

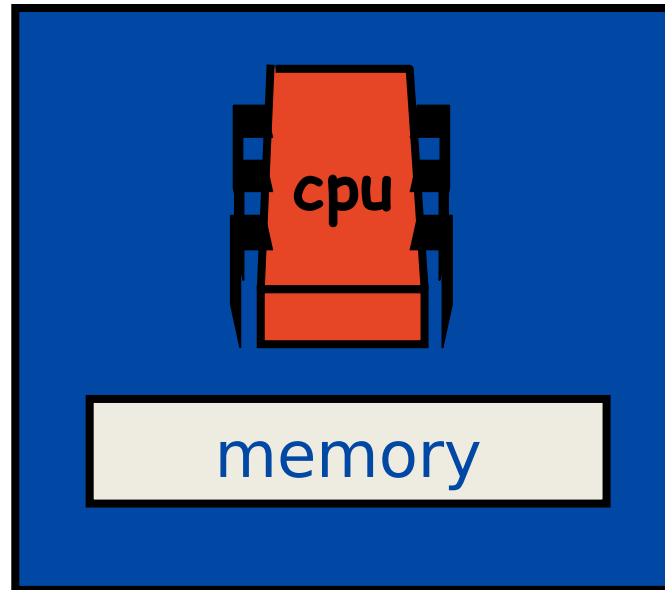
Chip Multi-Processor



THE UNIVERSITY OF
SYDNEY

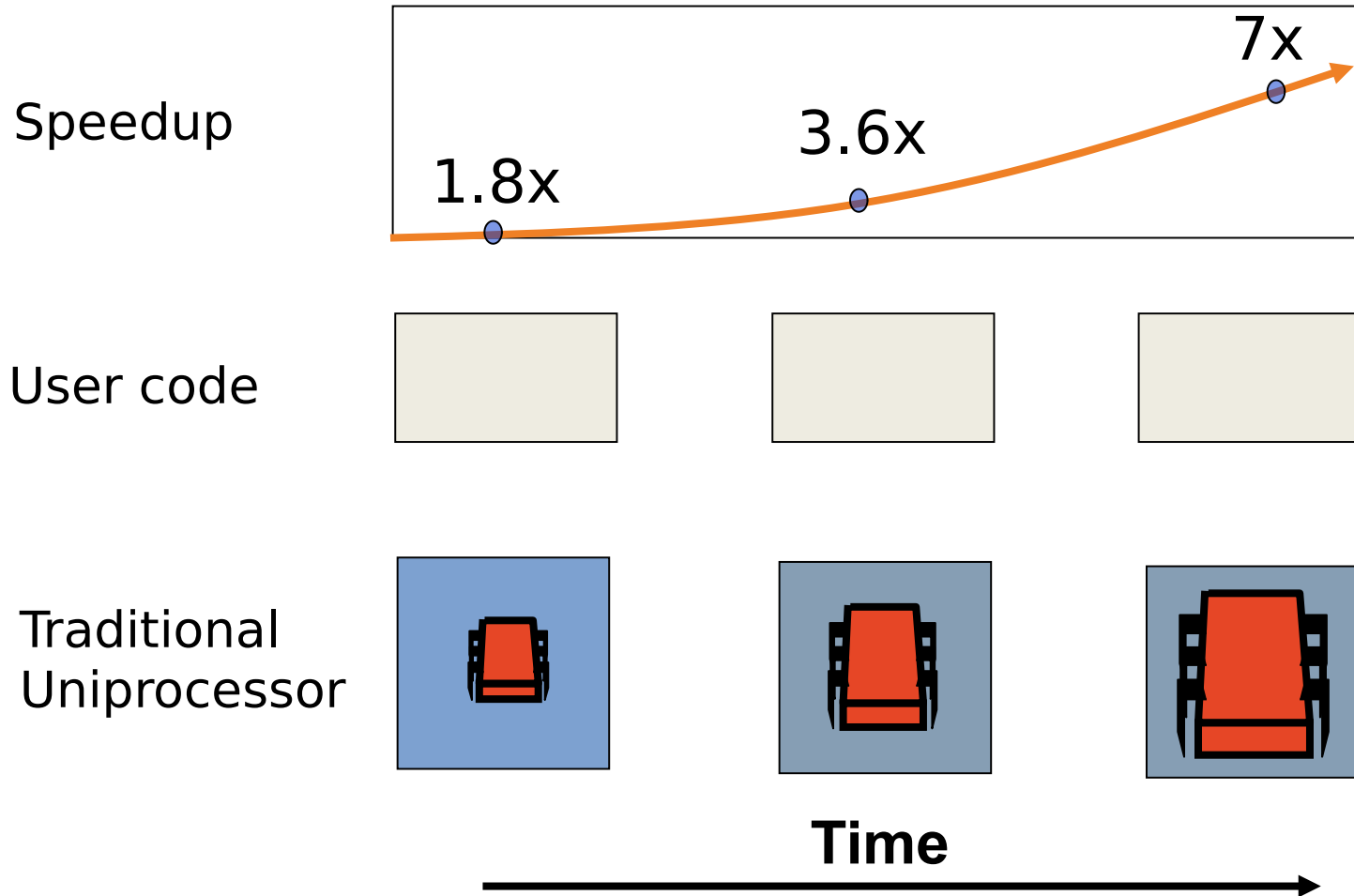
Traditional Uniprocessor

Single Central Processing Unit (CPU), single memory



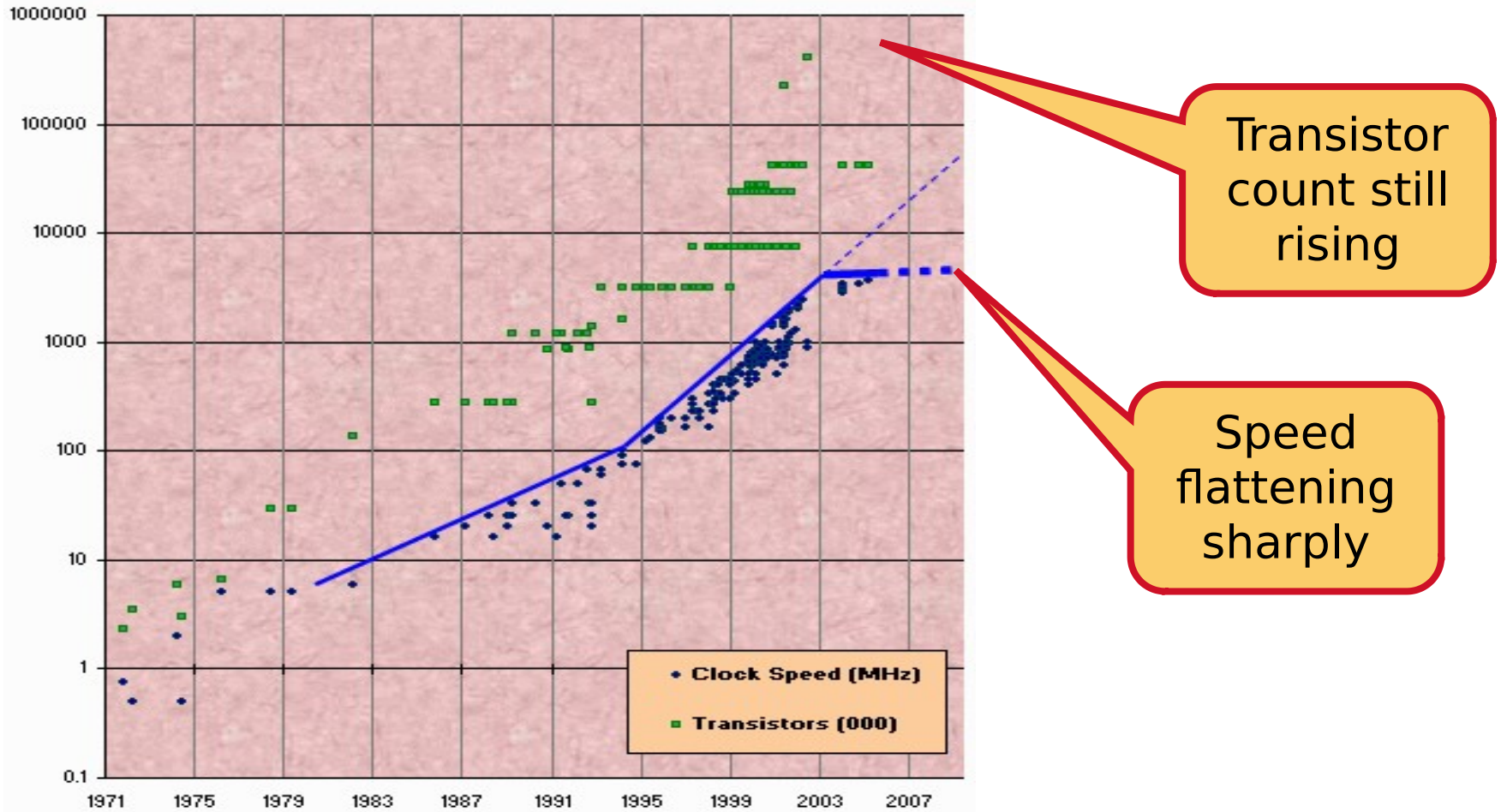
Performance Improvement

Moore's law: *# transistors per chip doubles every 2.5 years*



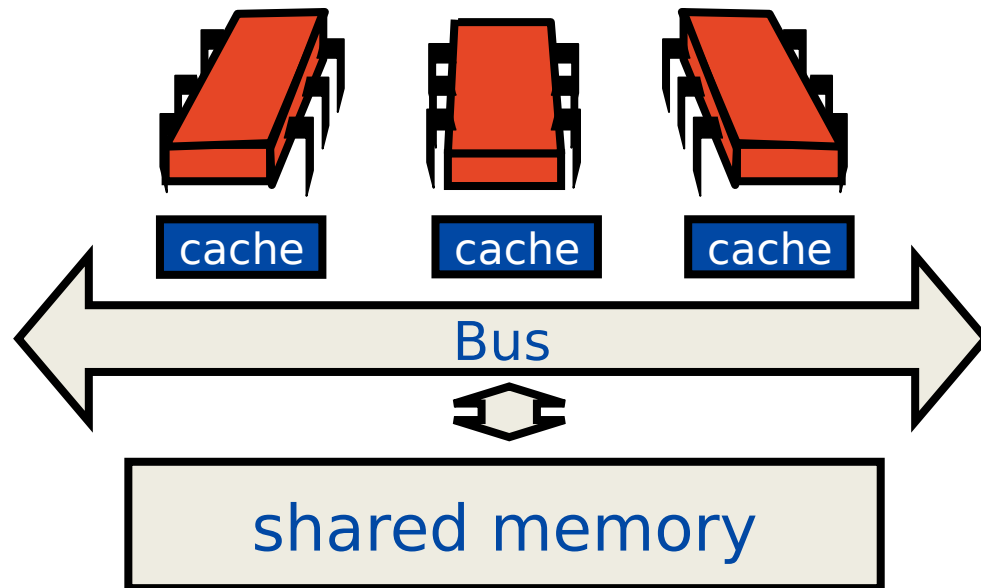
The Free Lunch is Over

But performance no longer increases with # transistors



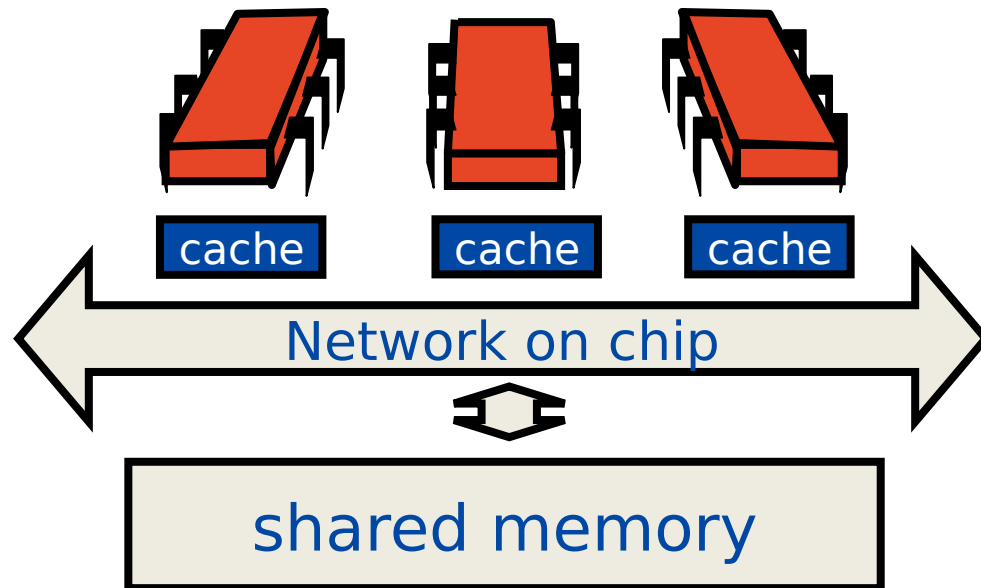
Multiprocessor machine

Symmetric Multi-Processor (SMP)



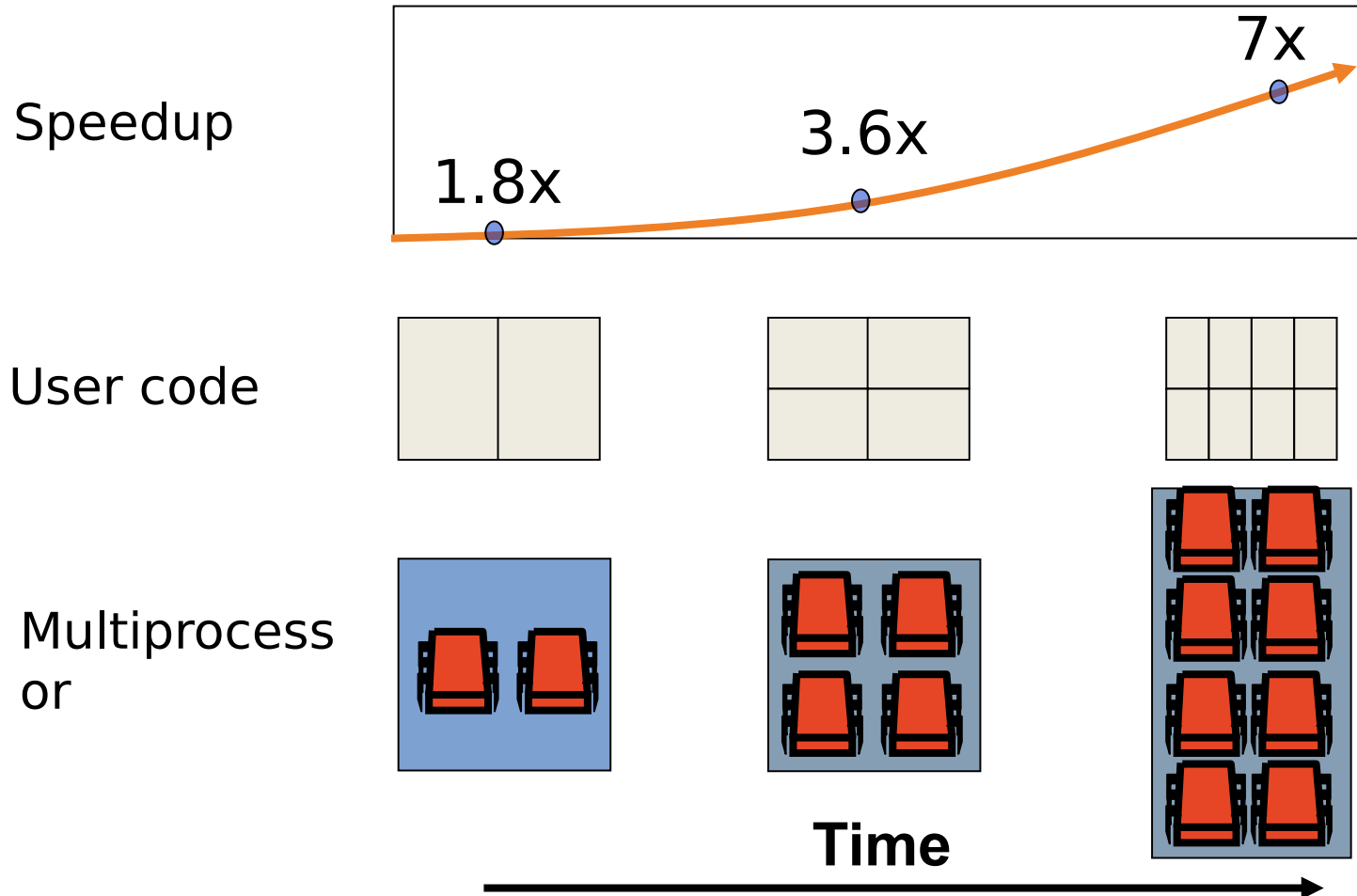
Multicore machine

Chip Multi-Processor (CMP)



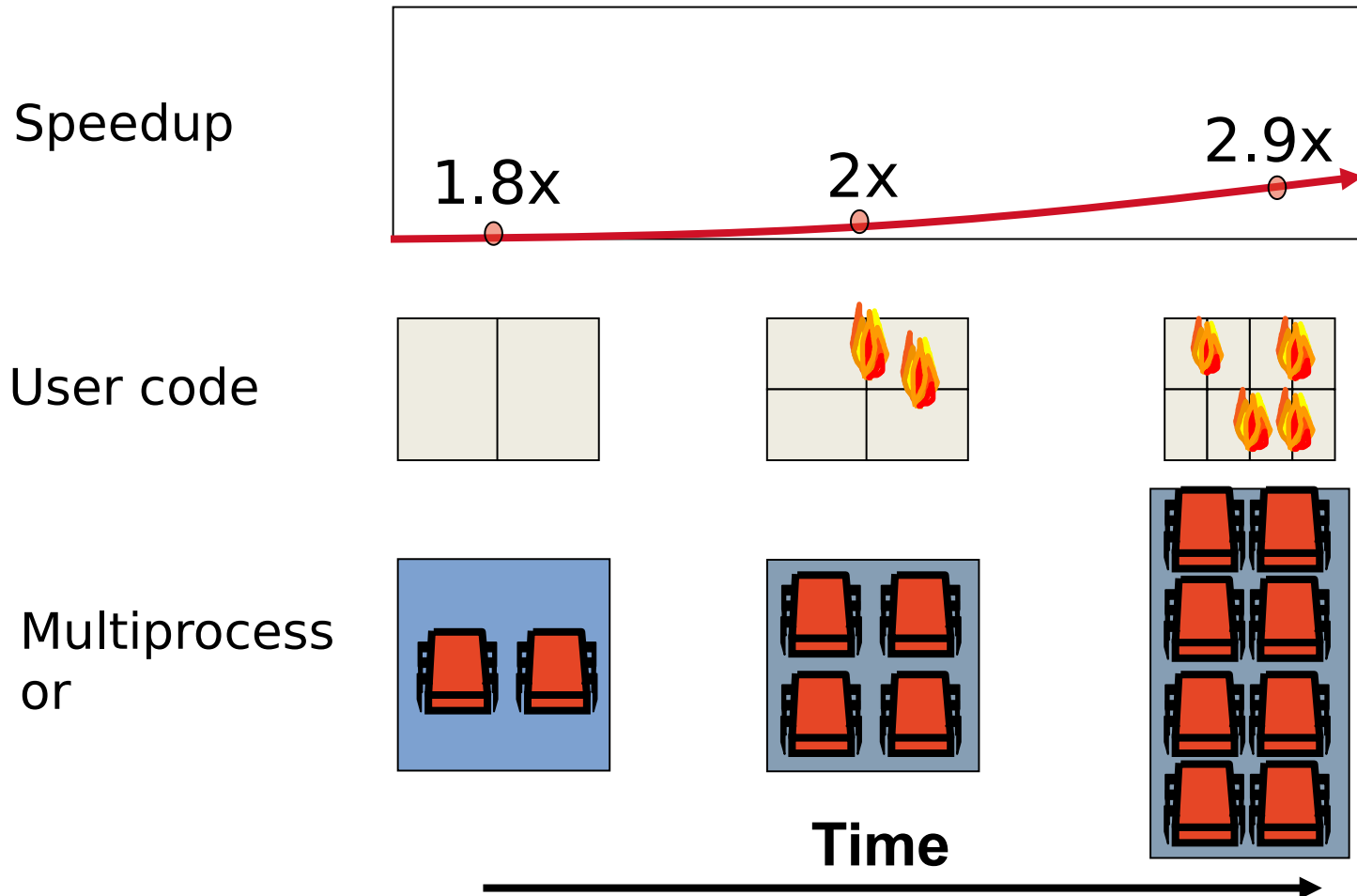
Performance Improvement requires Concurrency

The trend is to multiply the number of CPUs rather than frequency



Not in Reality

Problem: parallelisation and synchronisation require great care...



Amdahl's law



THE UNIVERSITY OF
SYDNEY

Amdahl's Law

Speedup=

$$\frac{\text{OldExecutionTime}}{\text{NewExecutionTime}}$$

...of computation given n CPUs instead of **1**

Amdahl's Law

Speedup=

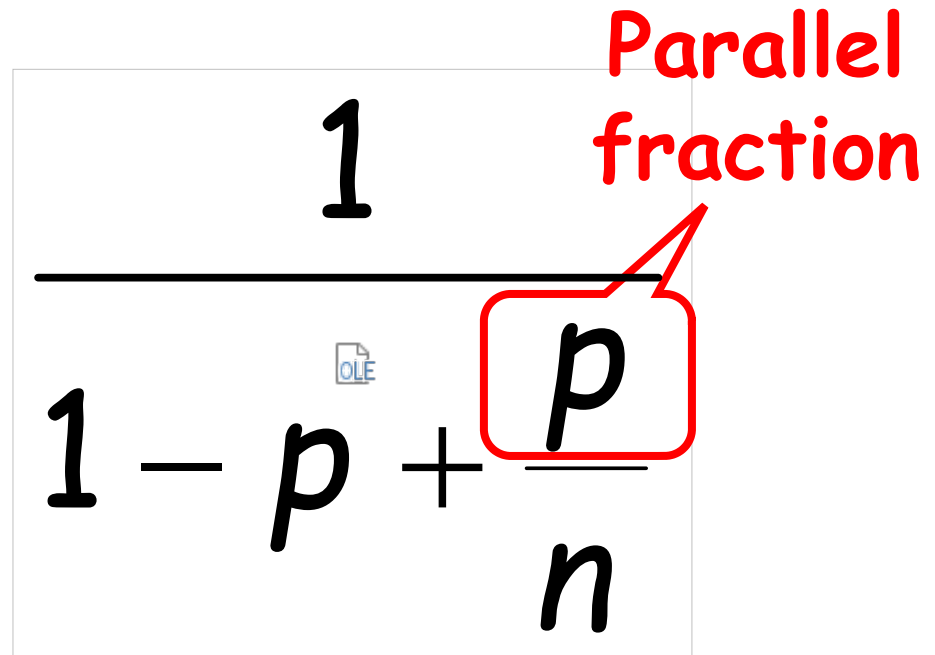
$$\frac{1}{1 - p + \frac{p}{n}}$$

Amdahl's Law

Speedup=

$$\frac{1}{1 - p + \frac{p}{n}}$$

Parallel fraction

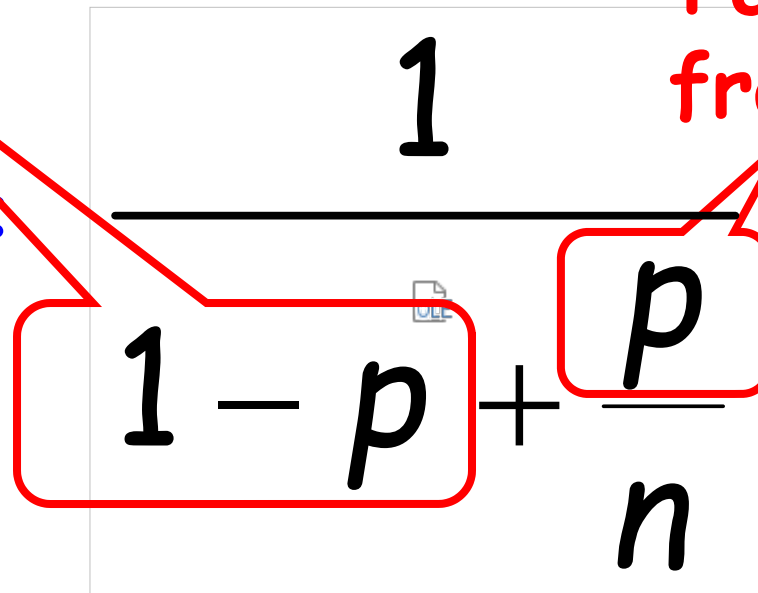
The diagram shows the Amdahl's Law formula: Speedup = 1 / (1 - p + p/n). The variable 'p' in the denominator is enclosed in a red box. A red arrow points from the text 'Parallel fraction' to this boxed 'p'. There is also a small 'OLE' icon above the plus sign in the denominator.

Amdahl's Law

Sequential
fraction

Speedup =

Parallel
fraction



The diagram shows the Amdahl's Law formula:
$$\text{Speedup} = \frac{1}{1 - p + \frac{p}{n}}$$
 The formula is enclosed in a light gray box. Red annotations highlight parts of the formula: a red box around $1 - p$ is pointed to by a red arrow from the text 'Sequential fraction'; a red box around $\frac{p}{n}$ is pointed to by a red arrow from the text 'Parallel fraction'. The word 'Speedup =' is written in blue to the left of the fraction.

Amdahl's Law

Sequential
fraction

Speedup =

Parallel
fraction

$$\frac{1}{1 - p + \frac{p}{n}}$$

Number of
processors

Example

- Ten processors
- 60% concurrent, 40% sequential
- How close to 10-fold speedup?

Example

- Ten processors
- 60% concurrent, 40% sequential
- How close to 10-fold speedup?

Speedup=2.17=

$$\frac{1}{1 - 0.6 + \frac{0.6}{10}}$$

Example

- Ten processors
- 80% concurrent, 20% sequential
- How close to 10-fold speedup?

Example

- Ten processors
- 80% concurrent, 20% sequential
- How close to 10-fold speedup?

$$\text{Speedup} = 3.57 =$$

$$\frac{1}{1 - 0.8 + \frac{0.8}{10}}$$

Example

- Ten processors
- 90% concurrent, 10% sequential
- How close to 10-fold speedup?

Example

- Ten processors
- 90% concurrent, 10% sequential
- How close to 10-fold speedup?

Speedup=5.26=

$$\frac{1}{1 - 0.9 + \frac{0.9}{10}}$$

Example

- Ten processors
- 99% concurrent, 01% sequential
- How close to 10-fold speedup?

Example

- Ten processors
- 99% concurrent, 01% sequential
- How close to 10-fold speedup?

Speedup=9.17=

$$\frac{1}{1 - 0.99 + \frac{0.99}{10}}$$

The Moral

- Making good use of our multiple processors (cores) means
- Finding ways to effectively parallelise our code
 - Minimize sequential parts
 - Reduce idle time in which threads **wait**

Multicore Programming

- This is what this course is about...
 - The % that is not easy to make concurrent yet may have a large impact on overall speedup
- Next week:
 - A more serious look at mutual exclusion