Analytical Modeling of Parallel Programs

S. Oliveira

Scalability of Parallel Systems

- Efficiency of a parallel program

  $E = S/P = T_s/PT_p$

- Using the parallel overhead expression

  $E = 1/(1 + T_o/T_s)$

- The total overhead function T_o is an increasing function of P.
- For a given problem size, T_s remains constant, as we increase the number of P, T_o increases.
- Efficiency decreases with increasing number of processing elements ~ characteristic common to all parallel programs.

Scalability of Parallel Systems Cont'd

- Speedup and efficiency as functions to P:
  - T_p = n /p +2 log p
  - S = n/ [(n/p) + 2 log p]
  - E = 1/ 1+(2p log p /n)

Efficiency as a function of n and p for adding n numbers on p processors.

n	$p = 1$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
64	1.0	*.80*	.57	.33	.17
192	1.0	.92	*.80*	.60	.38
320	1.0	.95	.87	.71	.50
512	1.0	.97	.91	*.80*	.62

  - Both Efficiency and Speedup increase with n, for the same number of P.
  - Both Efficiency and Speed up drop with increasing P. i

Scalability of Parallel Systems Cont'd

- It is possible to keep the efficiency in many parallel systems fixed by increasing both the size of the problem and the number of processing elements simultaneously scalable parallel system.
- Scalability: measure of a parallel system's capacity to increase speedup in proportion to the number of processing elements.
- Recall that a cost optimal parallel system has efficiency of $\Theta(1)$. So, scalability and cost-optimality of parallel systems are related. We can have a cost optimal parallel system if the # of P and size of computation are chosen appropriately.

Isoefficiency Metric of Scalability

- We said that for a given problem size, as we increase # of P, the overall efficiency of parallel system decreases.
- **Scalable Parallel System** is one in which efficiency can be kept constant as the number of processing elements is increased, provided problem size is also increased.
- The rate of change of both P and problem size determine the degree of scalability of a parallel system. A lower rate is more desirable than a higher growth rate in problem size.
- **Problem Size** is defined as the number of basic computation steps in the best sequential algorithm to solve the problem on a single P.
- We assume that it takes unit time to perform one basic computation of the algorithm. Thus Problem size = W = T_s (time of fastest known sequential algorithm).

Isoefficiency Function

- Parallel Execution Time can be expressed as a function of:
  - Problem Size
  - Overhead function
  - Number of Processing elements
- We can rewrite parallel runtime (T_p) as:

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

$$S = \frac{W}{T_P} = \frac{Wp}{W + T_o(W, p)}$$

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

Isoefficiency Function cont'd…

- Conclusions:
  - If the problem size (W) is kept constant and p is increased, efficiency (E) decreases because the total overhead (T_o) increases with p.
  - For different parallel systems, W must be increased at different rates with respect to p in order to maintain a fixed efficiency.
  - For scalable parallel systems E can be maintained at a fixed value (0<E<1) if the ratio T_o/W is maintained constant.

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

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

K = E /(1-E), we get:

$$W = K T_o(W, p)$$

Isoefficiency Function cont'd…

- **Isoefficiency Function:** $W = KT_o(W,P)$. This function determines the ease with which a parallel system can maintain a constant efficiency and achieve speedups increasing in proportion to number of processing elements.
- A small isoefficiency function means that small increments in problem size are sufficient for efficient utilization of increasing # of P, indicating that the system is highly scalable.
- However a large isoefficiency function indicates a poorly scalable parallel system.
- The isoefficiency function does not exist for unscalable parallel systems because the E can't be kept at any constant value as p increases, no matter how fast the problem size is increased.
- Finally, the isoefficiency function captures the characteristics of a parallel algorithm as well as the parallel architecture on which it is implemented.

Cost Optimality & Isoefficiency Function

- A parallel system is cost optimal if the product of the number of processing elements and the parallel execution time is proportional to the execution time of the fastest known sequential algorithm on a single processing element.
- A parallel system is cost optimal IFF: $pT_P = \Theta(W)$

$$
\begin{aligned}
W + T_o(W, p) &= \Theta(W) \\
T_o(W, p) &= O(W) \\
W &= \Omega(T_o(W, p))
\end{aligned}
$$

- The above 2 equations suggest that a parallel system is cost optimal IFF its overhead function does not asymptotically exceed the problem size.
- $W = \Omega(f(p))$ must be satisfied to ensure the cost-optimality of a parallel system as it is scaled up.

Lower Bound on Isoefficiency Function

- An ideally-scalable parallel system must have the lowest possible isoefficiency function.
- For a problem consisting of W units of work, no more than W processing elements can be used cost-optimally , as additional p will be idle.
- The problem size must increase at least as fast as Θ (p) to maintain fixed efficiency; hence Ω(p) is the asymptotic lower bound on the isoefficiency function.

Degree of Concurrency & Isoefficiency Function

- The maximum number of tasks that can be executed simultaneously at any time in a parallel algorithm is called its *degree of concurrency*.
- Degree of Concurrency: is a measure of the number of operations that an algorithm can perform in parallel for a problem of size W.
- If C(W) is the degree of concurrency of a parallel algorithm, then for a problem of size W, no more than C (W) processing elements can be employed effectively.
- The isoefficiency function due to concurrency is optimal, i.e. Θ (p), only if the degree of concurrency of the parallel algorithm is Θ (W).

Minimum Execution Time & Minimum Cost-Optimal Execution Time

- If we increase the number of processing elements for a given problem size, either the parallel runtime continues to decrease and asymptotically approaches a minimum value or it starts rising after attaining a minimum value.
- A problem of size W, a cost-optimal solution requires that $p = O(f^{-1}(W))$. Since the parallel runtime is $\Theta(W/P)$ for a cost optimal parallel system.

$$T_P^{cost\_opt} = \Omega\left(\frac{W}{f^{-1}(W)}\right)$$

Asymptotic Analysis of Parallel Programs.

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.

Algorithm	A1	A2	A3	A4
p	n^2	$\log n$	n	$\sqrt{n}$
T_P	1	n	$\sqrt{n}$	$\sqrt{n}\log n$
S	$n\log n$	$\log n$	$\sqrt{n}\log n$	$\sqrt{n}$
E	$\frac{\log n}{n}$	1	$\frac{\log n}{\sqrt{n}}$	1
pT_P	n^2	$n\log n$	$n^{1.5}$	$n\log n$