ECE5610/CSC6220
Introduction to Parallel and Distribution Computing

Lecture 6: Analytical Modeling of Parallel Systems (Part2)
Scalability of Parallel Systems

How do we extrapolate performance from small problems and small systems to larger problems on larger configurations?

- Consider three parallel algorithms for computing an $n$-point Fast Fourier Transform (FFT) on 64 processing elements.
- Clearly, it is difficult to infer scaling characteristics from observations on small datasets on small machines.

![Graph showing speedups](image)

**Figure 5.7** A comparison of the speedups obtained by the binary-exchange, 2-D transpose and 3-D transpose algorithms on 64 processing elements with $t_c = 2$, $t_w = 4$, $t_s = 25$, and $t_h = 2$ (see Chapter 13 for details).
Scaling Characteristics of Parallel Programs

• The efficiency of a parallel program can be written as:

\[ E = \frac{S}{p} = \frac{T_S}{pT_P} \]

or

\[ E = \frac{1}{1 + \frac{T_o}{T_S}}. \]

• For a given problem size (i.e., the value of \( T_S \) remains constant).

• As we increase the number of processing elements, \( T_o \) increases and the overall efficiency of the parallel program goes down. This is the case for all parallel programs. \((T_o > (p-1)t_{serial})\)
Scaling Characteristics of Parallel Programs: Example

Consider the problem of adding numbers on processing elements using constants instead of asymptotics.

We have seen that:

\[ T_P = \frac{n}{p} + 2 \log p \]

\[ S = \frac{n}{\frac{n}{p} + 2 \log p} \]

\[ E = \frac{1}{1 + \frac{2p \log p}{n}} \]

**Figure 5.8** Speedup versus the number of processing elements for adding a list of numbers.

**Table 5.1** Efficiency as a function of \( n \) and \( p \) for adding \( n \) numbers on \( p \) processing elements.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( p = 1 )</th>
<th>( p = 4 )</th>
<th>( p = 8 )</th>
<th>( p = 16 )</th>
<th>( p = 32 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>1.0</td>
<td>0.80</td>
<td>0.57</td>
<td>0.33</td>
<td>0.17</td>
</tr>
<tr>
<td>192</td>
<td>1.0</td>
<td>0.92</td>
<td>0.80</td>
<td>0.60</td>
<td>0.38</td>
</tr>
<tr>
<td>320</td>
<td>1.0</td>
<td>0.95</td>
<td>0.87</td>
<td>0.71</td>
<td>0.50</td>
</tr>
<tr>
<td>512</td>
<td>1.0</td>
<td>0.97</td>
<td>0.91</td>
<td>0.80</td>
<td>0.62</td>
</tr>
</tbody>
</table>

Speedup tends to saturate and efficiency drops as a consequence of Amdahl's law.
Scaling Characteristics of Parallel Programs

• Total overhead function $T_o$ is a function of both problem size $T_s$ and the number of processing elements $p$.

• In many cases, $T_o$ grows sublinearly with respect to $T_s$. So the efficiency increases if the problem size is increased.

• For such systems, we can simultaneously increase the problem size and number of processors to keep efficiency constant.

• We call such systems *scalable* parallel systems.

• The scalability of a parallel system is a measure of its capability to increase speedup in proportion to the number of processors.
Scaling Characteristics of Parallel Programs

• Recall that cost-optimal parallel systems have an efficiency of $\Theta(1)$.

• Scalability and cost-optimality are therefore related.

• A scalable parallel system can always be made cost-optimal if the number of processing elements and the size of the computation are chosen appropriately (recall the example using $p$ processors for adding $n$ numbers).
Isoefficiency Metric of Scalability

• For a given problem size, as we increase the number of processing elements, the overall efficiency of the parallel system goes down for all systems.

• For some systems, the efficiency of a parallel system increases if the problem size is increased while keeping the number of processing elements constant.

Figure 5.9  Variation of efficiency: (a) as the number of processing elements is increased for a given problem size; and (b) as the problem size is increased for a given number of processing elements. The phenomenon illustrated in graph (b) is not common to all parallel systems.
Isoefficiency Metric of Scalability

• What is the rate at which the problem size must increase with respect to the number of processing elements to keep the efficiency fixed?

• This rate determines the scalability of the system. The slower this rate, the better.

• Before we formalize this rate, we define the problem size $W$ as the asymptotic number of operations associated with the best serial algorithm to solve the problem.
Isoefficiency Metric of Scalability

We can write parallel runtime as:

\[ T_P = \frac{W + T_o(W, p)}{p} \]

The resulting expression for speedup is

\[ S = \frac{W}{T_P} = \frac{Wp}{W + T_o(W, p)}. \]

Finally, we write the expression for efficiency as

\[ E = \frac{S}{p} = \frac{W}{W + T_o(W, p)} = \frac{1}{1 + T_o(W, p)/W}. \]
Isoefficiency Metric of Scalability

For scalable parallel systems, efficiency can be maintained at a fixed value (between 0 and 1) if the ratio $T_o / W$ is maintained at a constant value.

For a desired value $E$ of efficiency,

$$E = \frac{1}{1 + \frac{T_o(W, p)}{W}}$$

$$\frac{T_o(W, p)}{W} = \frac{1 - E}{E},$$

$$W = \frac{E}{1 - E} T_o(W, p).$$

If $K = \frac{E}{1 - E}$ is a constant depending on the efficiency to be maintained, since $T_o$ is a function of $W$ and $p$, we have

$$W = KT_o(W, p).$$
**Isoefficiency Metric of Scalability**

- The problem size $W$ can usually be obtained as a function of $p$ by algebraic manipulations to keep efficiency constant.
- This function is called the *isoefficiency function*.
- This function determines the ease with which a parallel system can maintain a constant efficiency and hence achieve speedups increasing in proportion to the number of processing elements.
- The function does not exist for unscalable parallel systems because in such systems the efficiency cannot be kept at any constant value as $p$ increases.
Isoefficiency Metric: Example

The overhead function for the problem of adding $n$ numbers on $p$ processing elements is approximately $2p \log p$.

Substituting $T_o$ by $2p \log p$, we get

$$W = K 2p \log p.$$  

Thus, the asymptotic isoefficiency function for this parallel system is

$$\Theta(p \log p).$$

If the number of processing elements is increased from $p$ to $p'$, the problem size (in this case, $n$) must be increased by a factor of $(p' \log p') / (p \log p)$ to get the same efficiency as on $p$ processing elements.
Isoefficiency Metric: Example

Consider a more complex example where \( T_o = p^{3/2} + p^{3/4}W^{3/4} \)

Using only the first term of \( T_o \), we get

\[
W = Kp^{3/2}.
\]

Using only the second term, yield the following relation between \( W \) and \( p \):

\[
W = Kp^{3/4}W^{3/4}
\]

\[
W^{1/4} = Kp^{3/4}
\]

\[
W = K^4p^3
\]

The larger of these two asymptotic rates determines the isoefficiency. This is given by \( \Theta(p^3) \)
Asymptotic Analysis of Parallel Programs

Consider the problem of sorting a list of $n$ numbers. The fastest serial programs for this problem run in time $\Theta(n \log n)$. Consider four parallel algorithms, A1, A2, A3, and A4 as follows:

**Table 5.2** Comparison of four different algorithms for sorting a given list of numbers. The table shows number of processing elements, parallel runtime, speedup, efficiency and the $pT_P$ product.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>$n^2$</td>
<td>$\log n$</td>
<td>$n$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>$T_P$</td>
<td>1</td>
<td>$n$</td>
<td>$\sqrt{n}$</td>
<td>$\sqrt{n} \log n$</td>
</tr>
<tr>
<td>$S$</td>
<td>$n \log n$</td>
<td>$\log n$</td>
<td>$\sqrt{n} \log n$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>$E$</td>
<td>$\frac{\log n}{n}$</td>
<td>1</td>
<td>$\frac{\log n}{\sqrt{n}}$</td>
<td>1</td>
</tr>
<tr>
<td>$pT_P$</td>
<td>$n^2$</td>
<td>$n \log n$</td>
<td>$n^{1.5}$</td>
<td>$n \log n$</td>
</tr>
</tbody>
</table>

- If the metric is speed, algorithm A1 is the best, followed by A3, A4, and A2 (in order of increasing $T_P$).
- In terms of efficiency, A2 and A4 are the best, followed by A3 and A1.
- In terms of cost, algorithms A2 and A4 are cost optimal, A1 and A3 are not.
- It is important to identify the objectives of analysis and to use appropriate metrics!