GPU Programming

How fast can we get?

Christian Lessig

Quantitative performance measures

- Latency: time to completion of task
- Throughput: tasks processed per second
- Power consumption

Quantitative measures

$$\text{speedup} = \frac{\text{execution time for serial execution}}{\text{execution time for parallel execution}}$$

Quantitative measures

$$\text{efficiency} = \frac{\text{speedup}}{\#\text{processors}}$$

Quantitative measures

$$\text{cost} = |\text{time}| \times \#\text{processors}$$

Amdahl's law

$$\text{speedup} = \frac{\text{execution time for serial execution}}{\text{execution time for parallel execution } \textbf{\textit{when}} \text{ possible}}$$

J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach, fourth edition, Morgan Kaufmann, 2007; p. 40.

Amdahl's law

$$S = \frac{1}{(1-p) + p/K}$$

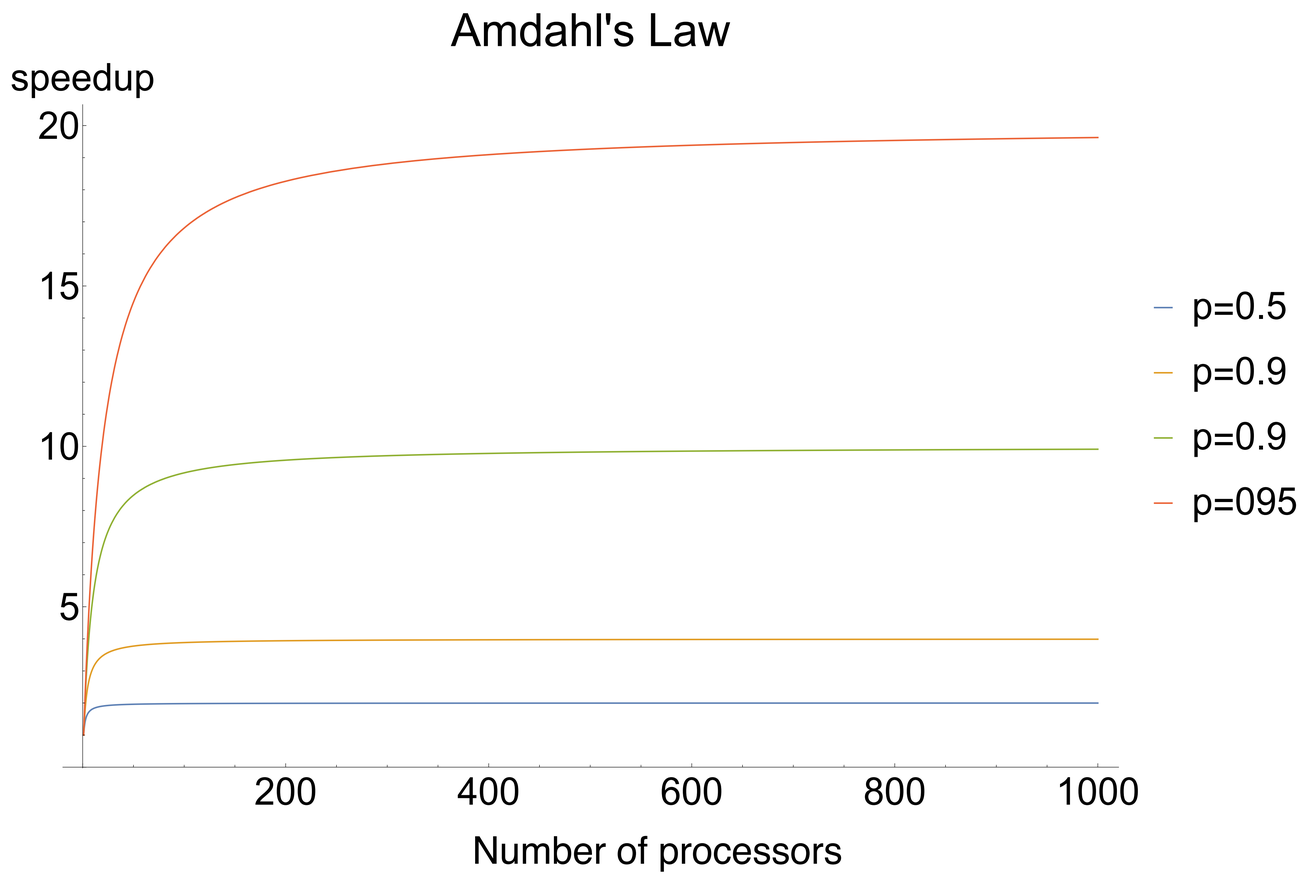
Amdahl's law

$$S = \frac{1}{(1-p) + p/K}$$

acceleration: 1/#P

Fraction of parallelizable code

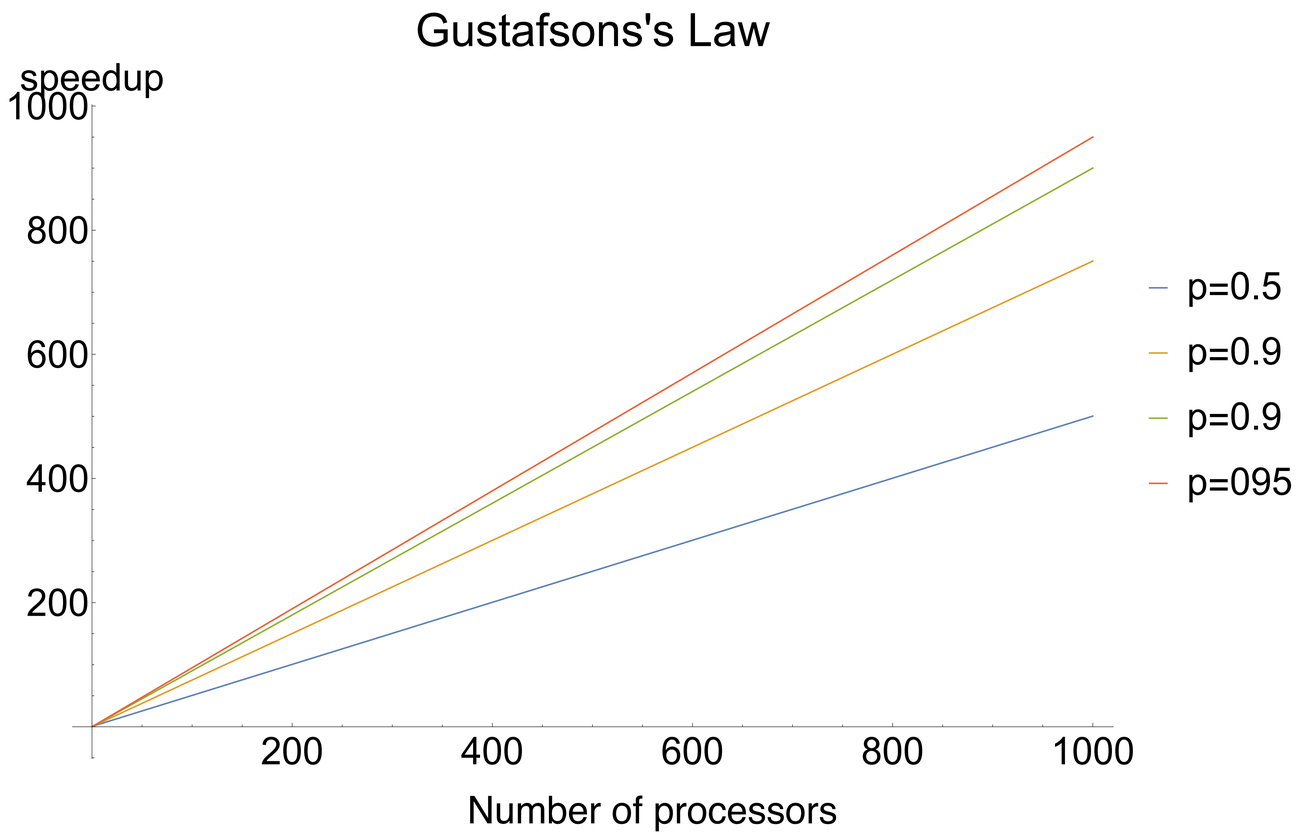
Amdahl’s law



Gustafson's law

$$S_G = (1 - p) + p \cdot K$$

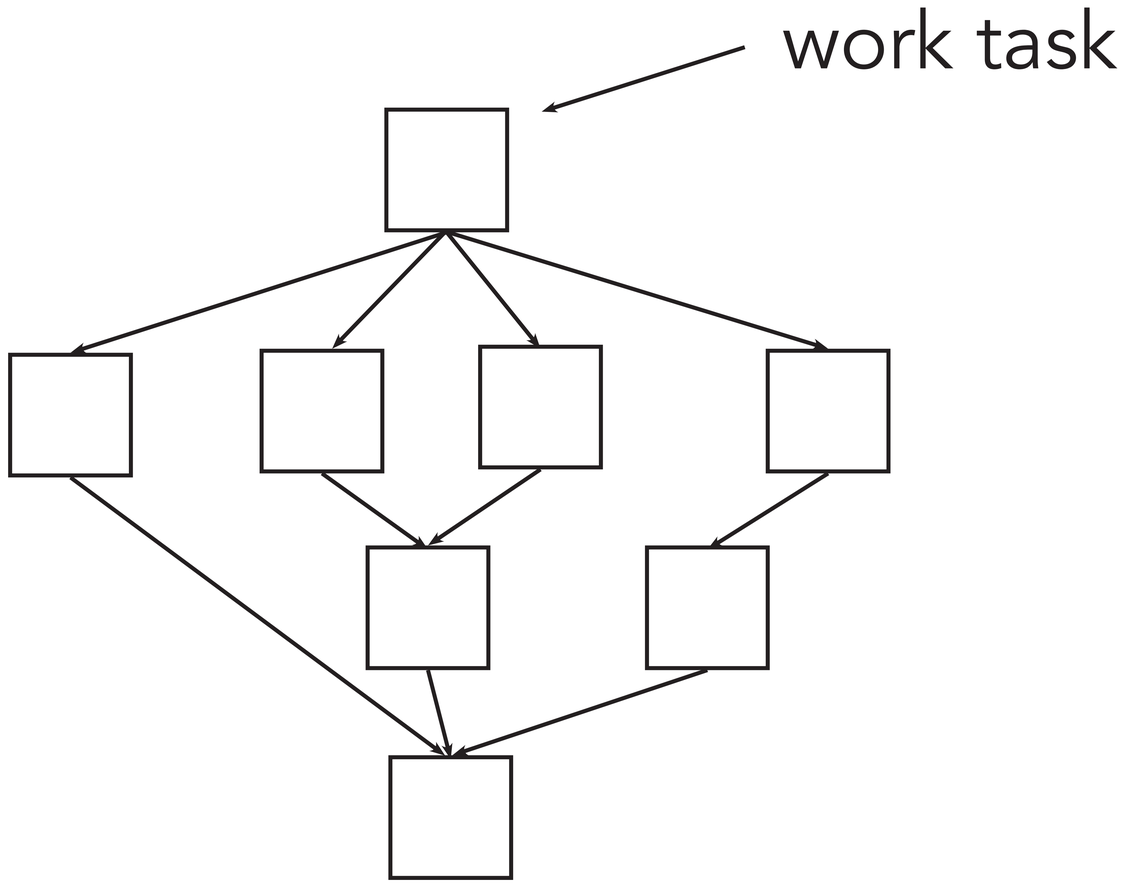
Gustafson’s law



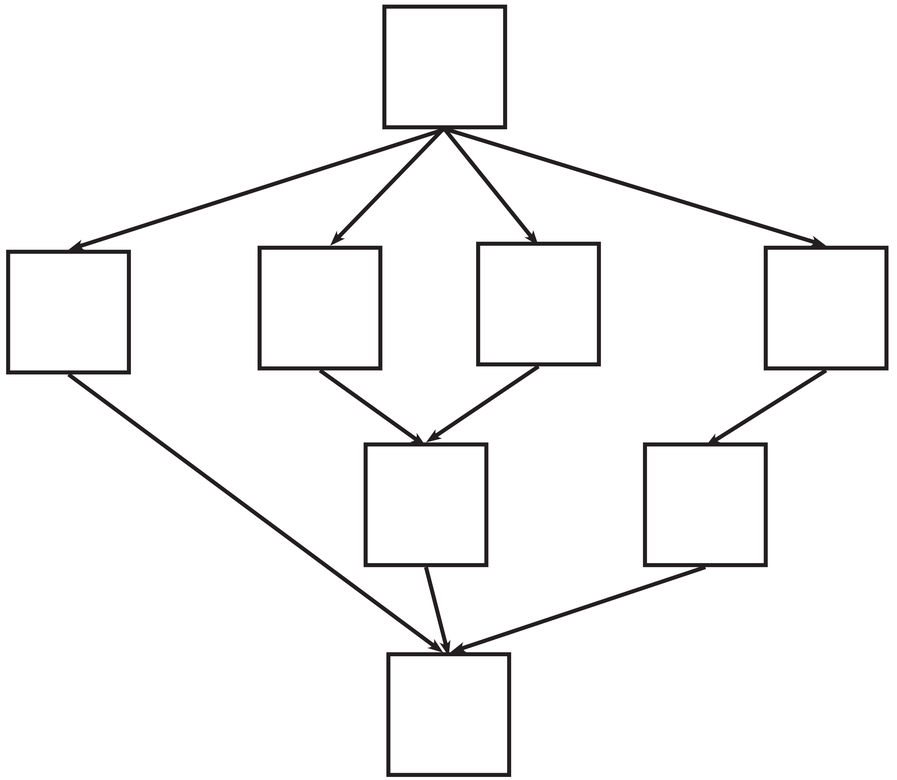
Can we do better?

Performance Analysis

DAG models program



Performance Analysis

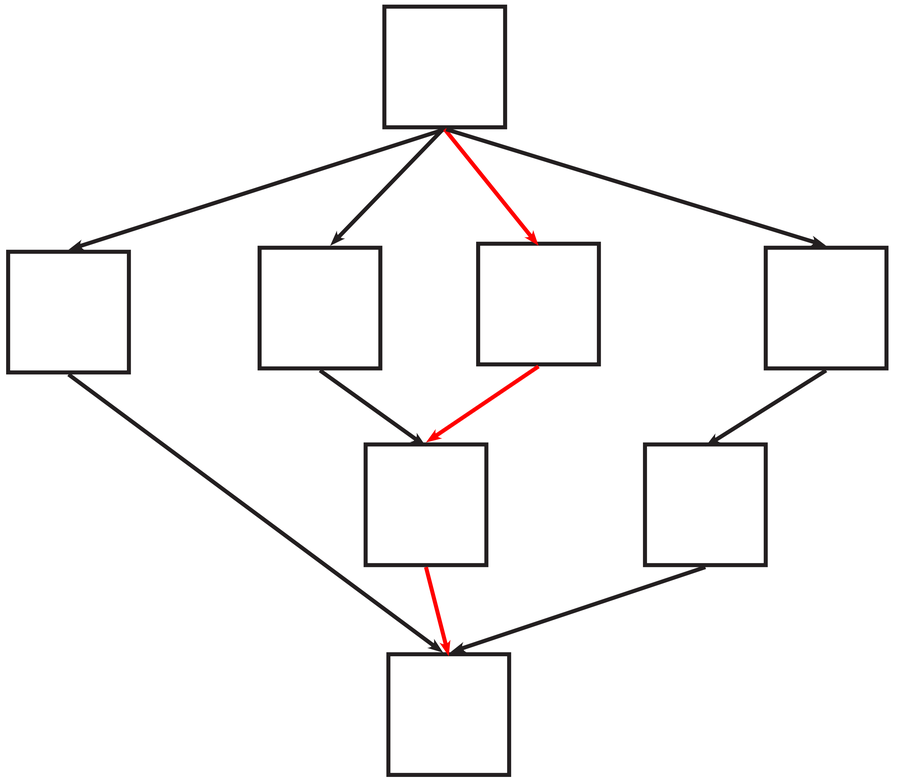


serial

parallel

serial

Work-Span Analysis

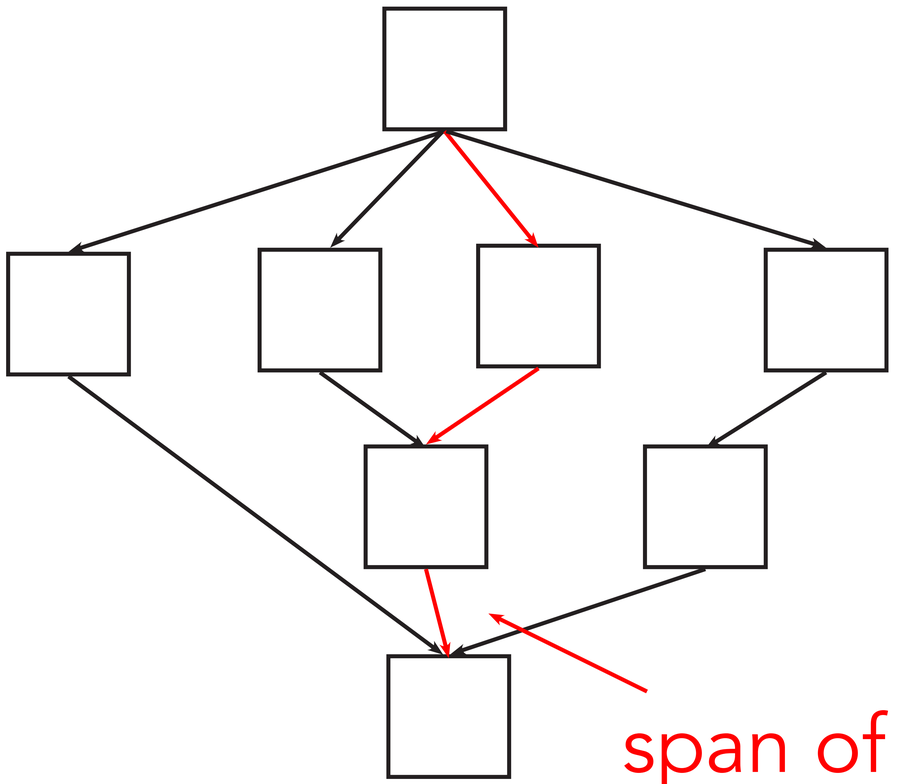


serial

parallel

serial

Work-Span Analysis



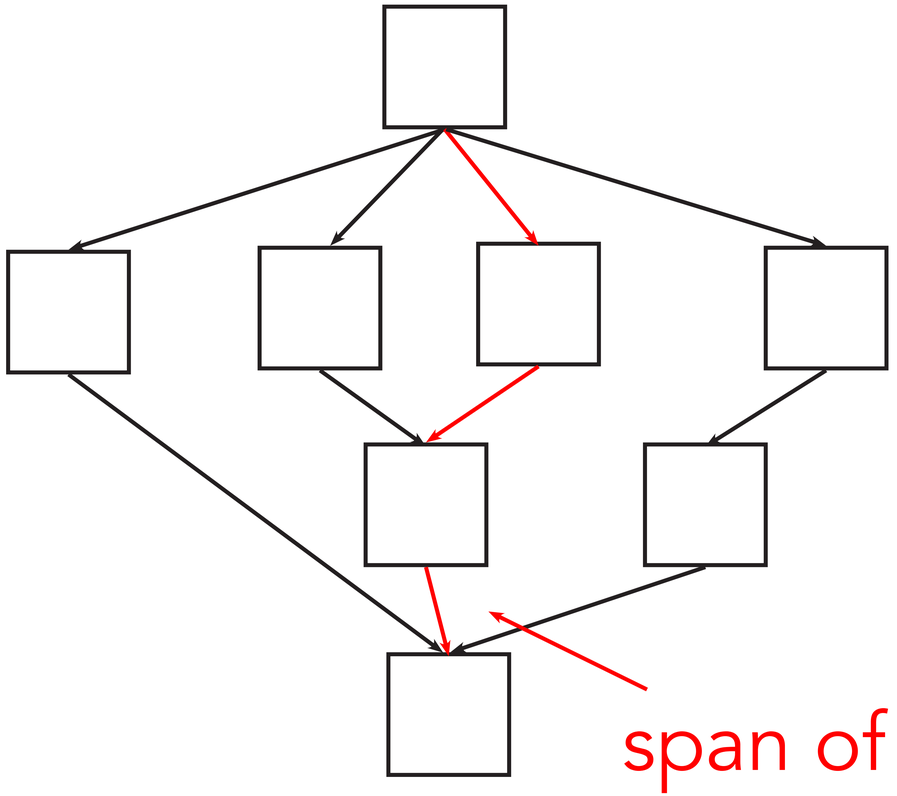
serial

parallel

serial

Work-Span Analysis

$$T_{\min} = 4$$



serial

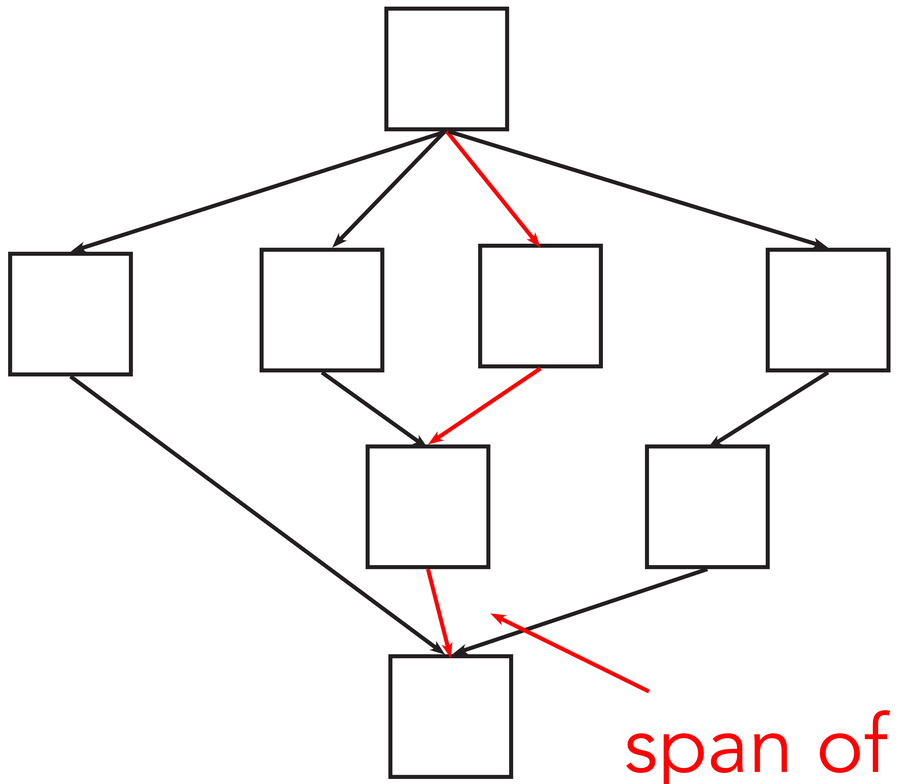
parallel

serial

Work-Span Analysis

$$T_{\min} = 4$$

$$S_{\max} = 8/4 = 2$$



serial

parallel

serial

Work-Span Analysis

Serial execution time:

$$T_1 = \#\text{work}$$

Work-Span Analysis

Serial execution time:

$$T_1 = \#\text{work}$$

Minimal parallel execution time:

$$T_\infty = \text{span}$$

Work-Span Analysis

Serial execution time:

$$T_1 = \#\text{work}$$

Minimal parallel execution time:

$$T_\infty = \text{span}$$

Minimal execution time with P threads:

$$T_P = \#\text{work}/P$$

Work-Span Analysis

Serial execution time:

$$T_1 = \#\text{work}$$

Minimal parallel execution time:

$$T_\infty = \text{span}$$

Minimal execution time with P threads:

$$T_P = \#\text{work}/P$$

But we cannot be faster than T_∞!

Work-Span Analysis

Serial execution time:

$$T_1 = \#\text{work}$$

Minimal parallel execution time:

$$T_\infty = \text{span}$$

Minimal execution time with P threads:

$$T_P = \max\left(\#\text{work}/P\,,\, T_\infty\right)$$

Work-Span Analysis

Serial execution time:

$$T_1 = \#\text{work}$$

Minimal parallel execution time:

$$T_\infty = \text{span}$$

Optimal speedup

$$S_\infty = \frac{T_1}{T_\infty}$$

Minimal execution time with P threads:

$$T_P = \max\left(\#\text{work}/P\,,\, T_\infty\right)$$

Work-Span Analysis

Serial execution time:

$$T_1 = \#\text{work}$$

Minimal parallel execution time:

$$T_\infty = \text{span}$$

Minimal execution time with P threads:

$$T_P = \max\left(\#\text{work}/P\,,\, T_\infty\right)$$

speedup with P threads

$$S_P = \min(P, S_\infty)$$

Work-Span Analysis

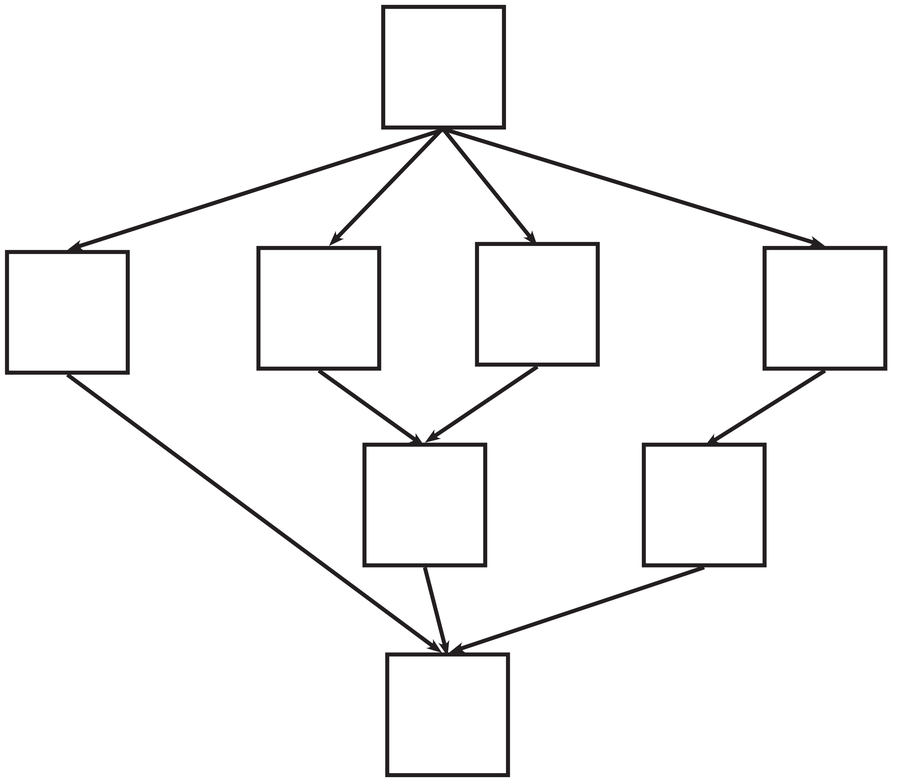
Optimal execution time
with P threads:

$$T_P = \max\left(\#\text{work}/P\,,\, T_\infty\right)$$

Work-Span Analysis

Optimal execution time
with P threads:

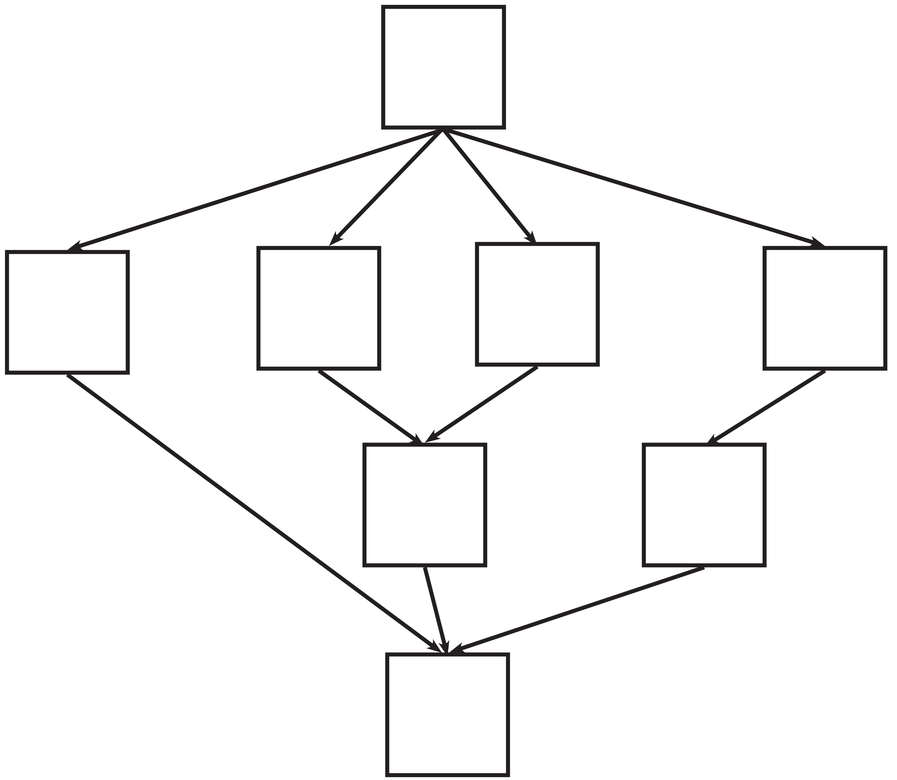
$$T_P = \max\left(\#\text{work}/P\,,\, T_\infty\right)$$



Work-Span Analysis

Optimal execution time
with P threads:

$$T_P = \max\left(\#\text{work}/P\,,\, T_\infty\right)$$



At each of the T_∞
steps we need time

$$T_i = W_i/P$$

Work-Span Analysis

Brent's theorem:

$$T_P \leq T_\infty + \frac{T_1 - T_\infty}{P}$$

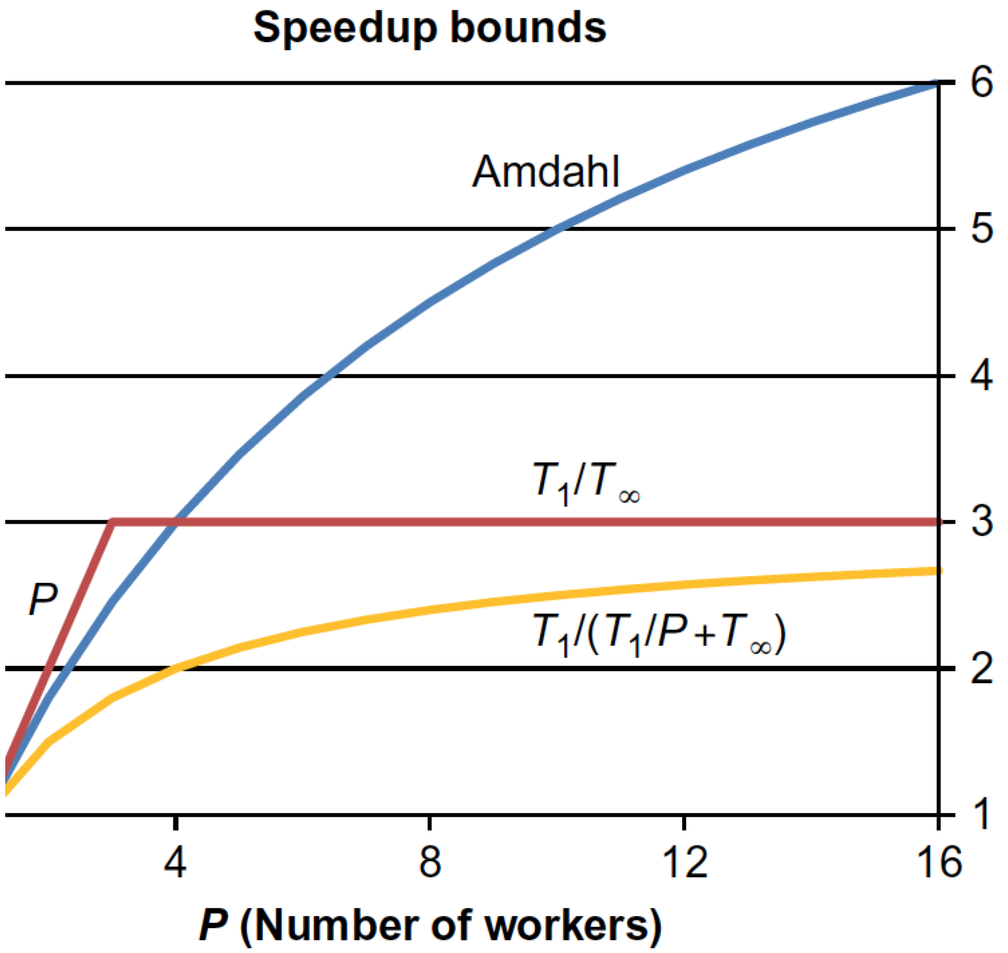
Work-Span Analysis

Brent's theorem:

$$T_P \leq T_\infty + \frac{T_1 - T_\infty}{P}$$

Upper bound on execution time

Work-Span Analysis



M. D. McCool, J. Reinders, and A. Robison, Structured parallel programming: patterns for efficient computation. Elsevier/Morgan Kaufmann, 2012.

Further reading

- D. Kirk and W. -m. Hwu, Programming massively parallel processors. Elsevier/Morgan Kaufmann, 2013, Ch. 1.
- M. D. McCool, J. Reinders, and A. Robison, Structured parallel programming : patterns for efficient computation. Elsevier/Morgan Kaufmann, 2012.
- T. Kielmann et al., Encyclopedia of Parallel Programming. Springer, 2011.