Parallel Programming
in C with MPI and OpenMP

Michael J. Quinn

Chapter 7
Performance Analysis

Learning Objectives

- Predict performance of parallel programs
- Understand barriers to higher performance

Outline

- General speedup formula
- Amdahl’s Law
- Gustafson-Barsis’ Law
- Karp-Flatt metric
- Isoefficiency metric

Speedup Formula

$$\text{Speedup} = \frac{\text{Sequential execution time}}{\text{Parallel execution time}}$$

Execution Time Components

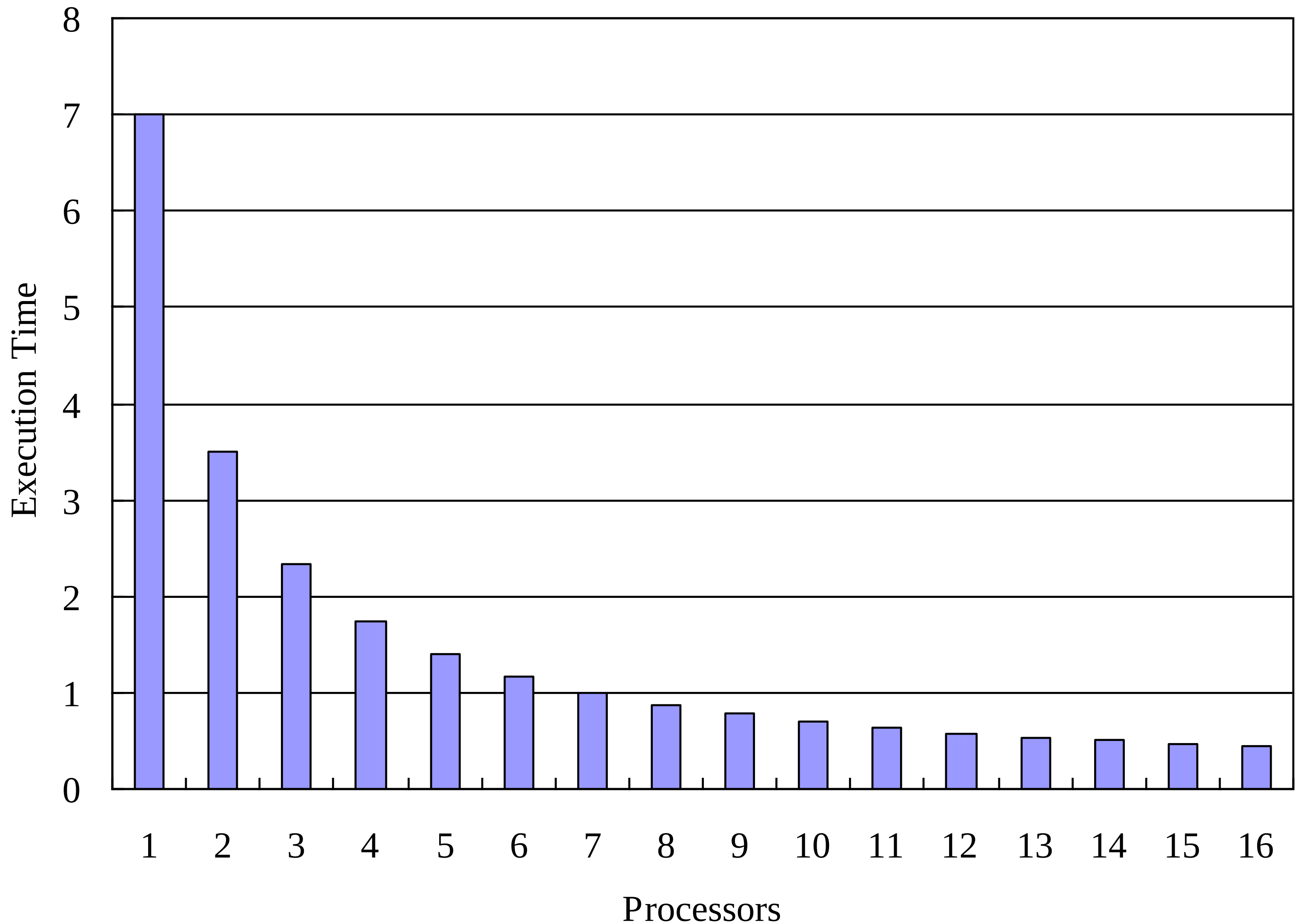
- Inherently sequential computations: $\sigma(n)$
- Potentially parallel computations: $\varphi(n)$
- Communication operations: $\kappa(n, p)$

Speedup Expression

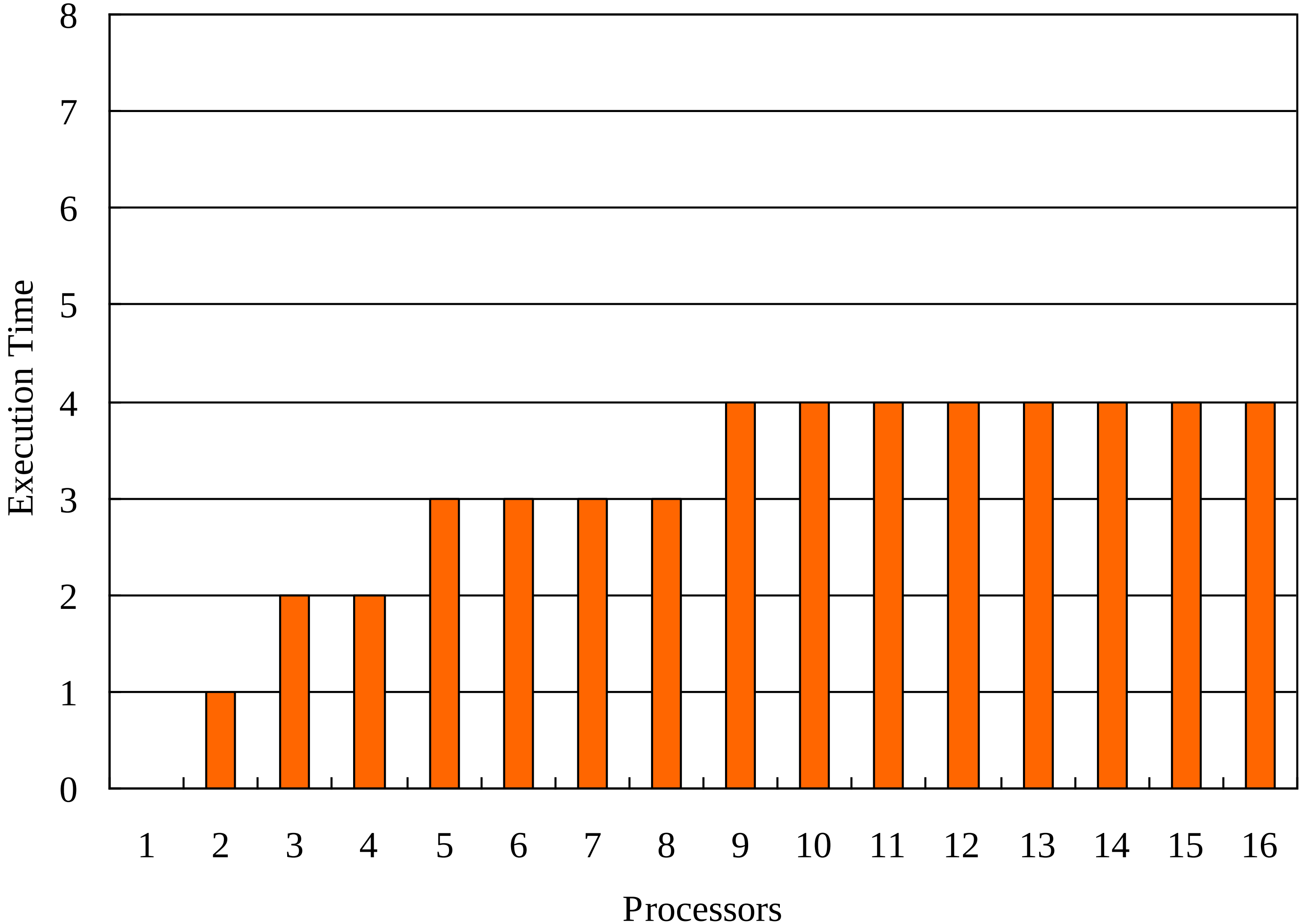
- $\psi(n, p)$: speedup achieved solving a problem of size n on p processors.

$$\psi(n,p) \le \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p+\kappa(n,p)}$$

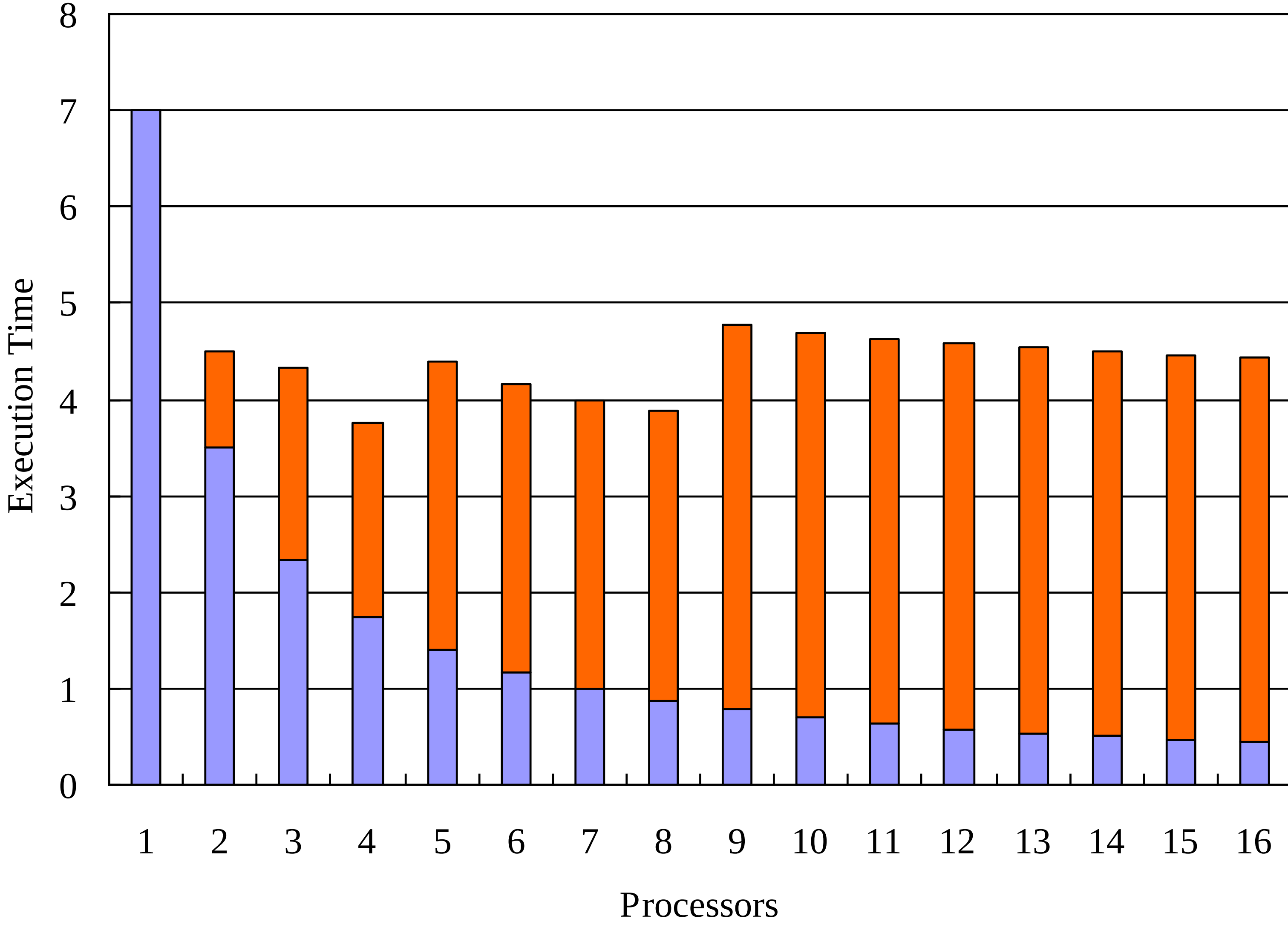
Parallel Component: $\varphi(n)/p$



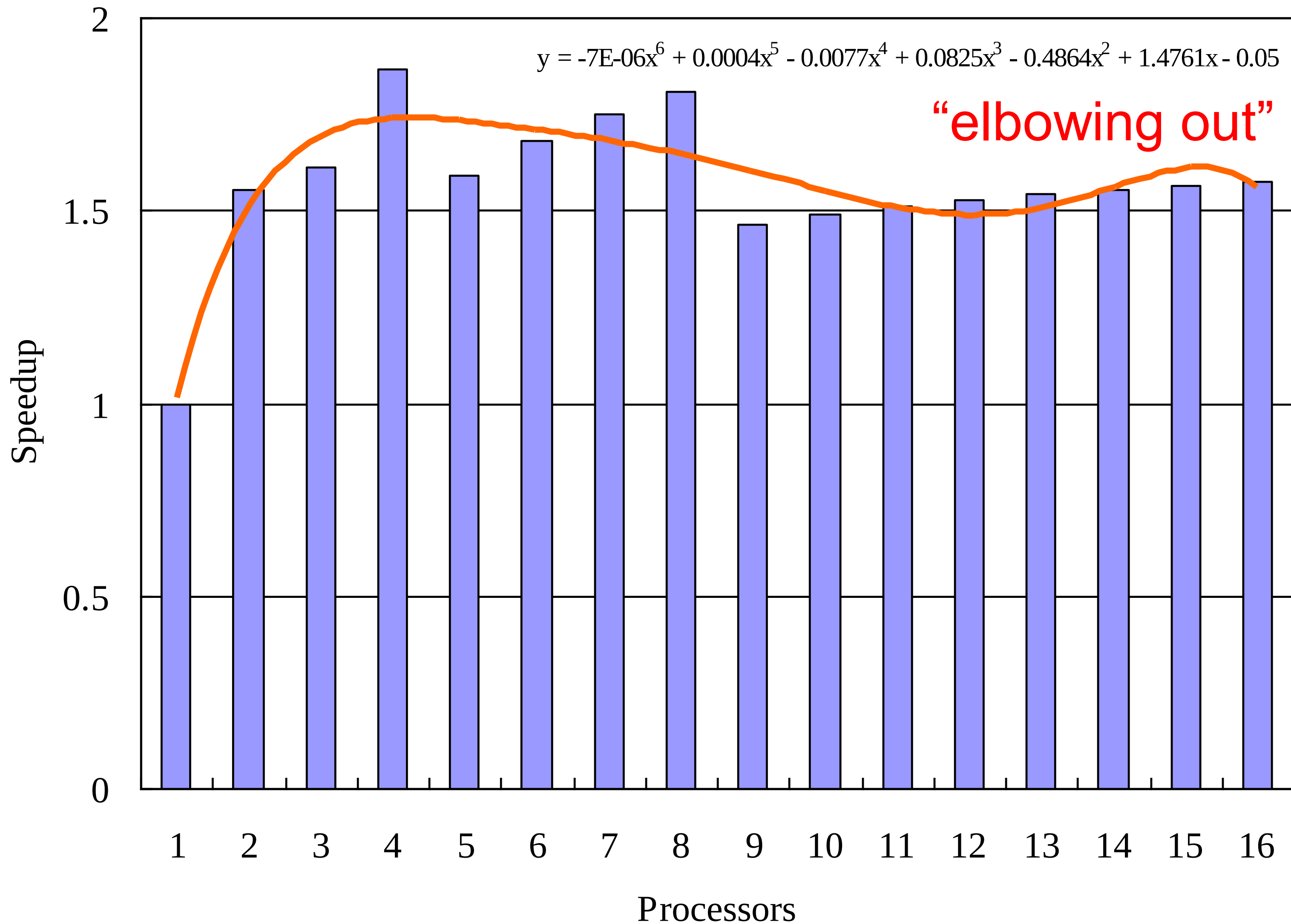
Communication Component: $\kappa(n, p)$



$\varphi(n)/p + \kappa(n, p)$



Speedup Plot



Efficiency

$$\text{Efficiency} = \frac{\text{Sequential execution time}}{\text{Processors used} \times \text{Parallel execution time}}$$

$$= \frac{\text{Speedup}}{\text{Processors used}}$$

$0 \le \varepsilon(n,p) \le 1$

$$\varepsilon(n,p) \le \frac{\sigma(n)+\varphi(n)}{p\sigma(n)+\varphi(n)+p\kappa(n,p)}$$

- All terms > 0 $\Rightarrow \varepsilon(n,p) > 0$
- Denominator > numerator $\Rightarrow \varepsilon(n,p) < 1$

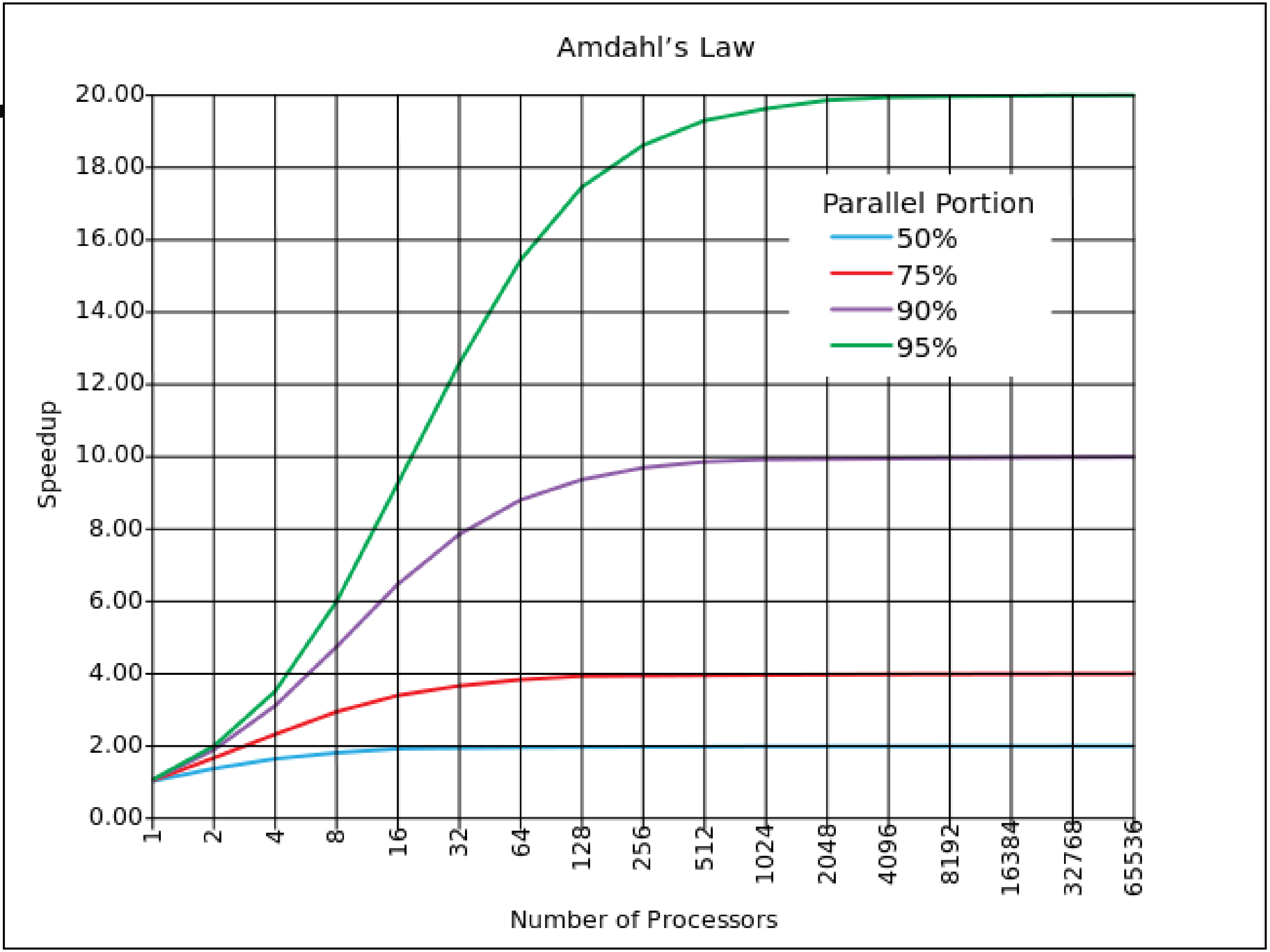
Amdahl's Law

$$\psi(n,p) \le \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p+\kappa(n,p)}$$

$$\le \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p}$$

Let $f = \sigma(n)/(\sigma(n) + \varphi(n))$

$$\psi \le \frac{1}{f+(1-f)/p}$$


Amdahl's Law
Parallel Portion
50%
75%
90%
95%
Speedup
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of Processors

Example 1

- 95% of a program's execution time occurs inside a loop that can be executed in parallel. What is the maximum speedup we should expect from a parallel version of the program executing on 8 CPUs?

$$\psi \leq \frac{1}{0.05 + (1 - 0.05)/8} \cong 5.9$$

Example 2

- 20% of a program's execution time is spent within inherently sequential code. What is the limit to the speedup achievable by a parallel version of the program?

$$\lim_{p \to \infty} \frac{1}{0.2 + (1 - 0.2)/p} = \frac{1}{0.2} = 5$$

Pop Quiz

- An oceanographer gives you a serial program and asks you how much faster it might run on 8 processors. You can only find one function amenable to a parallel solution. Benchmarking on a single processor reveals 80% of the execution time is spent inside this function. What is the best speedup a parallel version is likely to achieve on 8 processors?

Pop Quiz

- A computer animation program generates a feature movie frame-by-frame. Each frame can be generated independently and is output to its own file. If it takes 99 seconds to render a frame and 1 second to output it, how much speedup can be achieved by rendering the movie on 100 processors?

Limitations of Amdahl's Law

- Ignores $\kappa(n, p)$
- Overestimates speedup achievable

Amdahl Effect

- Typically $\kappa(n, p)$ has lower complexity than $\varphi(n)/p$
- As n increases, $\varphi(n)/p$ dominates $\kappa(n,p)$
- As n increases, speedup increases

Illustration of Amdahl Effect

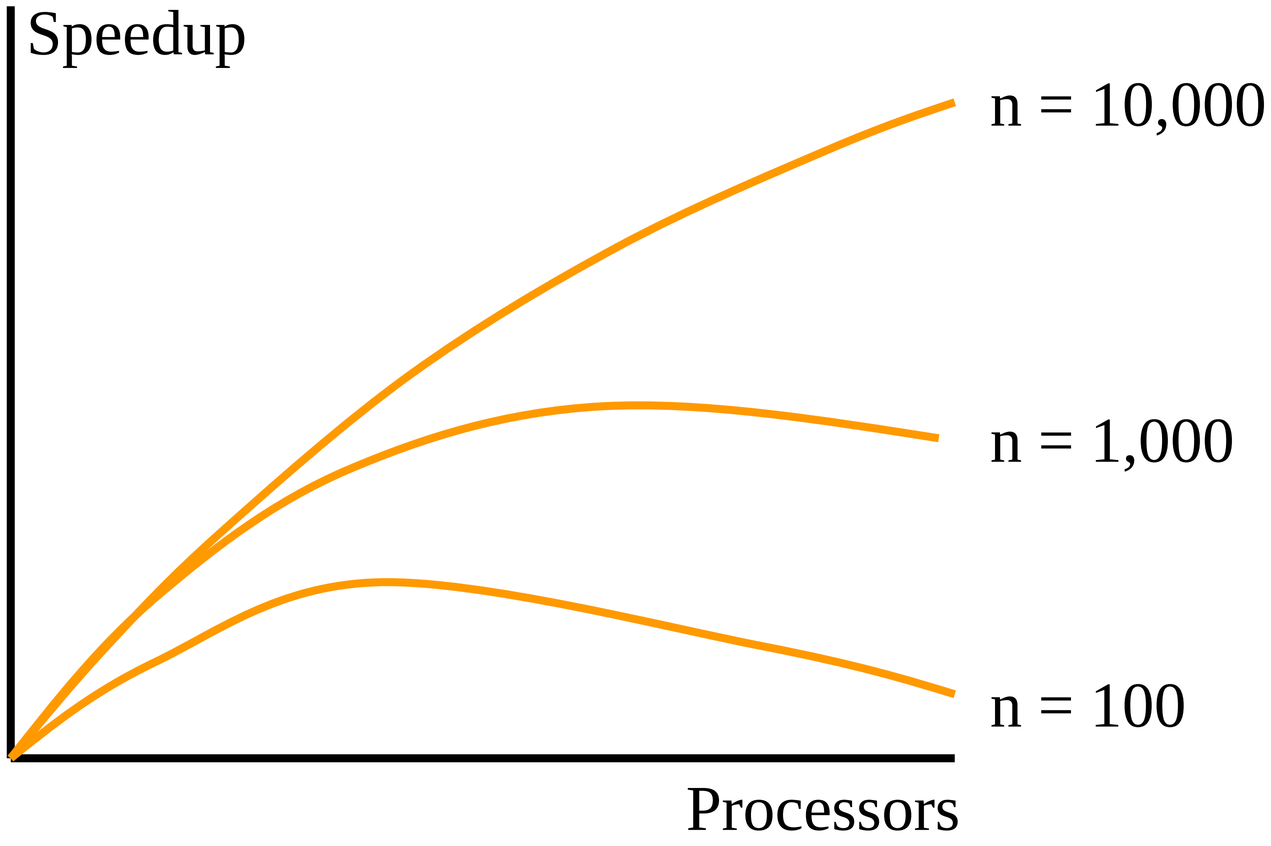
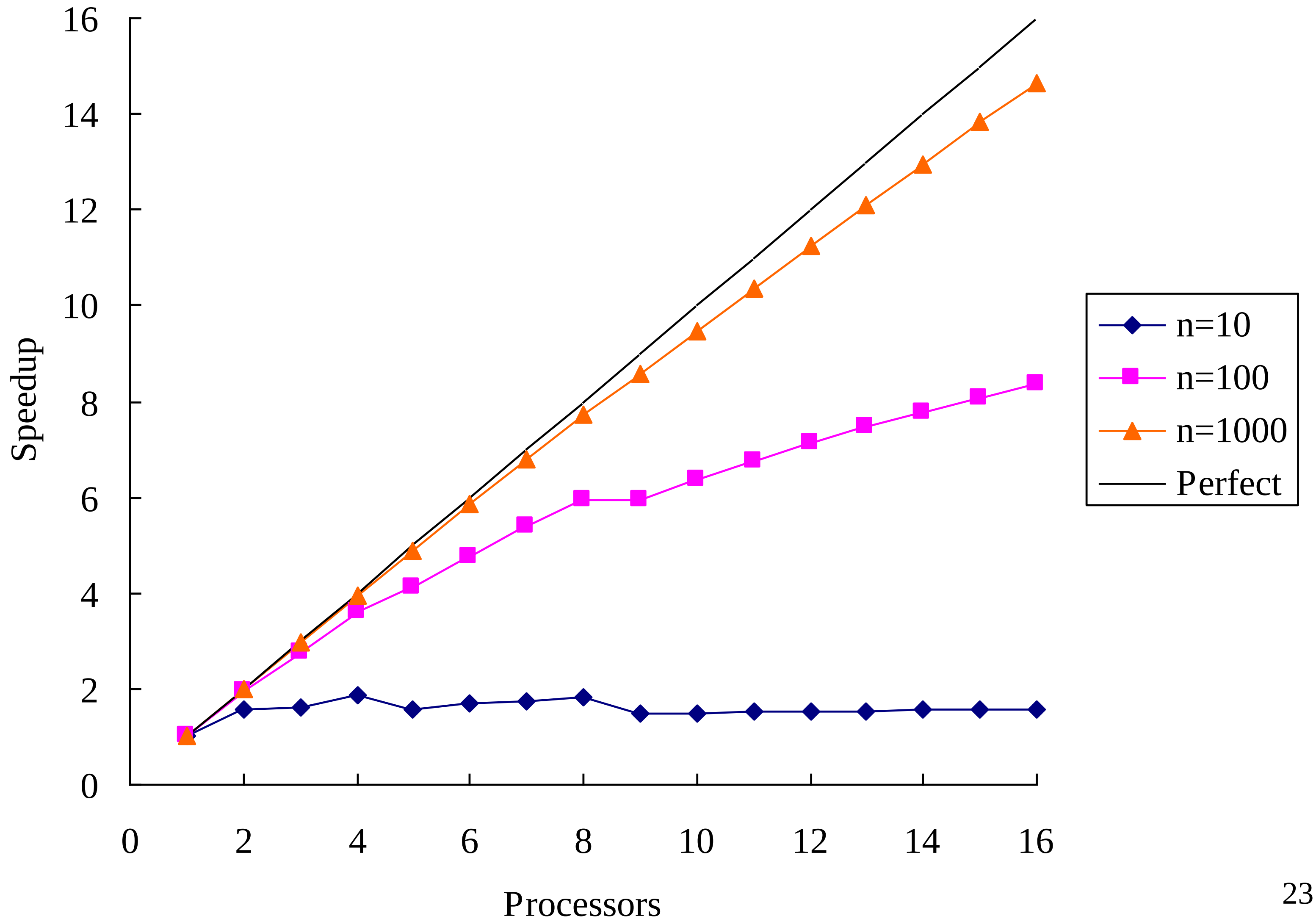


Illustration of Amdahl Effect



Review of Amdahl's Law

- Treats problem size as a constant
- Shows how execution time decreases as number of processors increases

Another Perspective

- We often use faster computers to solve larger problem instances
- Let's treat time as a constant and allow problem size to increase with number of processors

Gustafson-Barsis's Law

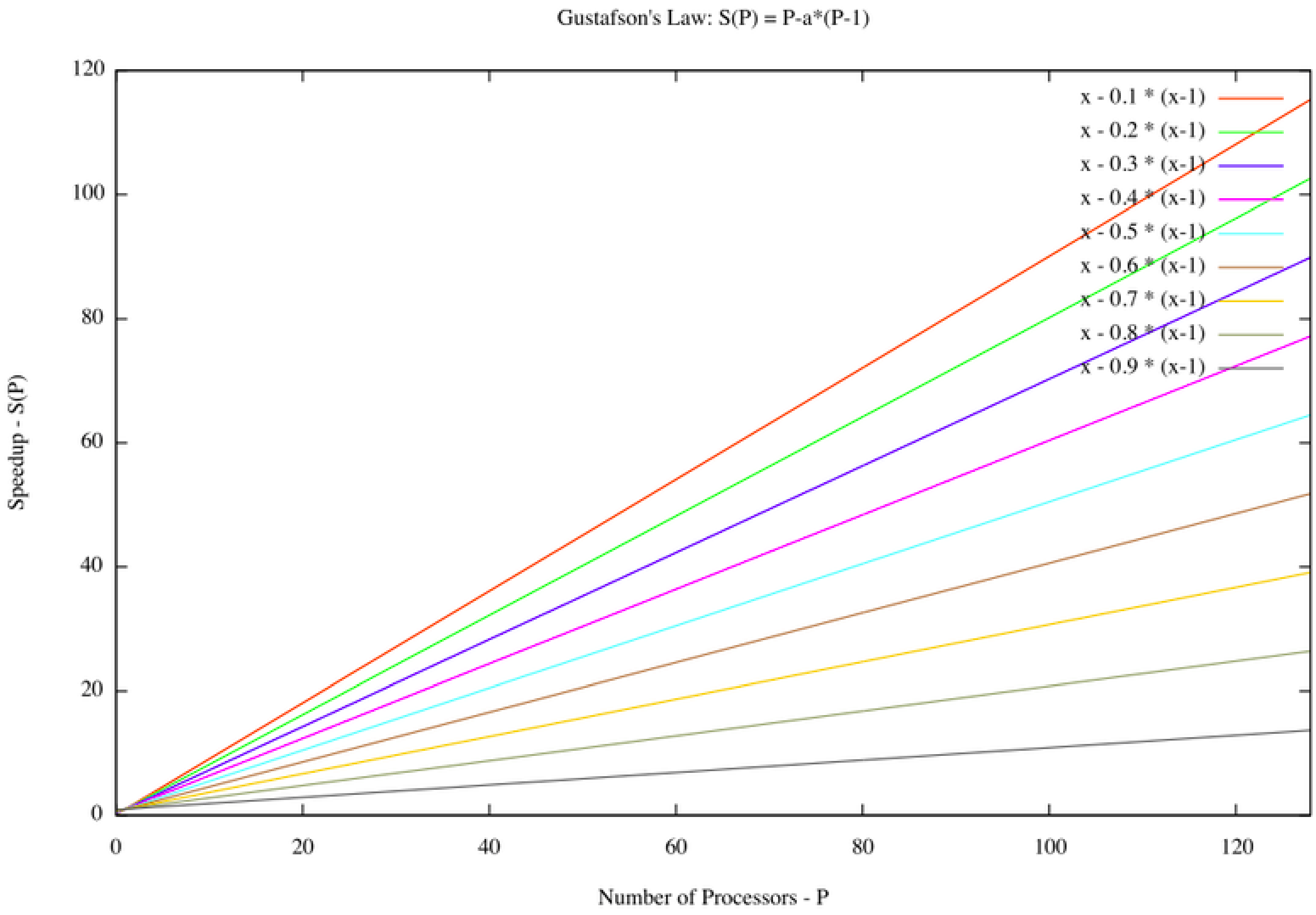
$$\psi(n,p) \le \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p}$$

- Let $s = \sigma(n)/(\sigma(n)+\varphi(n)/p)$

$$\psi \le p + (1-p)s$$

Gustafson-Barsis's Law

- Begin with parallel execution time
- Estimate sequential execution time to solve same problem
- Problem size is an increasing function of p
- Predicts **scaled speedup**


Gustafson's Law: S(P) = P-a*(P-1)
x - 0.1 * (x-1)
x - 0.2 * (x-1)
x - 0.3 * (x-1)
x - 0.4 * (x-1)
x - 0.5 * (x-1)
x - 0.6 * (x-1)
x - 0.7 * (x-1)
x - 0.8 * (x-1)
x - 0.9 * (x-1)
Speedup - S(P)
Number of Processors - P
0
20
40
60
80
100
120

Example 1

- An application running on 10 processors spends 3% of its time in serial code. What is the scaled speedup of the application?

$$\psi = 10 + (1 - 10)(0.03) = 10 - 0.27 = 9.73$$

…except 9 do not have to execute serial code

Execution on 1 CPU takes 10 times as long…

Example 2

- What is the maximum fraction of a program's parallel execution time that can be spent in serial code if it is to achieve a scaled speedup of 7 on 8 processors?

$$7 = 8 + (1-8)s \Rightarrow s \approx 0.14$$

Pop Quiz

- A parallel program executing on 32 processors spends 5% of its time in sequential code. What is the scaled speedup of this program?

The Karp-Flatt Metric

- Amdahl's Law and Gustafson-Barsis' Law ignore $\kappa(n, p)$
- They can overestimate speedup or scaled speedup
- Karp and Flatt proposed another metric

Experimentally Determined Serial Fraction

$$e = \frac{\sigma(n) + \kappa(n,p)}{T(n,1)}$$

$$e = \frac{(p-1)\sigma(n) + p\kappa(n,p)}{(p-1)T(n,1)}$$

Inherently serial component of parallel computation + processor communication and synchronization overhead
/
Single processor execution time

$$e = \frac{1/\psi - 1/p}{1 - 1/p}$$

Derive the Definition of *e*

$$T(n,p) = \sigma(n) + \frac{\varphi(n)}{p} + \kappa(n,p) \qquad T(n,1) = \sigma(n) + \varphi(n)$$

$$T(n,p) = T(n,1)e + \frac{T(n,1)(1-e)}{p}$$

$$pT(n,p) = pT(n,1)e + T(n,1) - eT(n,1)$$

$$e = \frac{pT(n,p) - T(n,1)}{(p-1)T(n,1)} = \frac{(p-1)\sigma(n) + p\kappa(n,p)}{(p-1)T(n,1)}$$

Compare *e* with *f*

$$T(n,p)=\sigma(n)+\frac{\varphi(n)}{p}$$

Simple model

$$T(n,p)=T(n,1)?+\frac{T(n,1)(1-?)}{p}$$

$$?=f=\frac{\sigma(n)}{T(n,1)}$$

Complex model

$$T(n,p)=\sigma(n)+\frac{\varphi(n)}{p}+\boxed{\kappa(n,p)}$$

$$?=e=\frac{(p-1)\sigma(n)+p\kappa(n,p)}{(p-1)T(n,1)}$$

Derive the Karp-Flatt Metric

$$T(n,p) = T(n,1)e + \frac{T(n,1)(1-e)}{p} \qquad \psi = \frac{T(n,1)}{T(n,p)}$$

$$T(n,p) = T(n,p)\psi e + \frac{T(n,p)\psi(1-e)}{p}$$

$$1 = \psi e + \frac{\psi(1-e)}{p}$$

$$\frac{1}{\psi} = e + \frac{1}{p} - \frac{e}{p}$$

$$e = \frac{1/\psi - 1/p}{1 - 1/p}$$

Experimentally Determined Serial Fraction

- Takes into account parallel overhead
- Detects other sources of overhead or inefficiency ignored in speedup model
  - Process startup time
  - Process synchronization time
  - Imbalanced workload
  - Architectural overhead

Example 1

- What is the primary reason for speedup of only 4.7 on 8 CPUs?

p	2	3	4	5	6	7	8
ψ	1.8	2.5	3.1	3.6	4.0	4.4	4.7

- Since e is constant, large serial fraction is the primary reason.

e	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Example 2

- What is the primary reason for speedup of only 4.7 on 8 CPUs?

p	2	3	4	5	6	7	8
ψ	1.9	2.6	3.2	3.7	4.1	4.5	4.7

- Since e is steadily increasing, overhead is the primary reason.

e	0.070	0.075	0.080	0.085	0.090	0.095	0.100

Pop Quiz

- Is this program likely to achieve a speedup of 10 on 12 processors?

p	4	8	12
ψ	3.9	6.5	?

$$e = \frac{(p-1)\sigma(n) + \boxed{p\kappa(n,p)\uparrow}}{(p-1)T(n,1)}$$

p	4	8	12
ψ	3.9	6.5	10
e	0.008547	0.032967	0.018182

e decreasing is impossible.

Isoefficiency Metric

- Parallel system: parallel program executing on a parallel computer
- Scalability of a parallel system: measure of its ability to increase performance as number of processors increases
- A scalable system maintains efficiency as processors are added
- Isoefficiency: way to measure scalability

Isoefficiency Derivation Steps

- Begin with speedup formula
- Compute total amount of overhead
- Assume efficiency remains constant
- Determine relation between sequential execution time and overhead

Deriving Isoefficiency Relation

- Determine overhead

$$T_O(n,p) = pT(n,p) - T(n,1)$$
$$= (p-1)\sigma(n) + p\kappa(n,p)$$

- Substitute overhead into speedup equation

$$\varepsilon(n,p) = \frac{\psi(n,p)}{p} \le \frac{(\sigma(n)+\varphi(n))}{\sigma(n)+\varphi(n)+T_O(n,p)}$$

- Substitute $T(n,1) = \sigma(n) + \varphi(n)$. Assume efficiency is constant.

$$T(n,1) \ge CT_O(n,p)$$

Isoefficiency Relation

Deriving Isoefficiency Relation (2)

$$\psi(n,p) \le \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\frac{\varphi(n)}{p}+\kappa(n,p)} = \frac{p(\sigma(n)+\varphi(n))}{p\sigma(n)+\varphi(n)+p\kappa(n,p)}$$

$$= \frac{p(\sigma(n)+\varphi(n))}{\sigma(n)+\varphi(n)+(p-1)\sigma(n)+p\kappa(n,p)} = \frac{p(T(n,1))}{T(n,1)+T_O(n,p)}$$

$$\varepsilon(n,p) = \frac{\psi(n,p)}{p} \le \frac{T(n,1)}{T(n,1)+T_O(n,p)}$$

$$T(n,1) \ge \frac{\varepsilon(n,p)}{1-\varepsilon(n,p)} T_O(n,p)$$

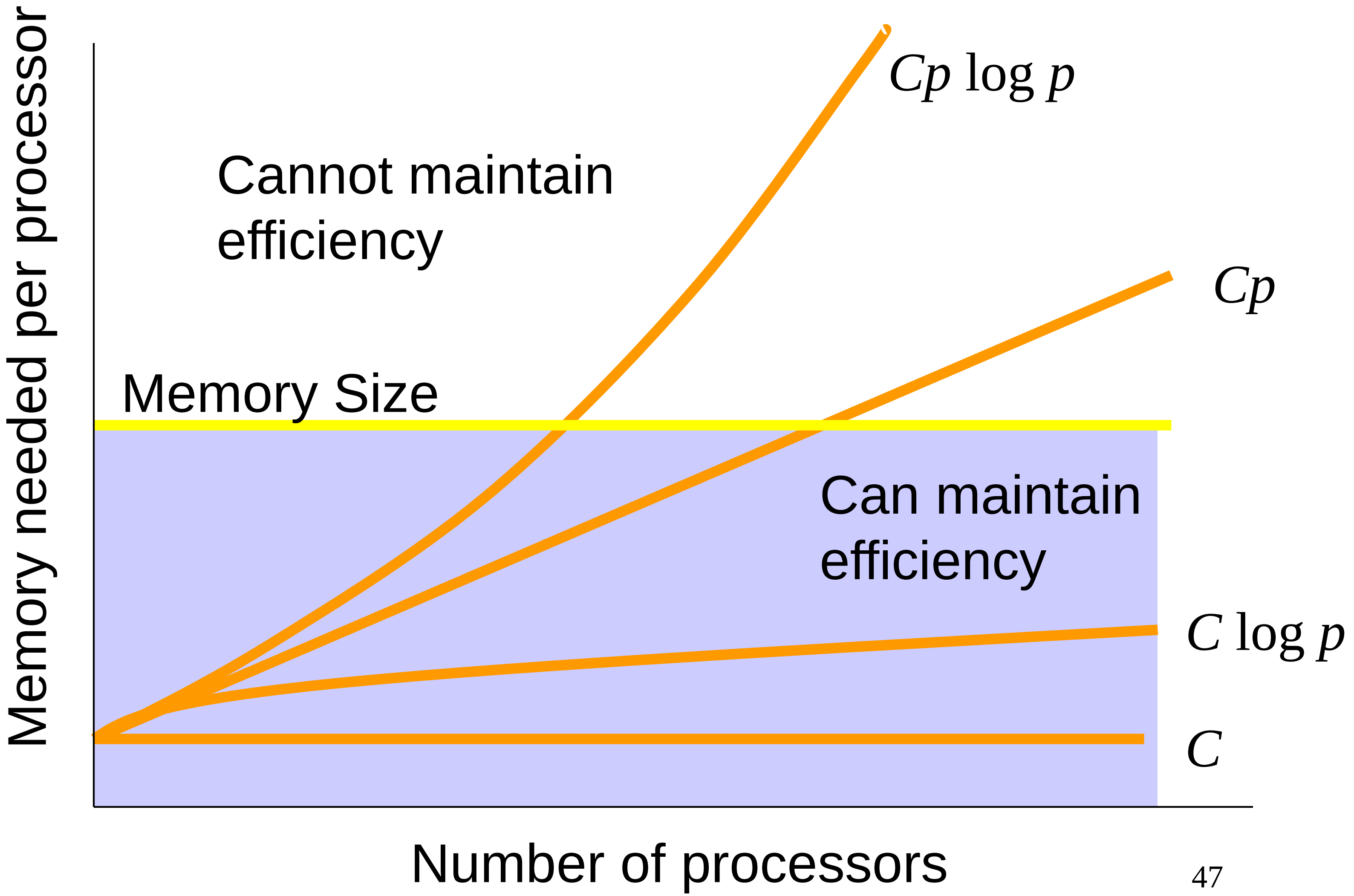
Scalability Function

- Suppose the final **isoefficiency relation** is $n \geq f(p)$
- Let $M(n)$ denote memory required for problem of size n
- $M(f(p))/p$ shows how memory usage **per processor** must increase to maintain same efficiency
- We call $M(f(p))/p$ the scalability function

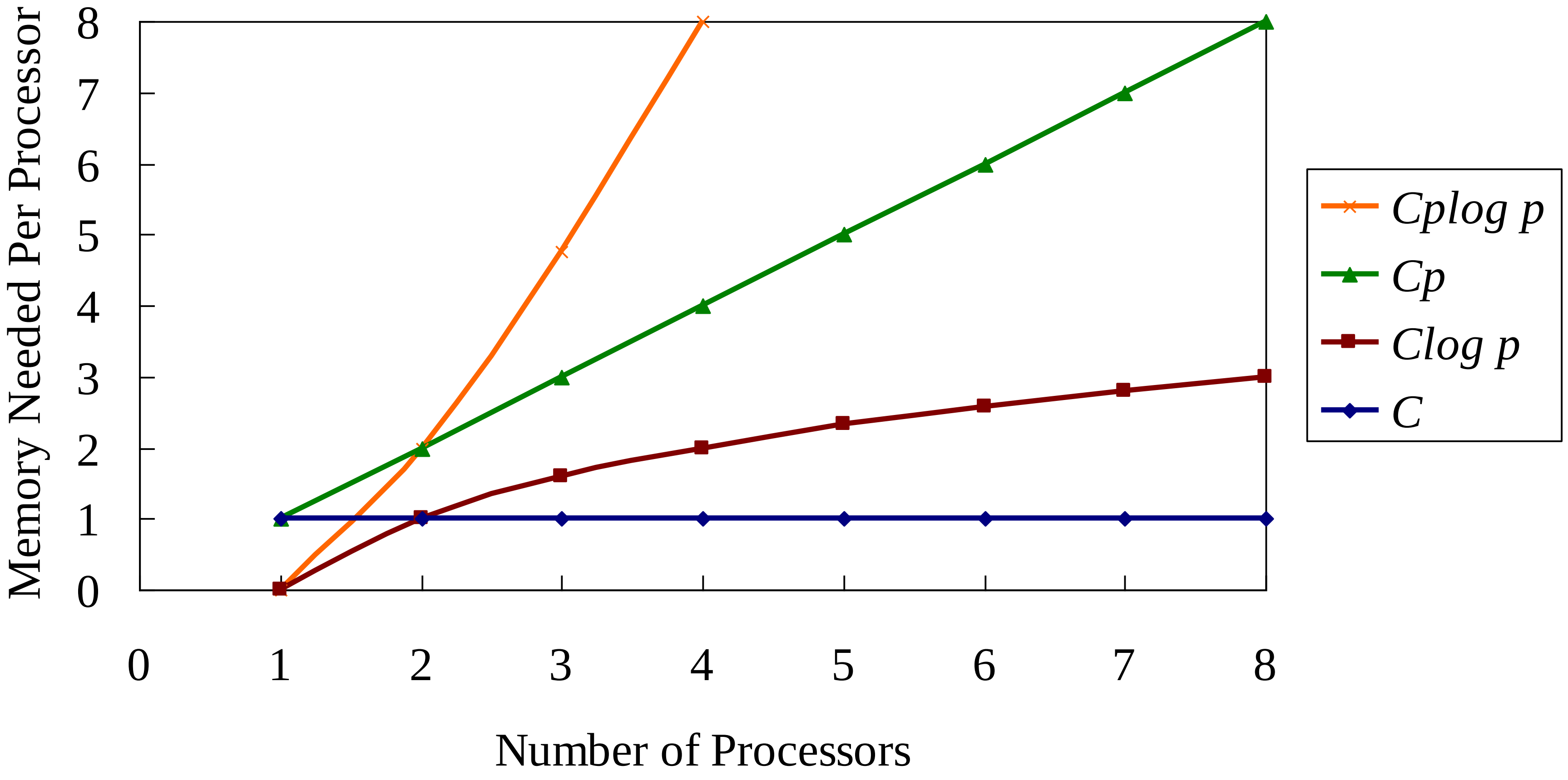
Meaning of Scalability Function

- To maintain efficiency when increasing p, we must increase n
- Maximum problem size limited by available memory, which is linear in p
- Scalability function shows how memory usage per processor must grow to maintain efficiency
- Scalability function a constant means parallel system is perfectly scalable

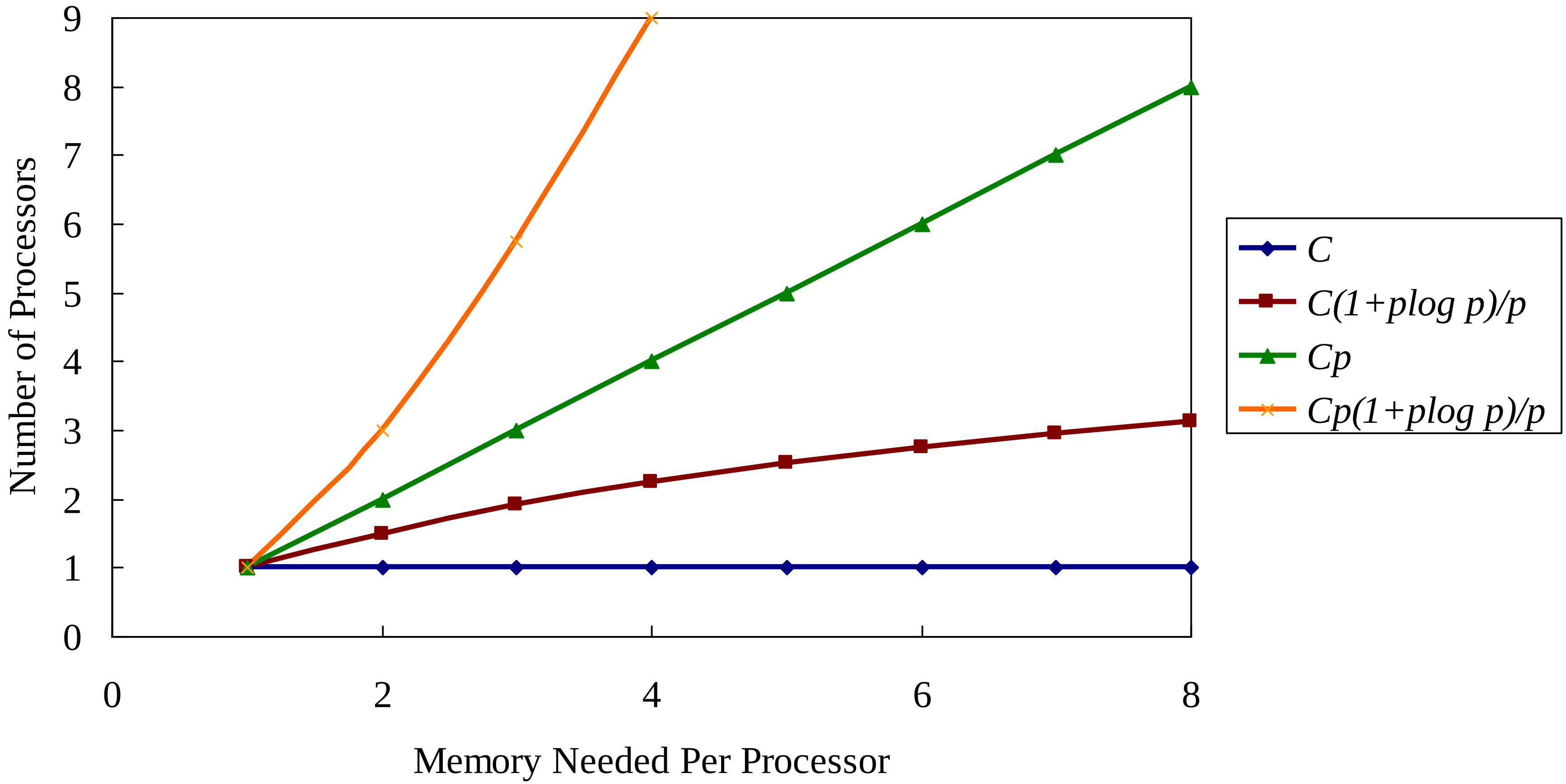
Interpreting Scalability Function

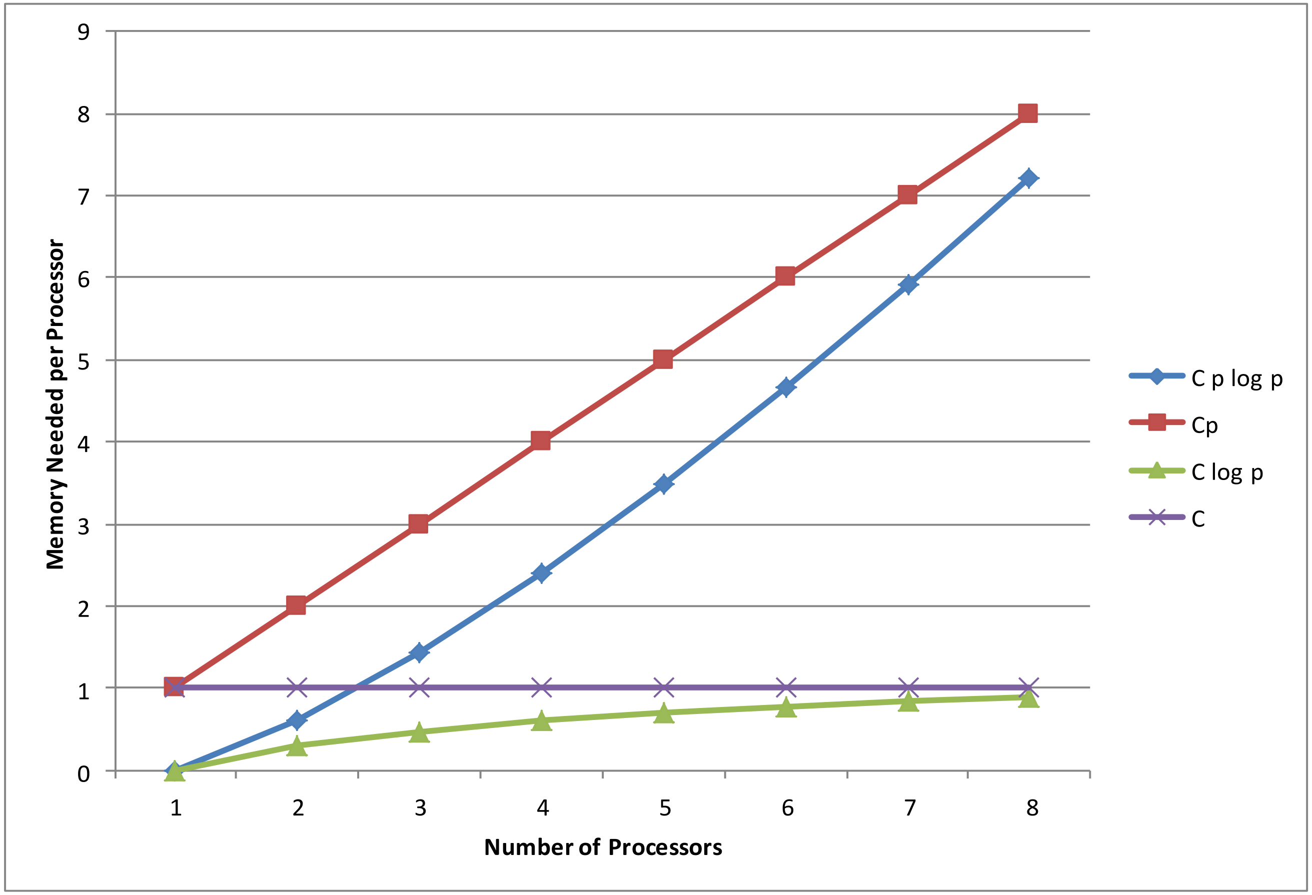


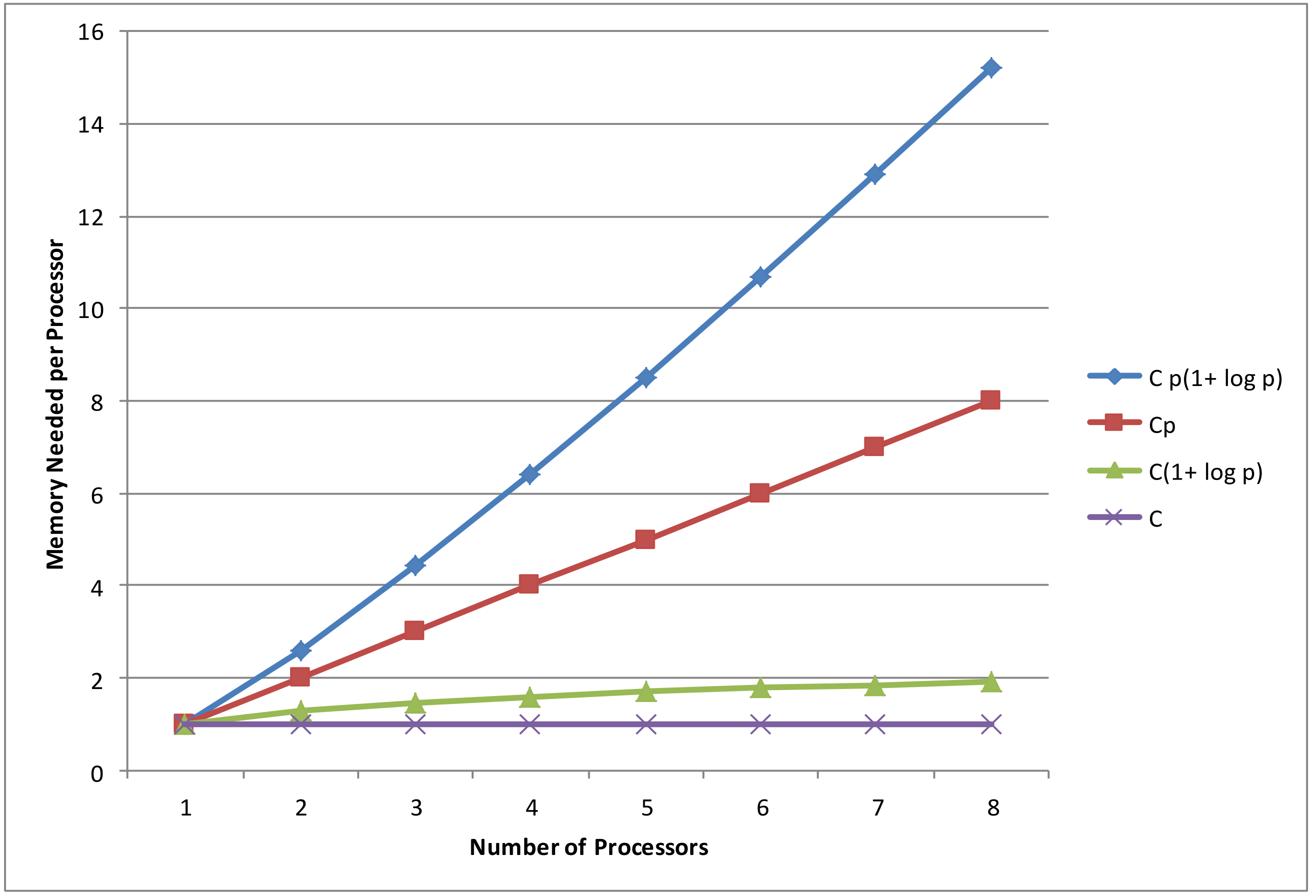
Interpreting Scalability Function



Interpreting Scalability Function




9
8
7
6
5
4
3
2
1
0
Memory Needed per Processor
1
2
3
4
5
6
7
8
Number of Processors
C p log p
Cp
C log p
C


16
14
12
10
8
6
4
2
0
Memory Needed per Processor
1
2
3
4
5
6
7
8
Number of Processors
C p(1+ log p)
Cp
C(1+ log p)
C

Example 1: Reduction

- Sequential algorithm complexity
  $T(n, 1) = \Theta(n)$
- Parallel algorithm
  - Computational complexity = $\Theta(n/p)$
  - Communication complexity = $\Theta(\log p)$
- Parallel overhead
  $T_O(n, p) = \Theta(p \log p)$

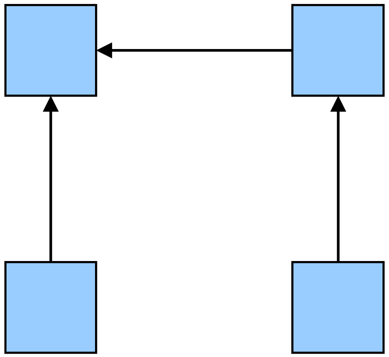
Reduction (continued)

- Isoefficiency relation: $n \geq Cp \log p$
- We ask: To maintain same level of efficiency, how must n increase when p increases?
- $M(n) = n$

$$M\left(Cp \log p\right)/p = Cp \log p / p = C \log p$$

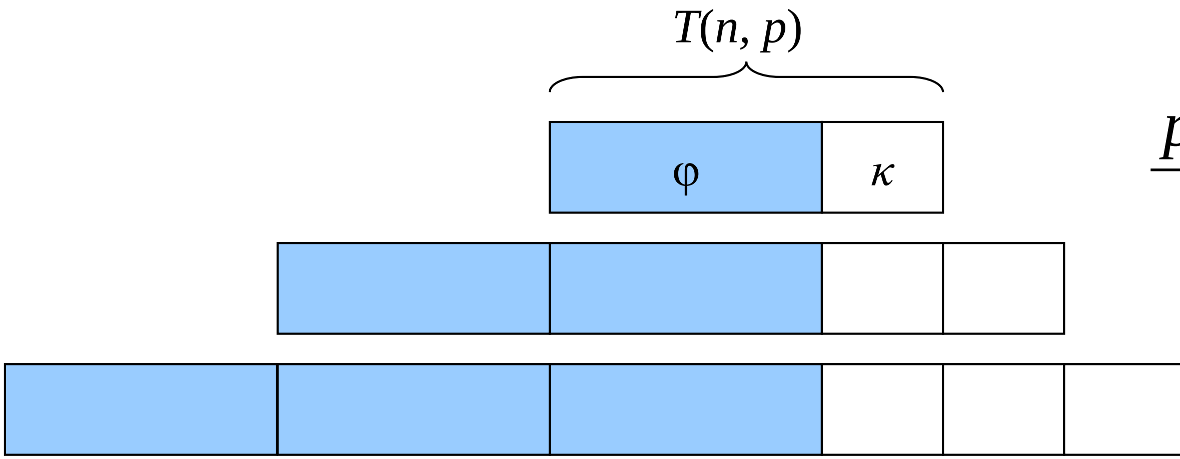
- The system has good scalability

Reduction



$$\varepsilon(n,p) \le \frac{\sigma(n)+\varphi(n)}{p\sigma(n)+\varphi(n)+p\kappa(n,p)}$$

Maintain isoefficiency



$$\sigma(n) \to \text{ignored}$$

$$\frac{p\kappa(n,p)}{\varphi(n)} \to \text{constant}$$

Example 2: Floyd's Algorithm

- Sequential time complexity: $\Theta(n^3)$
- Parallel computation time: $\Theta(n^3/p)$
- Parallel communication time: $\Theta(n^2 \log p)$
- Parallel overhead: $T_O(n, p) = \Theta(pn^2 \log p)$

Floyd's Algorithm (continued)

- Isoefficiency relation

  $n^3 \geq C(p\, n^2 \log p) \Rightarrow n \geq C\, p \log p$

- $M(n) = n^2$

$$M\left(Cp\log p\right)/p = C^2 p^2 \log^2 p/p = C^2 p \log^2 p$$

- The parallel system has poor scalability

Example 3: Finite Difference

- Sequential time complexity per iteration: $\Theta(n^2)$
- Parallel communication complexity per iteration: $\Theta(n/\sqrt{p})$
- Parallel overhead: $\Theta(n \sqrt{p})$

Finite Difference (continued)

- Isoefficiency relation
  $n^2 \geq Cn\sqrt{\mathrm{p}} \Rightarrow n \geq C\sqrt{p}$

- $M(n) = n^2$

$$M\left(C\sqrt{p}\right)/p = C^2 p/p = C^2$$

- This algorithm is perfectly scalable

Summary (1/3)

- Performance terms
  - Speedup
  - Efficiency
- Model of speedup
  - Serial component
  - Parallel component
  - Communication component

Summary (2/3)

- What prevents linear speedup?
  - Serial operations
  - Communication operations
  - Process start-up
  - Imbalanced workloads
  - Architectural limitations

Summary (3/3)

- Analyzing parallel performance
  - Amdahl's Law
  - Gustafson-Barsis' Law
  - Karp-Flatt metric
  - Isoefficiency metric

Exercise 7.11

$$f = \frac{\sigma(n)}{\sigma(n)+\varphi(n)} = \frac{1}{1+\frac{\varphi(n)}{\sigma(n)}}$$

$$\psi \le \frac{1}{f + (1-f)/p}$$

$$s = \frac{\sigma(n)}{\sigma(n)+\frac{\varphi(n)}{p}} = \frac{1}{1+\frac{1}{p}\frac{\varphi(n)}{\sigma(n)}}$$

$$\begin{aligned}\psi &\le p + (1-p)s \\ &= p(1-s)+s\end{aligned}$$

Exercises

- 5.9
- 6.12
- 7.10