**PERFORMANCE & SCALABILITY**

- parallel system \( \equiv \) algorithm + architecture
- Speedup \( S = \frac{T_s}{T_p} \) \( \approx ? \)
- Superlinear speedup ?
- Efficiency \( E = S / p \)
- Cost \( \propto p \cdot T_p \)
- Cost-optimal if \( \text{Cost} \propto T_s \), \( E = \Theta(1) \)

(a) Initial data distribution and the first communication step

(b) Second communication step

(c) Third communication step

(d) Fourth communication step

(e) Accumulation of the sum at processor 0 after the final communication

**Figure 4.1** Computing the sum of 16 numbers on a 16-processor hypercube. \( \Sigma_i^j \) denotes the sum of numbers with consecutive labels from \( i \) to \( j \).

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

- \( S = \Theta \left( \frac{n}{\log n} \right) \)
- \( E = \Theta \left( \frac{1}{\log n} \right) \)
- \( \text{Cost} = \Theta \left( n \log n \right) \)
- not cost optimal

Examine effect of granularity and data mapping:
"if system with \( n \) procs is cost-optimal, using \( p \) procs (\( p < n \)) to simulate \( n \) procs preserves cost-optimality"
\[ \Theta\left(\frac{n}{p}\log p\right) + \Theta\left(\frac{n}{p}\right) = \Theta\left(\frac{n}{p}\log p\right) \text{ not cost-optimal} \]

- Overhead \( T_0 = \Theta(n\log p) \)
- Problem size \( w \) to add \( n \) numbers is \( \Theta(n) \)

(a) Four processors simulating the first communication step of 16 processors

\[ 12 \quad 13 \quad 14 \quad 15 \]
\[ 8 \quad 9 \quad 10 \quad 11 \]
\[ 4 \quad 5 \quad 6 \quad 7 \]
\[ 0 \quad 1 \quad 2 \quad 3 \]

Substep 1

\[ 12 \quad 13 \quad 14 \quad 15 \]
\[ 8 \quad 9 \quad 10 \quad 11 \]
\[ \Sigma^1_0 \quad \Sigma^3_2 \]
\[ 0 \quad 1 \quad 2 \quad 3 \]

Substep 2

\[ 12 \quad 13 \quad 14 \quad 15 \]
\[ 8 \quad 9 \quad 10 \quad 11 \]
\[ \Sigma^5_4 \quad \Sigma^7_6 \]
\[ 0 \quad 1 \quad 2 \quad 3 \]

Substep 3

\[ 12 \quad 13 \quad 14 \quad 15 \]
\[ 8 \quad 9 \quad 10 \quad 11 \]
\[ \Sigma^9_8 \quad \Sigma^{11}_{10} \]
\[ 0 \quad 1 \quad 2 \quad 3 \]

Substep 4

(b) Four processors simulating the second communication step of 16 processors

\[ \Sigma^{13}_{12} \quad \Sigma^{15}_{14} \]
\[ \Sigma^9_8 \quad \Sigma^{11}_{10} \]
\[ \Sigma^5_4 \quad \Sigma^7_6 \]
\[ \Sigma^1_0 \quad \Sigma^3_2 \]

Substep 1

\[ \Sigma^{13}_{12} \quad \Sigma^{15}_{14} \]
\[ \Sigma^9_8 \quad \Sigma^{11}_{10} \]
\[ \Sigma^5_4 \quad \Sigma^7_6 \]
\[ \Sigma^1_0 \quad \Sigma^3_2 \]

Substep 3

\[ \Sigma^{13}_{12} \quad \Sigma^{15}_{14} \]
\[ \Sigma^9_8 \quad \Sigma^{11}_{10} \]
\[ \Sigma^5_4 \quad \Sigma^7_6 \]
\[ \Sigma^1_0 \quad \Sigma^3_2 \]

Substep 4

---

Figure 4.2 Four processors simulating 16 processors to compute the sum of 16 numbers (first two steps). \( \Sigma^i_j \) denotes the sum of numbers with consecutive labels from \( i \) to \( j \).

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

- Isoefficiency function does not exist because \( [w = KT_0(w, p)] \) cannot be satisfied for any \( K \). System not scalable.
(c) Simulation of the third step in two substeps

(d) Simulation of the fourth step

(e) Final result

Figure 4.3 (cont.) Four processors simulating 16 processors to compute the sum of 16 numbers (last three steps).
Copyright (r) 1994 Benjamin/Cummings Publishing Co.
\[ \Theta\left(\frac{n}{p}\right) + \Theta\left(\log p\right) = T_p \]

- Cost = \( \Theta\left(n + p \log p\right) = \text{cost-optimal as long as } n = \Omega\left(p \log p\right) \)

- \( T_p = \frac{n}{P} - 1 + 2 \log p \)
- \( S = \frac{n}{n/p + 2 \log p} = \frac{np}{n + 2p \log p} \)
- \( E = \frac{n}{n + 2p \log p} \)

- \( T_0(W, p) = p \left(\frac{n}{p} + 2 \log p\right) - n = 2p \log p \)

- (Isoefficiency function): \( W = 2K p \log p \) i.e., \( \Theta(p \log p) \)

- \( W = n & T_0 = \Theta(p \log p) \) & condn. for cost-optimality: \( W = \Omega(p \log p) \)

---

**Figure 4.4** A cost-optimal way of computing the sum of 16 numbers on a four-processor hypercube.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
1. Speedup does not increase linearly as number of processors increases. 
2. Larger instance of same problem yields speedup and efficiency (for fixed p).

> Very common trends

- Scalable parallel system = (increase in parallelism to keep efficiency fixed)
- Scalability = measure of capacity to increase in proportion to p
- \( E = \Theta(1) \) for cost-optimal system

- Scalability & cost-optimality are related: scalable system can always be made cost-optimal if p and (size of computation) chosen correctly.

Example below: system stays cost-optimal at \( E = 0.8 \) if \( n \) is increased as \( p \log p \)

![Graph showing speedup versus number of processors](image)

**Figure 4.5** Speedup versus the number of processors for adding a list of numbers on a hypercube.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

<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</td>
<td>.8</td>
<td>.57</td>
<td>.33</td>
<td>.17</td>
</tr>
<tr>
<td>192</td>
<td>1</td>
<td>.92</td>
<td>.8</td>
<td>.6</td>
<td>.38</td>
</tr>
<tr>
<td>320</td>
<td>1</td>
<td>.95</td>
<td>.87</td>
<td>.71</td>
<td>.5</td>
</tr>
<tr>
<td>512</td>
<td>1</td>
<td>.97</td>
<td>.91</td>
<td>.8</td>
<td>.62</td>
</tr>
</tbody>
</table>

Efficiency as a function of \( n \) & \( p \) for Fig 4.4:

- Cost-optimal when \( n = \lfloor 2(p \log p) \rfloor \). For \( n = 64 \), \( n = 8 + \log p \)
- 64 = \( 8 \times \lceil 4 \log 4 \rceil \)
- 512 = \( 8 \times \lceil 16 \log 16 \rceil \)
- 192 = \( 8 \times \lceil 8 \log 8 \rceil \)
- Performance of scaled down algo may be different for different assignments of (virtual $\rightarrow$ real) processors

$$\begin{bmatrix} n \times n \end{bmatrix} \times \begin{bmatrix} n \times 1 \end{bmatrix} \text{ on } p \text{ proc HC } \mapsto p \text{ square blocks}$$

- ISOEFFICIENCY METRIC OF SCALABILITY

- Useful to determine rate at which problem size must $\uparrow$ w.r.t. $p$ to keep the efficiency fixed

- Problem size $W = \#$ computation steps in best seq. algo. on 1 proc.

- Overhead $T_o(W, p) = pT_p - W$

$$T_p = \frac{T_o + W}{p}$$

$$S = \frac{W}{T_p} = \frac{Wp}{T_o + W}$$

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

Examine effect on $E$!

Typically, $T_o$ grows slower than $\Theta(W)$ for a fixed $p$

$$W = \left(\frac{E}{1-E}\right)T_o(W, p) = K T_o(W, p) \quad \text{isoefficiency func.}$$

- Func determines ease with which parallel system can maintain const. efficiency & hence achieve speedups increasing in proportion to $p$

- Small $\Rightarrow$ small $\uparrow$ in $W$ sufficient to efficiently utilize $\#$ proc.

- Large $\Rightarrow$ poorly scalable

- $T_o$ may have terms of different orders of magnitude component that requires $W$ to grow at the highest rate w.r.t. $p$ determines overall asymptotic isoefficiency function

$$T_o = p^{3/2} + p^{3/4}W^{3/4} \quad \Rightarrow \quad W = K p^{3/4} W^{3/4} \Rightarrow W = K p^{3/4}$$

$\Theta(p^{3/2})$ 1st term & $\Theta(p^3)$ 2nd term
• Parallel system is cost-optimal iff $p \cdot T_p = \Theta(w)$

\[
\begin{align*}
\text{ie, } w + T_o(W, p) & = \Theta(w) \\
T_o(w, p) & = O(w) \\
W & = \Omega(T_o(w, p)) \\
\end{align*}
\]

\[W\]

\[\text{ie iff } T_o \text{ does not asymptotically exceed} \]

\[W\]

• If $W = KT_o(w, p)$ gives iso-eficiency function $f(p)$, then

$W = \Omega(f(p))$ must hold to ensure cost-optimality with scaling

• **Lower Bound on Isoefficiency Function**

→ Asymptotically, $W$ must increase at least as fast as $\Theta(p)$

→ to maintain fixed efficiency

→ $\Omega(p)$ is asymptotic lower bound on iso-eficiency $fn$.

→ Ideally, $\Theta(p)$.

• **Degree of Concurrency & Isoefficiency Function**

→ measure of # ops in parallel as a function of $W$

→ eq solve $n$ eqns in $n$ variables using Gaussian elimination

\[\begin{align*}
\text{Computation} & = \Theta(n^3) \text{ totally, but} \\
\text{n vars are sequentially eliminated, each needs } \Theta(n^2) \text{ computations} \\
W & = \Theta(n^3) , \ C(W) = \Theta(W^{2/3}) \\
\end{align*}\]

→ Given $p$ processors, problem size $W \geq \Omega(p^{3/2})$ to use all proc

→ Iso-eficiency func (due to concurrency) is optimal, ie $\Theta(p)$ only if

$C(W) = \Theta(W)$

→ $C(W) < \Theta(W)$, iso-eficiency fn (due to concurrency) $> \Theta(p)$

• **Sources of Parallel Overhead $T_o$**

1. Interprocessor communication
2. Load imbalance → cannot predict / need to sync/seq. components
3. Extra computation → fastest seq. algo may not hard to parallelize

\[\Omega(\text{serial}) \text{ reuse results, (parallel): cannot reuse, as generated by different processors}}\]

\[\Omega(\text{FF})\]