Cost and Optimality

- **Cost** = \( p \cdot T_p \)
  - \( p \): number of processors
  - \( T_p \): Time complexity for parallel execution
  - Also referred to as **processor-time product**
  - Time can take communication into account
    - Problem with mixing processing time and communication time
    - Simple but unrealistic:
      - operation: 1 time unit
      - communicate with direct neighbor: 1 time unit

- **Cost optimal** if Cost = \( O(T_1) \)
E.g. - Add $n$ numbers on hypercube

- $n$ numbers on $n$ processor cube
  - Cost?, cost optimal?
  - assume 1 add = 1 time step
    - 1 comms = 1 time step
  - Assume the numbers are already distributed over the cube

- $n$ numbers on $p$ ($<n$) processor cube
  - Cost?, cost optimal? $S(n)$? $E(n)$?
  - Again, assume the numbers are already distributed over the cube
E.g. - Add \( n \) numbers on hypercube

- \( n \) numbers on \( n \) processor cube
  - Cost = \( O(n \cdot \log(n)) \), not cost optimal

- \( n \) numbers on \( p \) (<\( n \)) processor cube
  - \( T_p = \frac{n}{p} + 2 \cdot \log(p) \)
  - Cost = \( O(n + p \cdot \log(p)) \),
    cost optimal if \( n = O(p \cdot \log(p)) \)
  - \( S = \frac{n \cdot p}{n + 2 \cdot p \cdot \log(p)} \)
  - \( E = \frac{n}{n + 2 \cdot p \cdot \log(p)} \)
E.g. - Add $n$ numbers on hypercube

- $n$ numbers on $p$ ($<n$) processor cube
  - $T_p = \frac{n}{p} + 2 \cdot \log(p)$
  - Cost = $O(n + p \cdot \log(p))$,
    cost optimal if $n = O(p \cdot \log(p))$
  - $S = \frac{n \cdot p}{n + 2 \cdot p \cdot \log(p)}$
  - $E = \frac{n}{n + 2 \cdot p \cdot \log(p)}$
- Build a table: $E$ as function of $n$ and $p$
  - Rows: $n = 64, 192, 512$  Cols: $p = 1, 4, 8, 16$
  - larger $n$ $\rightarrow$ higher $E$, larger $p$ $\rightarrow$ lower $E$
\[ E = \frac{n}{n + 2p \log(p)} \]

<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>1</td>
<td>( \frac{64}{64+16} = \frac{4}{5} )</td>
<td>( \frac{64}{64+48} = \frac{4}{7} )</td>
<td>( \frac{64}{64+128} = \frac{1}{3} )</td>
</tr>
<tr>
<td>192</td>
<td>1</td>
<td>( \frac{192}{192+16} = \frac{12}{13} )</td>
<td>( \frac{192}{192+48} = \frac{4}{5} )</td>
<td>( \frac{192}{192+128} = \frac{3}{5} )</td>
</tr>
<tr>
<td>512</td>
<td>1</td>
<td>( \frac{512}{512+16} = \frac{32}{33} )</td>
<td>( \frac{512}{512+48} = \frac{32}{35} )</td>
<td>( \frac{512}{512+128} = \frac{4}{5} )</td>
</tr>
</tbody>
</table>
Observations

- to keep $E=80\%$ when growing $p$, we need to grow $n$
  - larger $n \rightarrow$ larger $E$
  - larger $p \rightarrow$ smaller $E$
Scalability

- Ability to keep the efficiency fixed, when $p$ is increasing, provided we also increase $n$

- e.g. Add $n$ numbers on $p$ processors (cont.)
  - Look at the $(n,p)$ efficiency table
  - Efficiency is fixed (at 80%) with $p$ increasing
    - only if $n$ is increased
Efficiency is fixed (at 80%) with $p$ increasing only if $n$ is increased
How much?
$E = \frac{n}{n + 2p \log p} = \frac{4}{5}$
$4(n + 2p \log p) = 5n$
n = 8p \log p
(Check with the table)
Iso-efficiency metric

- Iso-efficiency of a scalable system
  - measures degree of scalability of parallel system
  - parallel system: algorithm + topology
    + compute / communication cost model

- Iso-efficiency of a system: the growth rate of problem size $n$, in terms of number of processors $p$, to keep efficiency fixed
  - eg $n = O(p \log p)$ for adding on a hypercube
Sources of Overhead

- Communication
  - PE - PE
  - PE – memory
  - And the busy waiting associated with this

- Load imbalance
  - Synchronization causes **idle processors**
  - Program parallelism does not match machine parallelism all the time
    - Sequential components in computation

- Extra work
  - To achieve independence (avoid communication), parallel algorithms sometimes re-compute values