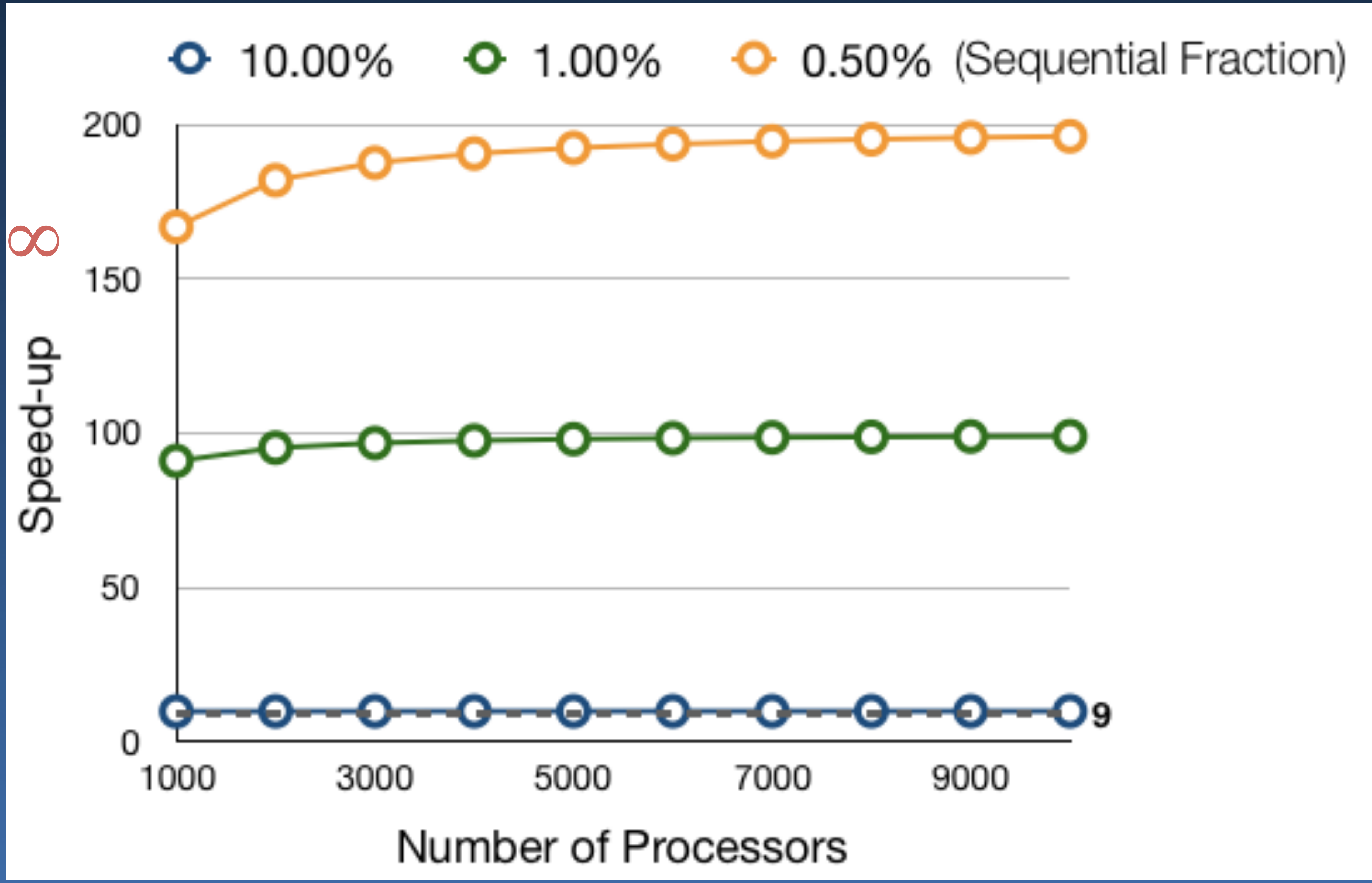
$$S_p = \frac{1}{f + \frac{1-f}{p}} \quad \rightarrow 0$$

- Speed-up due to *p* processors

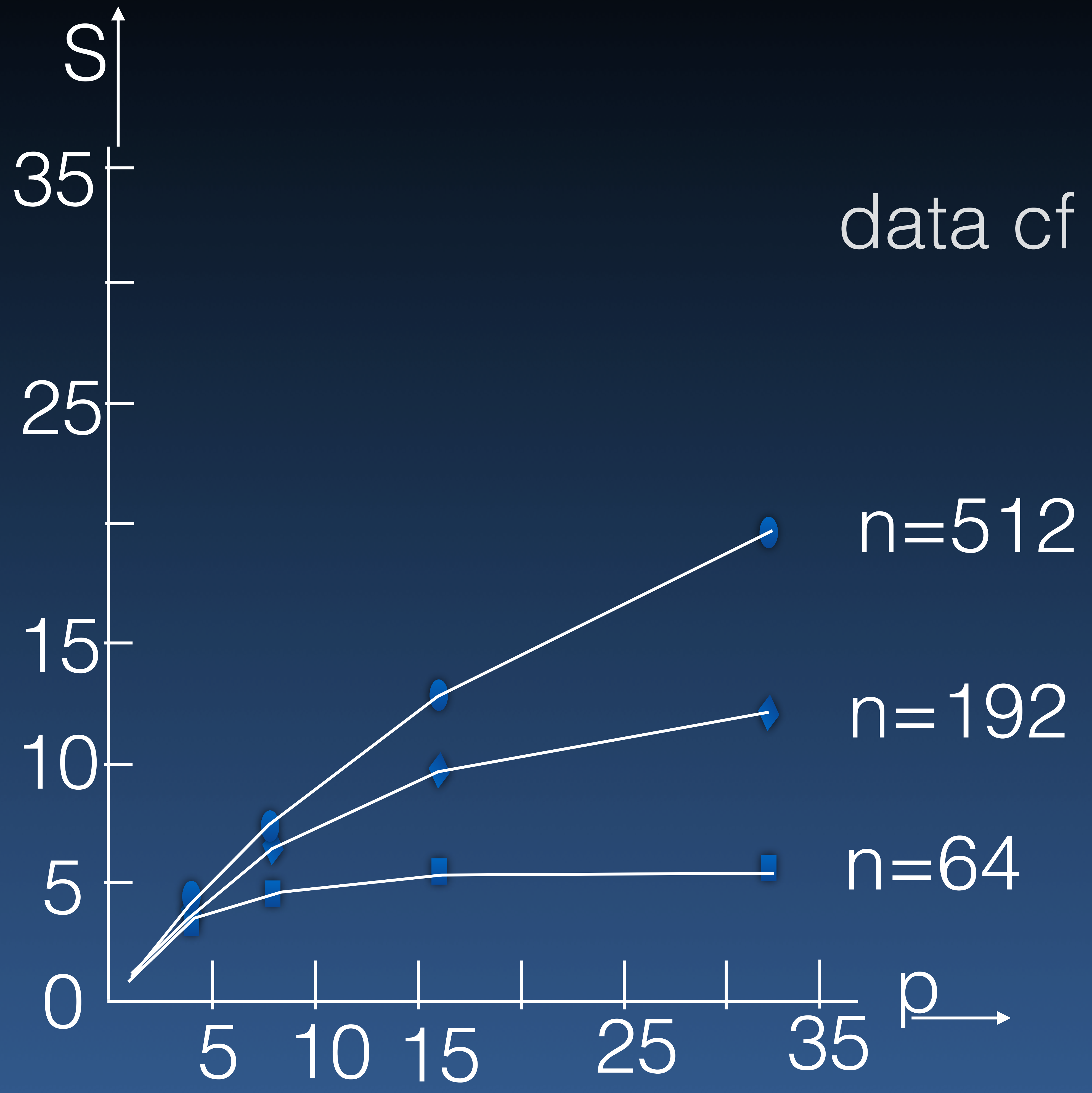
- Upper bound on speedup at ***p* = ∞**

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

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



Example Scaling



data cf [Gramma et al.]

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

Speedup saturates and efficiency drop
(Amdahl's law)
But the limit depends on n:
size of the problem

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

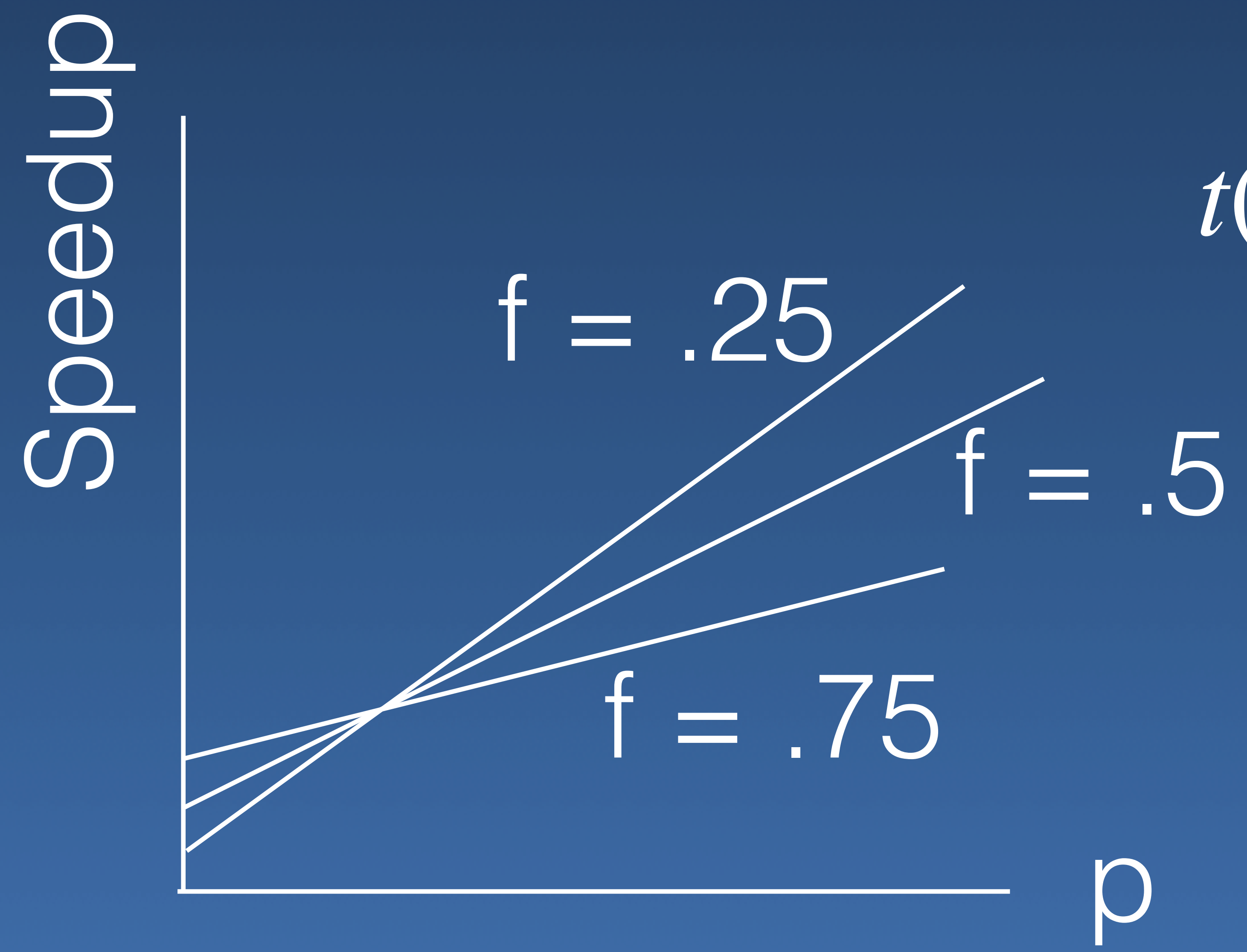
Time taken for a problem on p processors = $\underbrace{\text{tseq}}_{f\ t(n,p)} + \underbrace{\text{tpar}}_{(1-f)\ t(n,p)} = t(n,p)$

f = fraction of time spent in sequential work

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

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

(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

Subodh Kumar

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 $\Rightarrow$ *More* Scalable

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

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

Measure of problem size: $n(p) = \mathscr{I}(p)\ \ s.t.\ \mathscr{E} = k$

Isoefficiency Function $\mathscr{I}(p) = \Omega(\bar{o}(p))$

Example:

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

$t(n,1) = \Theta(n)$ $\qquad \mathscr{I}(p) = \Omega(p \log p)$

Subodh Kumar