Parallel Programming: Speedups and Amdahl’s law

Definition of Speedup

If you are using \( n \) cores, your Speedup, \( Speedup_n \), is:

\[
Speedup_n = \frac{T_1}{T_n} = \frac{P_n}{P_1}
\]

Where:
- \( T_1 \) is the execution time on one core and \( T_n \) is the execution time on \( n \) cores.
- \( P_1 \) is the performance on one core and \( P_n \) is the performance on \( n \) cores.

Note that Speedup, \( n \), should be > 1.

And your Speedup Efficiency, \( Efficiency_n \), is:

\[
Efficiency_n = \frac{Speedup_n}{n}
\]

which could be as high as 1., but probably never will be.
However, Multicore is not a Free Lunch: Amdahl’s Law

If you buy a system with $n$ cores, you should get $n$ times Speedup (and 100% Speedup Efficiency), right? Wrong!

There is 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:

$$Speedup_n = \frac{T_1}{T_n} = \frac{1}{F_{\text{parallel}}/n + F_{\text{sequential}}/n + (1 - F_{\text{parallel}})}$$

This fraction can be reduced by deploying multiple cores. This fraction can’t.

A Visual Explanation of Amdahl’s Law

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

SpeedUp as a Function of $n$ (Number of Cores) and $F_{\text{parallel}}$

$F_{\text{parallel}}$:
- 90%
- 80%
- 60%
- 40%
- 20%

$n=1$  
$n=2$  
$n=3$  
$n=10$

SpeedUp as a Function of $F_{\text{parallel}}$ and $n$ (Number of Cores)
SpeedUp Efficiency \( \frac{S}{n} \) as a Function of Number of Cores and \( F_{\text{parallel}} \)

![Graph showing SpeedUp Efficiency vs. # of Processors](image)

**Legend:**
- 90%
- 80%
- 60%
- 40%
- 20%

**Axes:**
- SpeedUp Efficiency
- n (# of Processors)

---

SpeedUp Efficiency \( \frac{S}{n} \) as a Function of \( F_{\text{parallel}} \) and Number of Cores

![Graph showing SpeedUp Efficiency vs. \( F_{\text{parallel}} \)](image)

**Legend:**
- n=1
- n=2
- n=3
- n=10

**Axes:**
- SpeedUp Efficiency
- \( F_{\text{parallel}} \)
You can also solve for $F_{parallel}$ using Amdahl's Law if you know your speedup and the number of cores.

Amdahl's law says:

$$S = \frac{T \parallel}{T_sequential} = \frac{1}{\frac{F}{n} + (1 - F)}$$

$$\Rightarrow \frac{1}{S} = \frac{F}{n} + (1 - F) = 1 + \frac{F - nF}{n}$$

$$\Rightarrow \frac{1}{S} - 1 = \frac{F (1 - n)}{n}$$

Solving for $F$, the Parallel Fraction:

$$F = \frac{\frac{1}{S} - 1}{\frac{1}{n} - \frac{1}{n}} = \frac{n}{n - 1} \cdot \frac{Speedup - 1}{Speedup}$$

Thus, if you know your Speedup and how many cores you used to get that Speedup, you can compute the Parallel Fraction.

Amdahl's Law can also give us the Maximum Possible SpeedUp.

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

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

<table>
<thead>
<tr>
<th>$F_{parallel}$</th>
<th>maxSpeedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>0.10</td>
<td>1.11</td>
</tr>
<tr>
<td>0.20</td>
<td>1.25</td>
</tr>
<tr>
<td>0.30</td>
<td>1.43</td>
</tr>
<tr>
<td>0.40</td>
<td>1.67</td>
</tr>
<tr>
<td>0.50</td>
<td>2.00</td>
</tr>
<tr>
<td>0.60</td>
<td>2.50</td>
</tr>
<tr>
<td>0.70</td>
<td>3.33</td>
</tr>
<tr>
<td>0.80</td>
<td>5.00</td>
</tr>
<tr>
<td>0.90</td>
<td>10.00</td>
</tr>
<tr>
<td>0.95</td>
<td>20.00</td>
</tr>
<tr>
<td>0.99</td>
<td>100.00</td>
</tr>
</tbody>
</table>
Gustafson observed that as you increase the number of cores, 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

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

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

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

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

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: $F'_p = \frac{P'}{P' + S} = \frac{NP}{NP + S}$

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

$$F'_p = \frac{N}{N + S} = \frac{N}{N + \frac{1 - F_p}{F_p}}$$
A More Optimistic Take on Amdahl's Law: The Gustafson-Baris Observation

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

<table>
<thead>
<tr>
<th>Original $F_p$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.10</td>
<td>0.18</td>
<td>0.25</td>
<td>0.31</td>
<td>0.36</td>
<td>0.40</td>
<td>0.44</td>
<td>0.47</td>
<td>0.50</td>
<td>0.53</td>
</tr>
<tr>
<td>0.2</td>
<td>0.20</td>
<td>0.33</td>
<td>0.43</td>
<td>0.50</td>
<td>0.56</td>
<td>0.60</td>
<td>0.64</td>
<td>0.67</td>
<td>0.69</td>
<td>0.71</td>
</tr>
<tr>
<td>0.3</td>
<td>0.30</td>
<td>0.46</td>
<td>0.56</td>
<td>0.63</td>
<td>0.68</td>
<td>0.72</td>
<td>0.75</td>
<td>0.77</td>
<td>0.79</td>
<td>0.81</td>
</tr>
<tr>
<td>0.4</td>
<td>0.40</td>
<td>0.57</td>
<td>0.67</td>
<td>0.73</td>
<td>0.77</td>
<td>0.80</td>
<td>0.82</td>
<td>0.84</td>
<td>0.86</td>
<td>0.87</td>
</tr>
<tr>
<td>0.5</td>
<td>0.50</td>
<td>0.67</td>
<td>0.75</td>
<td>0.80</td>
<td>0.83</td>
<td>0.86</td>
<td>0.88</td>
<td>0.90</td>
<td>0.91</td>
<td>0.91</td>
</tr>
<tr>
<td>0.6</td>
<td>0.60</td>
<td>0.75</td>
<td>0.82</td>
<td>0.86</td>
<td>0.88</td>
<td>0.90</td>
<td>0.91</td>
<td>0.92</td>
<td>0.93</td>
<td>0.94</td>
</tr>
<tr>
<td>0.7</td>
<td>0.70</td>
<td>0.82</td>
<td>0.88</td>
<td>0.90</td>
<td>0.92</td>
<td>0.93</td>
<td>0.94</td>
<td>0.95</td>
<td>0.95</td>
<td>0.96</td>
</tr>
<tr>
<td>0.8</td>
<td>0.80</td>
<td>0.89</td>
<td>0.92</td>
<td>0.94</td>
<td>0.95</td>
<td>0.96</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>0.9</td>
<td>0.90</td>
<td>0.95</td>
<td>0.96</td>
<td>0.97</td>
<td>0.98</td>
<td>0.98</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>1.0</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Or, graphing it:
A More Optimistic Take on Amdahl’s Law: The Gustafson-Baris Observation

We can also turn $F_p'$ into a Maximum Speedup:

![Graph showing Maximum Speedup vs. How Many Times More Data to Process]