



# Parallel Software Overview

## Lecture Objectives:

- 1) Define the terms load balancing, parallelization, embarrassingly parallel
- 2) Define the concepts of speedup and efficiency
- 3) Explain Amdahl's law
- 4) Calculate the maximum speedup that can be obtained when parallelizing an algorithm
- 5) Explain Foster's methodology and apply Foster's methodology to a simple problem

# A simple example

```
double x[100], y[100]
```

```
.  
.  
.  
for (int i = 0; i < 100; i++)  
{  
    x[i] += y[i];  
}
```

Overhead

49

2 processors

for (int i = 0; ...  
{  
 x[i] += y[i];

3

Modify this so that it is easily parallelized...

Overhead

# Definitions

- Parallelization
  - The process of converting a serial algorithm into a parallel algorithm
- Goals of parallelization
  - Roughly the same amount of work done by each processor
    - Aka load balancing
  - Minimize communication between processors  $\Rightarrow$  overhead/wasted time.
- "Embarrassingly Parallel"
  - Programs that can be parallelized by simply dividing the work between two processors

# Speedup and Efficiency

How much faster do we get? ⇒

Linear speedup

$$T_{Parallel} = \frac{T_{serial}}{p}$$

1 core  
processor

• Speedup Definition

$$S = \frac{T_{serial}}{T_{parallel}}$$

(Linear speedup)

• Efficiency

$$E = \frac{S}{p} = \frac{T_{serial} / T_{parallel}}{p} = \frac{T_{serial}}{p \cdot T_{parallel}}$$

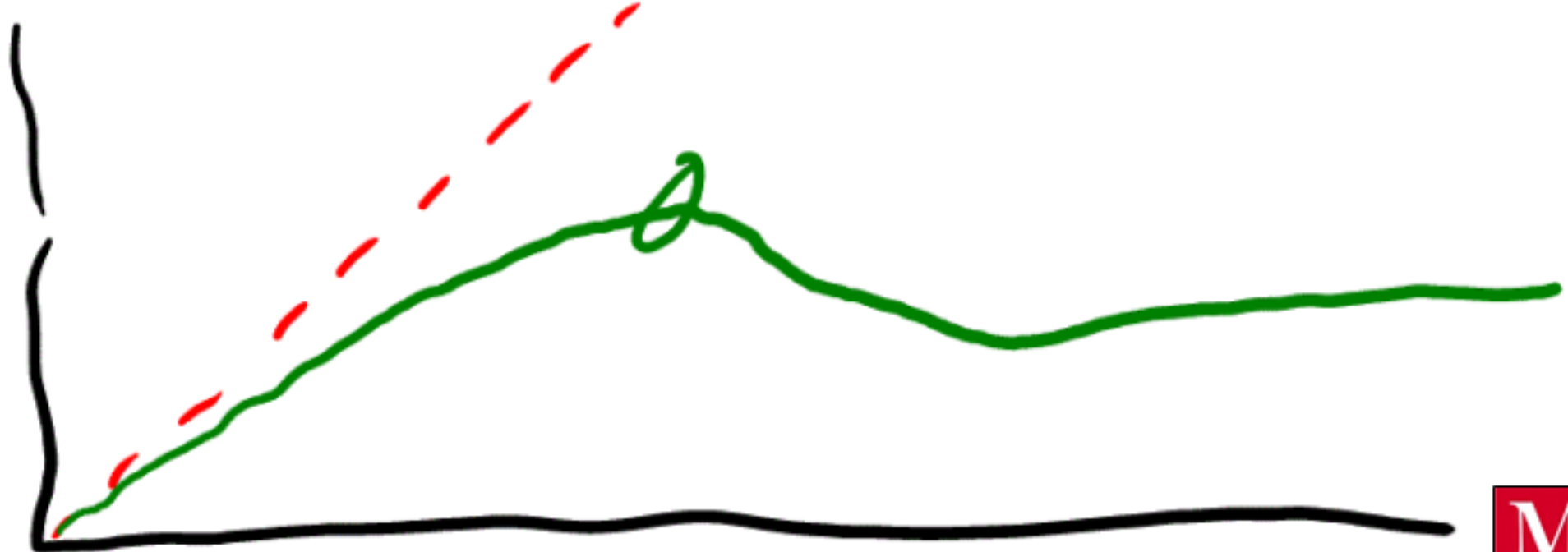
probably < 1



# Lets work some examples

T Serial	T Parallel	Number of Processors	Speedup	Effeciency
→ 100	→ 67	→ 2	1.49	.74
100	50	3	2	.66
100	45	4	2.22	.55

Minimum time




# Lets work some examples

T Serial	T Parallel	Number of Processors	Speedup	Effeciency
100	67	2	1.492537	0.746269
100	50	3	2	0.666667
100	45	4	2.222222	0.555556

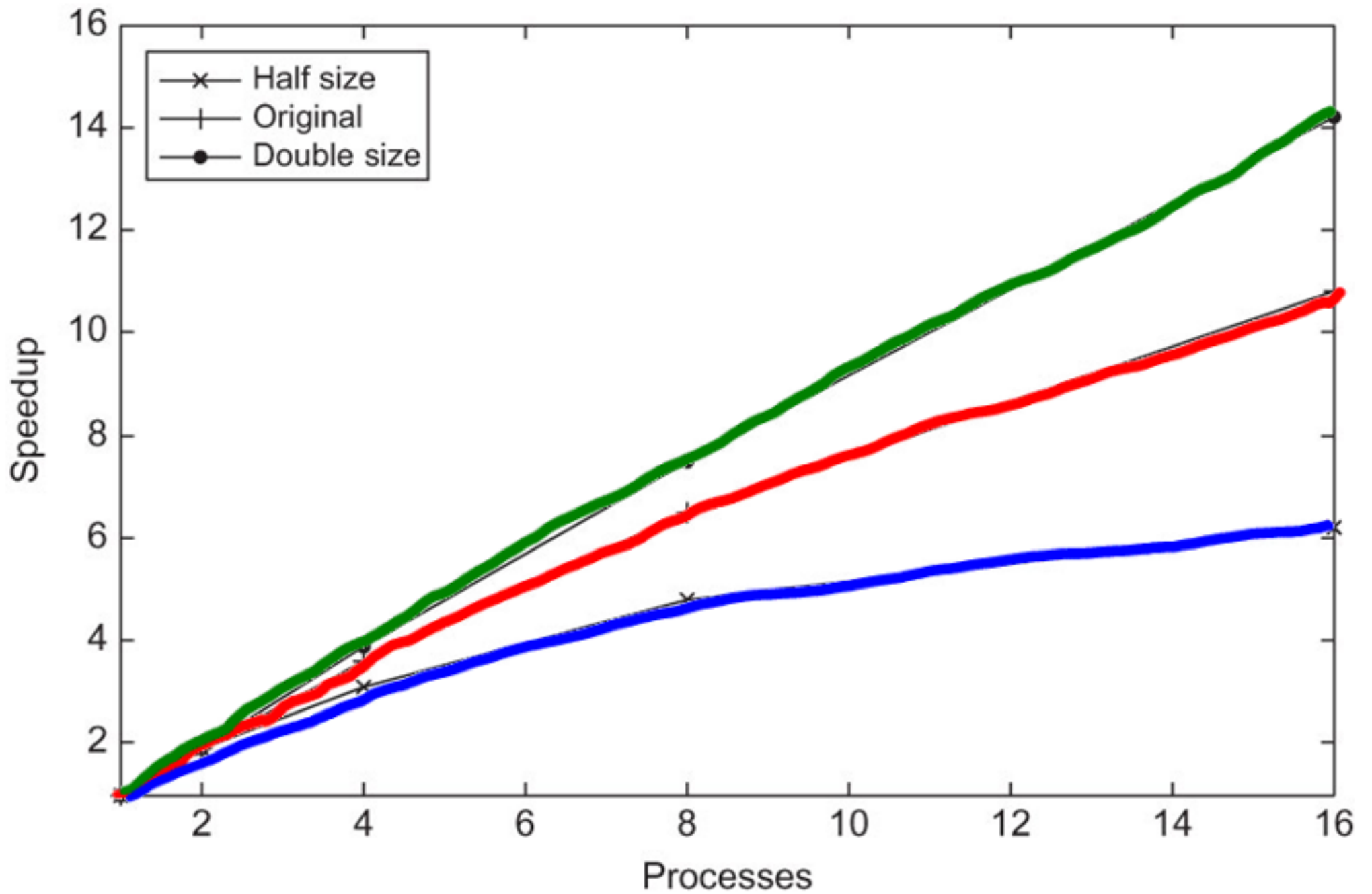
# What does this lead to?

- The ultimate equation

$$T_{parallel} = \frac{T_{serial}}{p} - T_{overhead}$$


Extra resources  
of parallelizing

# In the real world

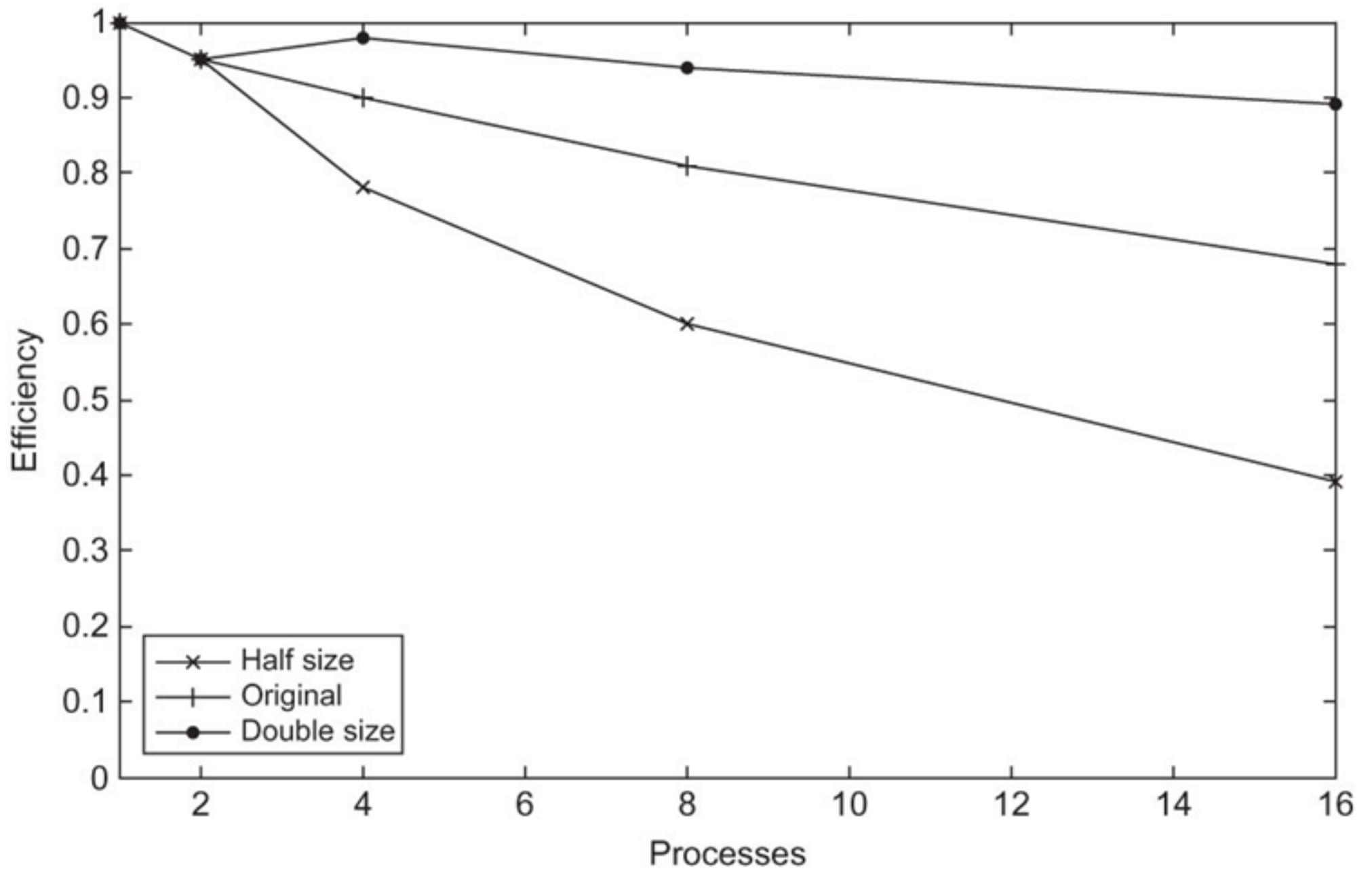


1-20K

1-5K



# In the real world



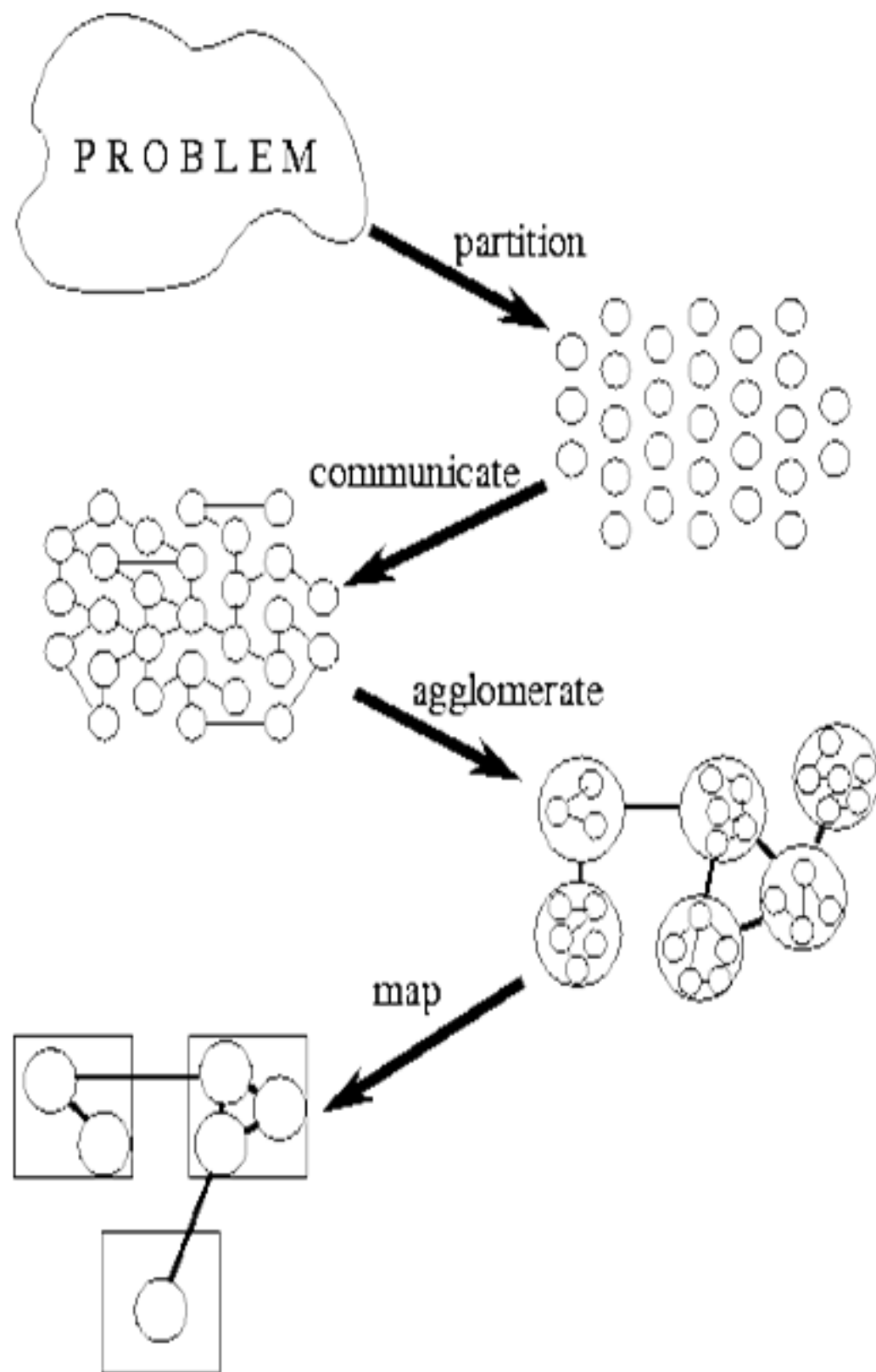
*decreases*

# Amdahl's Law

- The speedup of a program will be limited by how much of the program can be parallelized.
  - Example: A program takes 100 seconds to run and 80% of the code can be parallelized. What is the minimum time it will take to execute the program?

*Minimum would be  
20 s*

# Foster's Methodology



- **Partitioning**
  - The computation that is to be performed and the data operated on by this computation are decomposed into small tasks. Practical issues such as the number of processors in the target computer are ignored, and attention is focused on recognizing opportunities for parallel execution.
- **Communication**
  - The communication required to coordinate task execution is determined, and appropriate communication structures and algorithms are defined.
- **Agglomeration**
  - The task and communication structures defined in the first two stages of a design are evaluated with respect to performance requirements and implementation costs. If necessary, tasks are combined into larger tasks to improve performance or to reduce development costs.
- **Mapping**
  - Each task is assigned to a processor in a manner that attempts to satisfy the competing goals of maximizing processor utilization and minimizing communication costs. Mapping can be specified statically or determined at runtime by load-balancing algorithms.

# Example Problem

- Lets assume we want to write a program which will layout music for printing
  - A song is composed of measures, notes, and rests.
  - We want to figure out how to best layout the music pages and render them to a pdf file for printing
- How might we do this?

