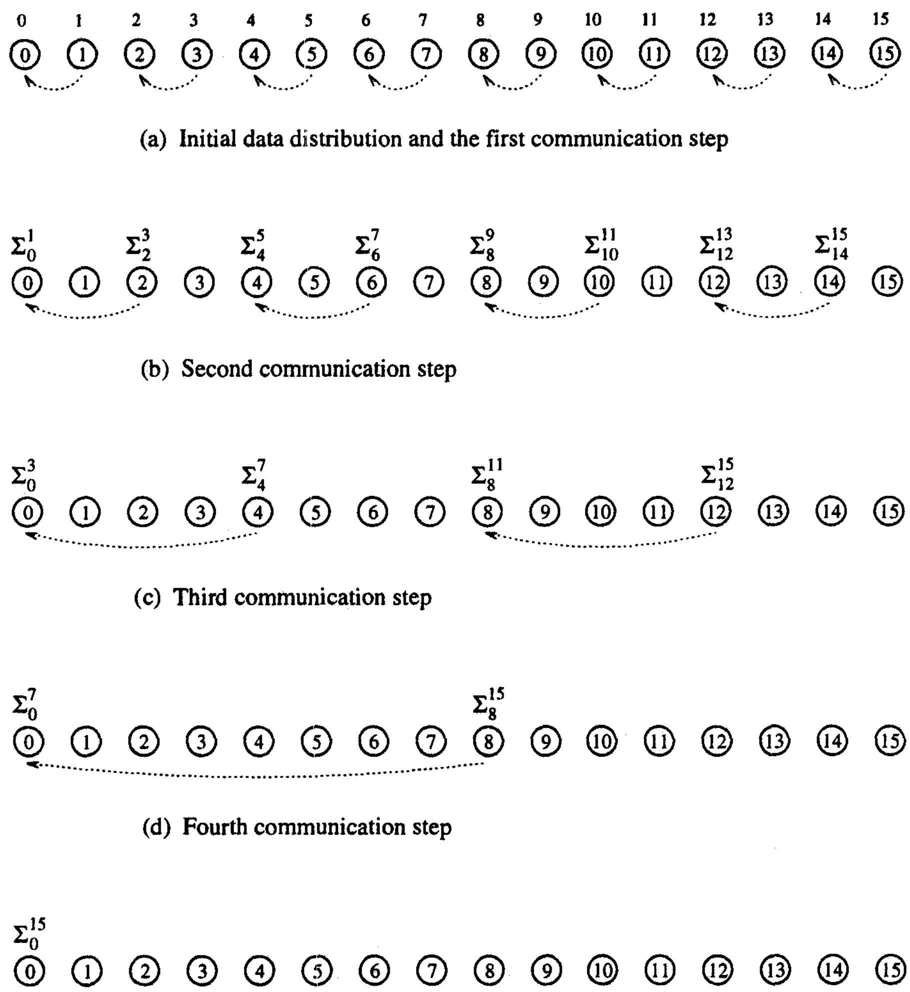
PERFORMANCE & SCALABILITY

- parallel system $\equiv$ algorithm + architecture
- Speedup $S = \dfrac{T_s \rightsquigarrow ?}{T_p \rightsquigarrow ?}$
- Superlinear speedup ?
- Efficiency $E = S/p$
- Cost $= p \cdot T_p$
- Cost-optimal if 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. Σ_i^j denotes the sum of numbers with consecutive labels from i to j.

- $S = \Theta\left(\frac{n}{\log n}\right)$
- $E = \Theta\left(\frac{1}{\log n}\right)$
- Cost $= \Theta(n \log n)$
- 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"

• $\Theta\left(\frac{n}{p} \log p\right) + \Theta\left(\frac{n}{p}\right) = \Theta\left(\frac{n}{p} \log p\right)$ ∴ <u>not</u> cost-optimal

• Overhead $T_0 = \Theta(n \log p)$

• Problem size W to add n numbers is $\Theta(n)$

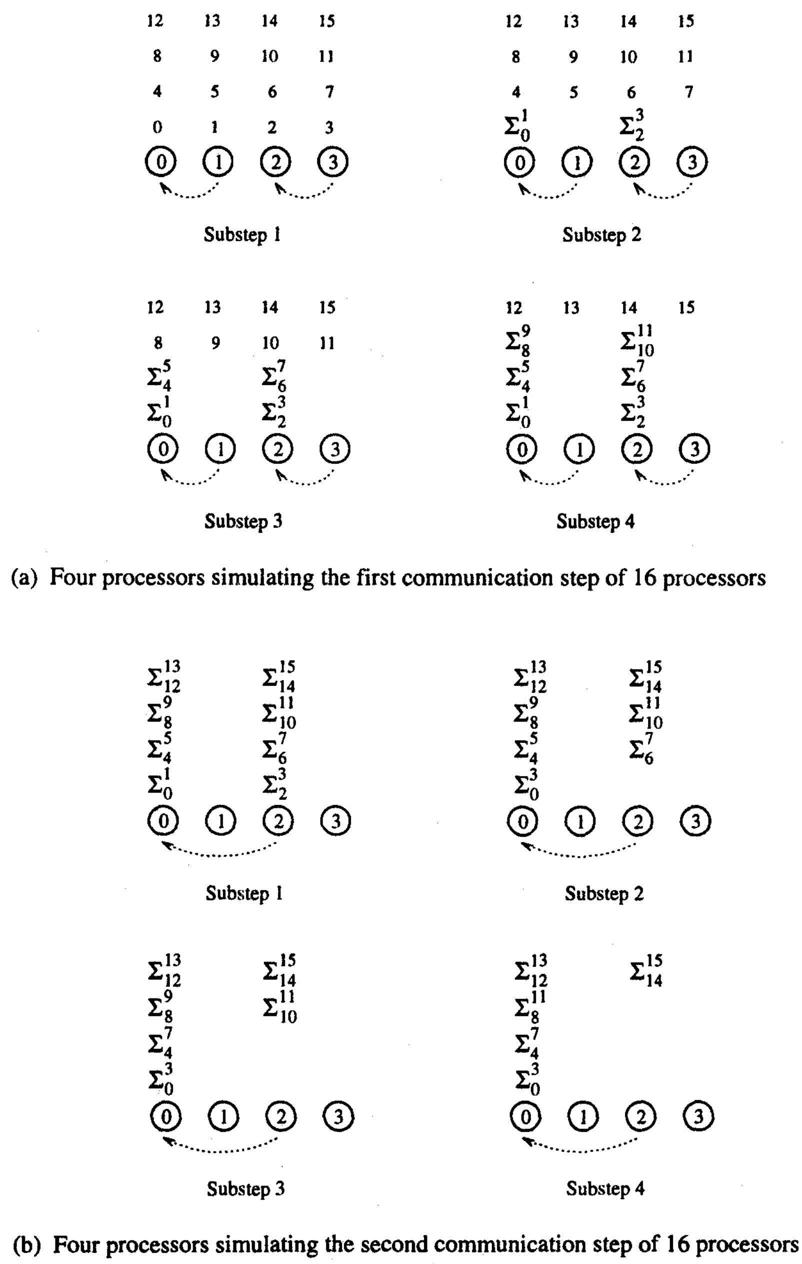


Figure 4.2 Four processors simulating 16 processors to compute the sum of 16 numbers (first two steps). Σ_i^j denotes the sum of numbers with consecutive labels from i to j.

• Isoefficiency function does not exist because $[W = K T_0(W, p)]$ cannot be satisfied for any K ∴ system not scalable

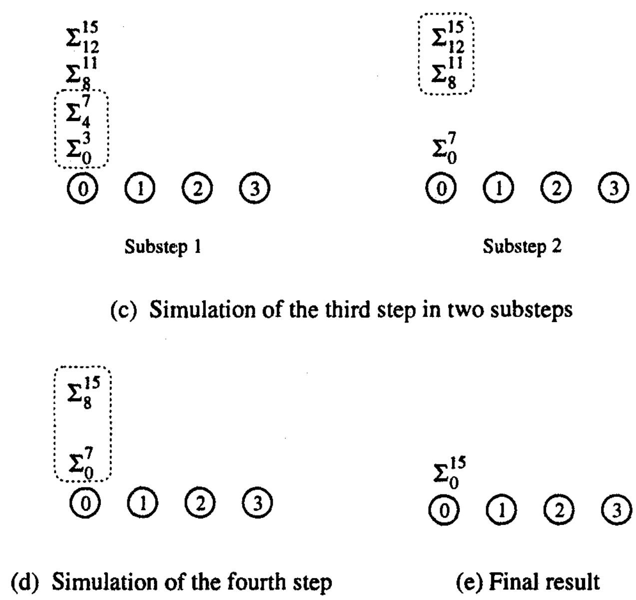


Figure 4.3 (cont.) Four processors simulating 16 processors to compute the sum of 16 numbers (last three steps).

- $\Theta(n/p) + \Theta(\log p) = T_p$
- Cost $= \Theta(n + p\log p)$ = cost-optimal as long as $n = \Omega(p\log p)$
- $T_p = \frac{n}{p} - 1 + 2\log p$
- $S = \frac{n}{n/p + 2\log p} = \frac{np}{n + 2p\log p}$
- $E = \frac{n}{n + 2p\log p}$
- $T_o(W, p) = p\left(\frac{n}{p} + 2\log p\right) - n = 2p\log p$
- (isoefficiency func): $W = 2Kp\log p$ ie, $\Theta(p\log p)$
- $W \approx n$ & $T_o = \Theta(p\log p)$ & condn. for cost-optimality: $W = \Omega(p\log p)$

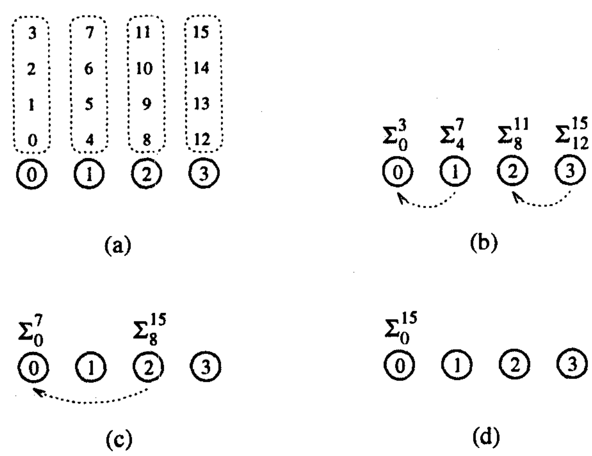


Figure 4.4 A cost-optimal way of computing the sum of 16 numbers on a four-processor hypercube.

① Speedup does not ↑ linearly as # procs increases ∴ E ↓
② larger instance of same problem yields ↑ S & E (for fixed p)
→ Very common trends

- Scalable parallel system ≡ (↑ p & problem size) to keep E fixed
- Scalability ≡ measure of capacity to ↑ S 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.

eg below : system stays cost-optimal at $E = 0.8$ if n is increased as $p \log p$

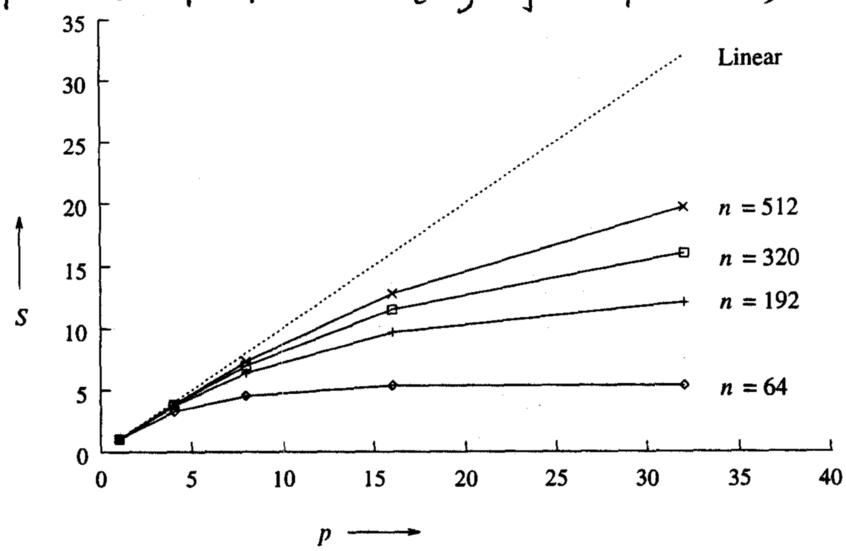


Figure 4.5 Speedup versus the number of processors for adding a list of numbers on a hypercube.

n	p=1	p=4	p=8	p=16	p=32
64	1	.8	.57	.33	.17
192	1	.92	.8	.6	.38
320	1	.95	.87	.71	.5
512	1	.97	.91	.8	.62

[Efficiency as a function of n & p for Fig 4.4]

- Cost-optimal when $n = \Omega(p \log p)$. For $\begin{bmatrix} n=64 \\ p=4 \end{bmatrix}$, $n = 8\, p \log p$

$64 = 8 \times [4 \log 4]$ $\quad 512 = 8\,[16 \log 16]$
$192 = 8 \times [8 \log 8]$

- performance of scaled down algo may be different for different assignments of (virtual $\to$ real) processors

eg $[n \times n] \times [n \times 1]$ on p-proc HC $\to$ p square blocks / $\to$ p stripes of n/p cols/rows

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

$$T_p = (T_o + W)/p$$

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

$$E = \frac{S}{p} = \frac{W}{W + T_o} = \frac{1}{1 + T_o(W,p)/W}$$. Examine effect on E!

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

$$W = \left(\frac{E}{1-E}\right) T_o(W,p) = K T_o(W,p)$$ -------- 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 $\uparrow$# proce
- 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

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

$\therefore \Theta(p^{3/2})$ 1st term & $\Theta(p^3)$ 2nd term

- Parallel system is cost-optimal <u>iff</u> $p \cdot T_p = \Theta(W)$

<u>ie</u>, $W + T_o(W, p) = \Theta(W)$

$T_o(W, p) = O(W)$
$W = \Omega(T_o(W, p))$ } <u>ie</u> iff T_o does not asymptotically exceed W

- If $W = K T_o(W, p)$ gives isoefficiency function $f(p)$, then $W = \Omega(f(p))$ must hold to ensure cost-optimality with scaling

- <u>LOWER BOUND ON ISOEFFICIENCY FUNCTION</u>

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

$\therefore \Omega(p)$ is asymptotic lower bound on isoefficiency fn.

Ideally, $\Theta(p)$.

- <u>DEGREE OF CONCURRENCY & ISOEFFICIENCY FUNCTION</u>

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

→ eg solve n eqns in n variables using Gaussian elimination

Computation $= \Theta(n^3)$ totally, but

n vars are sequentially eliminated, each needs $\Theta(n^2)$ computations

$W = \Theta(n^3)$, $C(W) = \Theta(W^{2/3})$

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

→ Isoefficiency func (due to concurrency) is optimal, <u>ie</u> $\Theta(p)$ only if $C(W) = \Theta(W)$

If $C(W) < \Theta(W)$, isoefficiency fn (due to concurrency) $> \Theta(p)$

- <u>SOURCES OF PARALLEL OVHD T_o:</u> ① Interprocessor communication

② Load imbalance → cannot predict / need to sync/seq. components

③ Extra computation → fastest seq. algo may be hard to parallelize

↳ (serial:) reuse results, (parallel:) cannot reuse, as generated by different processors ∴ extra computation eg FFT