Parallel Programming: Speedups and Amdahl's law
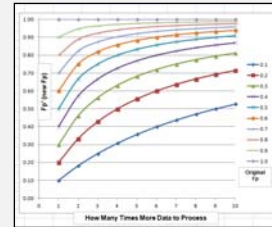
1
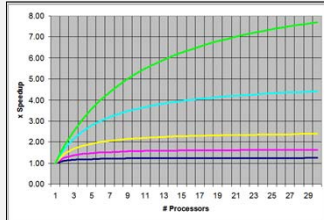
Oregon State
University

Mike Bailey

mjb@cs.oregonstate.edu

Oregon State
University
Computer Graphics

speedups.and.amdahls.law.pptx   mjb – March 15, 2020

Definition of Speedup

2

If you are using **n** processors, your **Speedup$_n$** is:

$$Speedup_n = \frac{T_1}{T_n}$$

where T_1 is the execution time on **one core**
and T_n is the execution time on **n cores**.
Note that Speedup$_n$ should be > 1.

And your **Speedup Efficiency$_n$** is:

$$Efficiency_n = \frac{Speedup_n}{n}$$

which could be as high as 1., but probably never will be.

Oregon State
University
Computer Graphics

mjb – March 15, 2020

**However, Multicore is not a Free Lunch:**
Amdahl's Law

3

If you put in *n* processors, you should get *n* times Speedup
(and 100% Speedup Efficiency), right? Wrong!

There are always some fraction of the total operation that is inherently
sequential and cannot be parallelized no matter what you do. This includes
reading data, setting up calculations, control logic, storing results, etc.

If you think of all the operations that a program needs to do as being
divided between a fraction that is parallelizable and a fraction that isn't
(i.e., is stuck at being sequential), then **Amdahl's Law** says:

$$Speedup_n = \frac{T_1}{T_n} = \frac{1}{\frac{F_{parallel}}{n} + F_{sequential}} = \frac{1}{\frac{F_{parallel}}{n} + (1 - F_{parallel})}$$
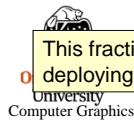
This fraction can be reduced by deploying multiple processors.

This fraction can't.

University
Computer Graphics

mjb – March 15, 2020

A Visual Explanation of Amdahl's Law

4

Execution Time

Sequential
Portion

Parallel
Portion

Sequential
Portion

Parallel
Portion

Sequential
Portion

Parallel
Portion

Sequential
Portion

The Sequential Portion doesn't go
away, and it also doesn't get any
smaller. It just gets more and
more dominant.

1   2   4   ~∞

# of Cores

Oregon State
University
Computer Graphics
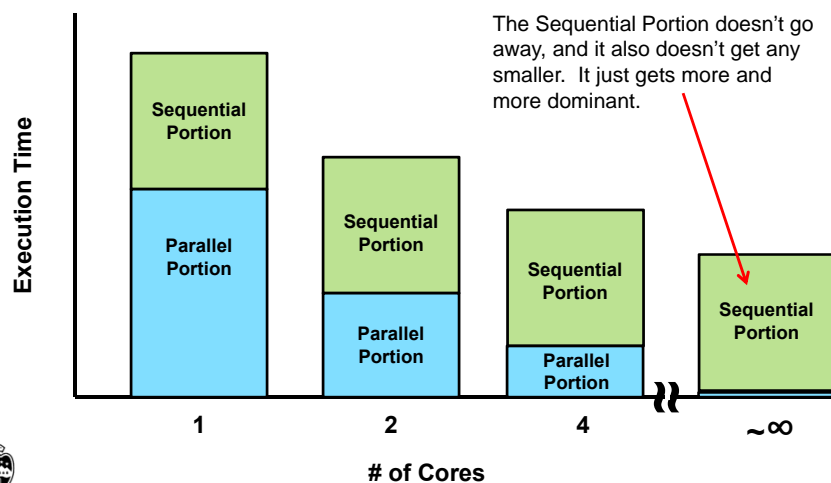
mjb – March 15, 2020

Amdahl's Law as a Function of Number of Processors and $F_{parallel}$
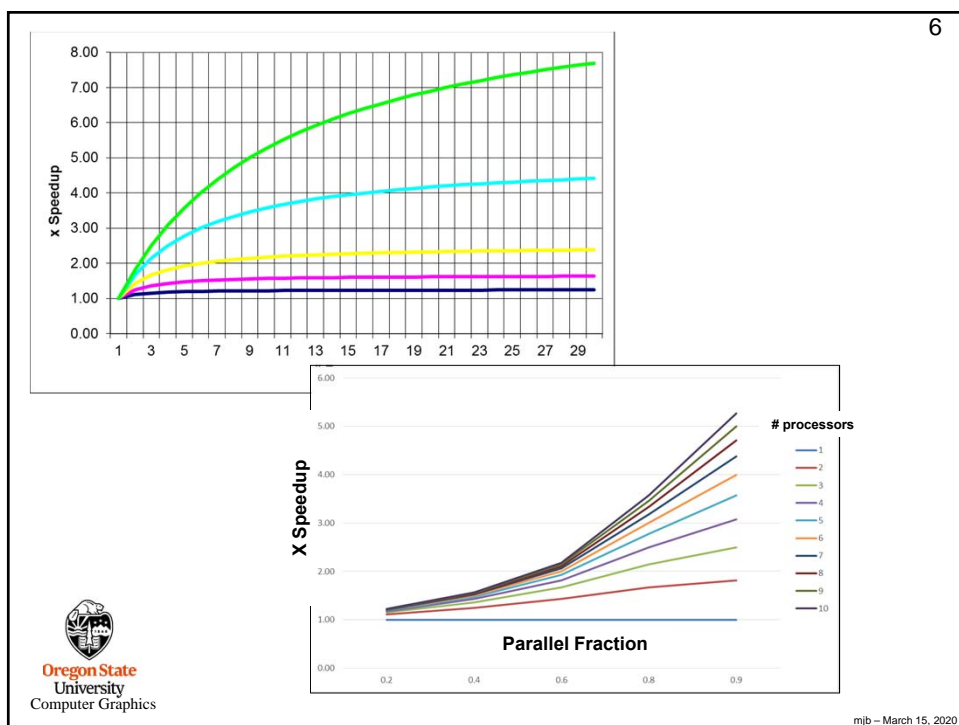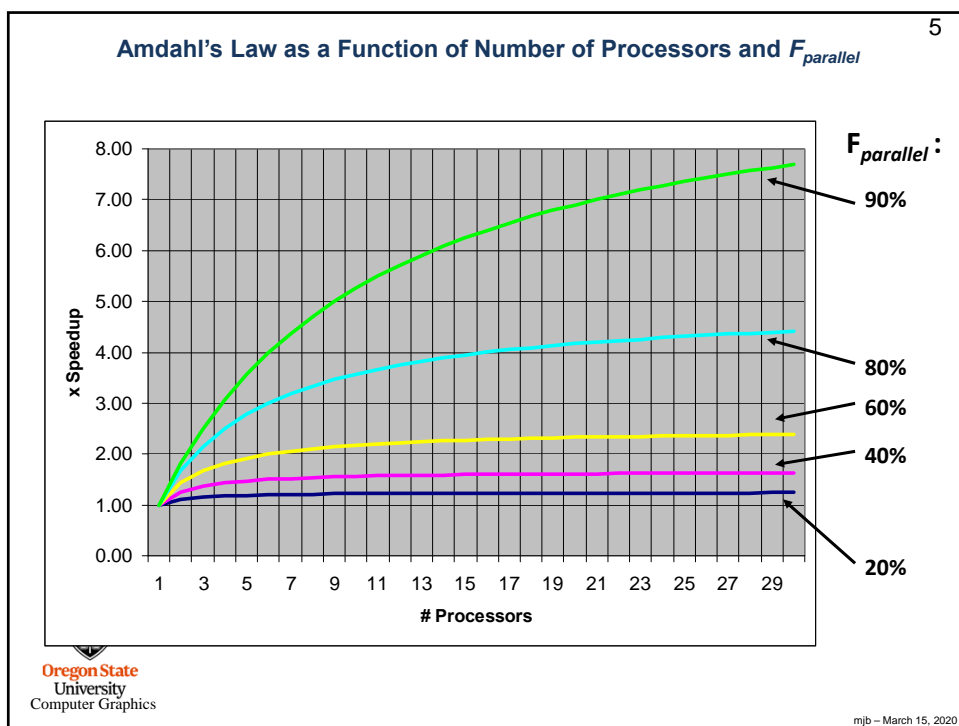
Amdahl's Law

Note that these fractions put an upper bound on how much benefit you will get from adding more processors:

$$\max Speedup = \lim_{n \to \infty} Speedup = \frac{1}{F_{sequential}} = \frac{1}{1 - F_{parallel}}$$

Fparallel	maxSpeedup
0.00	1.00
0.10	1.11
0.20	1.25
0.30	1.43
0.40	1.67
0.50	2.00
0.60	2.50
0.70	3.33
0.80	5.00
0.90	10.00
0.95	20.00
0.99	100.00

Oregon State
University
Computer Graphics

mjb – March 15, 2020

You can also solve for $F_{parallel}$ using Amdahl's Law if you know your speedup and the number of processors

Amdahl's law says:

$$S = \frac{T_1}{T_n} = \frac{1}{\frac{F}{n} + (1-F)} \implies \frac{1}{S} = \frac{F}{n} + (1-F) = 1 + \frac{F - nF}{n} \implies \frac{1}{S} - 1 = F\frac{(1-n)}{n}$$

Use this if you know the timing

Use this if you know the speedup

Solving for F:

$$F = \frac{\frac{1}{S} - 1}{\frac{1-n}{n}} = \frac{\frac{T_n}{T_1} - 1}{\frac{1-n}{n}} = \frac{\frac{T_n - T_1}{T_1}}{\frac{1-n}{n}} = \frac{\frac{T_1 - T_n}{T_1}}{\frac{n-1}{n}} = \frac{n(T_1 - T_n)}{T_1(n-1)} = \frac{n}{(n-1)}\frac{T_1 - T_n}{T_1} = \frac{n}{(n-1)}\left(1 - \frac{1}{Speedup}\right)$$

If you've got several (n,S) values, you can take the average (which is actually a least squares fit):

$$F_i = \frac{n_i}{(n_i - 1)}\frac{T_1 - T_{n_i}}{T_1}, i = 2..N$$

$$\bar{F} = \frac{\sum_{i=2}^{N} F_i}{N-1}$$

note that when i=1, $T_{ni} = T_1$

Oregon State
University
Computer Graphics

mjb – March 15, 2020

A More Optimistic Take on Amdahl's Law:
Gustafson's Observation

9

Gustafson observed that as you increase the number of processors, you have a tendency to attack larger and larger versions of the problem. He also observed that when you use the same parallel program on larger datasets, the parallel fraction, F_p, increases.

Let P be the amount of time spent on the parallel portion of an original task and S spent on the serial portion. Then

$$F_p = \frac{P}{P+S} \qquad \text{or} \qquad S = \frac{P - PF_p}{F_p}$$

Parallel
Time

Serial
Time

Without loss of generality, we can set P=1 so that, really, S is now a fraction of P. We now have:

$$S = \frac{1 - F_p}{F_p}$$

Oregon State
University
Computer Graphics

mjb – March 15, 2020

A More Optimistic Take on Amdahl's Law:
Gustafson's Observation

10

We know that if we multiply the amount of data to process by N, then the amount of parallel work becomes NP. Surely the serial work must increase too, but we don't know how much. Let's say it doesn't increase at all, so that we know we are getting an upper bound answer.

In that case, the new parallel fraction is: $F_p^{'} = \dfrac{P'}{P'+S} = \dfrac{NP}{NP+S}$

And substituting for P (=1) and for S, we have:

$$F_p^{'} = \frac{N}{N+S} = \frac{N}{N + \dfrac{1 - F_p}{F_p}}$$

Oregon State
University
Computer Graphics

mjb – March 15, 2020

A More Optimistic Take on Amdahl's Law: Gustafson's Observation

If we tabulate this, we get a table of F_p' values:

		How Many Times More Data to Process									
		1	2	3	4	5	6	7	8	9	10
	0.1	0.10	0.18	0.25	0.31	0.36	0.40	0.44	0.47	0.50	0.53
	0.2	0.20	0.33	0.43	0.50	0.56	0.60	0.64	0.67	0.69	0.71
	0.3	0.30	0.46	0.56	0.63	0.68	0.72	0.75	0.77	0.79	0.81
Original Fp	0.4	0.40	0.57	0.67	0.73	0.77	0.80	0.82	0.84	0.86	0.87
	0.5	0.50	0.67	0.75	0.80	0.83	0.86	0.88	0.89	0.90	0.91
	0.6	0.60	0.75	0.82	0.86	0.88	0.90	0.91	0.92	0.93	0.94
	0.7	0.70	0.82	0.88	0.90	0.92	0.93	0.94	0.95	0.95	0.96
	0.8	0.80	0.89	0.92	0.94	0.95	0.96	0.97	0.97	0.97	0.98
	0.9	0.90	0.95	0.96	0.97	0.98	0.98	0.98	0.99	0.99	0.99
	1.0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

mjb – March 15, 2020

A More Optimistic Take on Amdahl's Law: Gustafson's Observation

Or, graphing it:

mjb – March 15, 2020
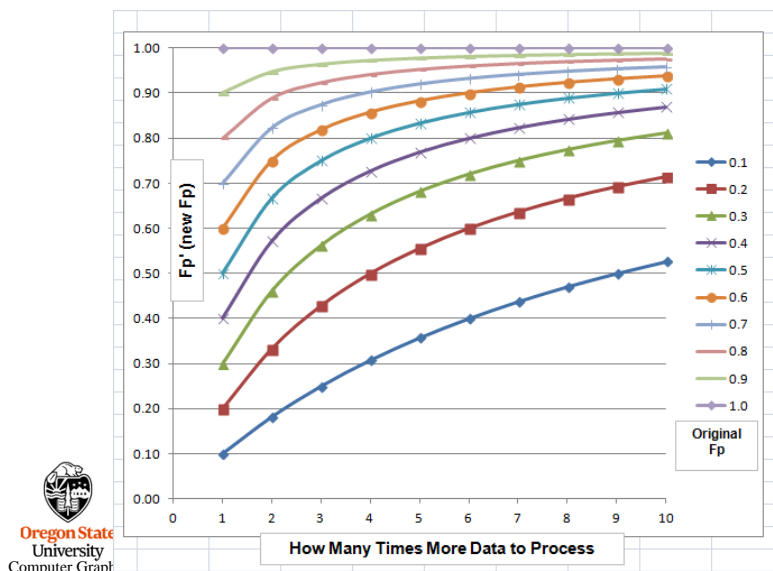
A More Optimistic Take on Amdahl's Law:
Gustafson's Observation

13

We can also turn F_p' into a Maximum Speedup: