Scalable Multiprocessors

[§7.1] A scalable system is one in which resources can be added to the system without reaching a hard limit.

Of course, there may still be economic limits.

As the size of the system is increased, is the increase in capacity worth the increase in cost?

Are bus-based designs scalable?

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical length</td>
<td>≈ 1 ft</td>
</tr>
<tr>
<td>Number of connections</td>
<td>fixed</td>
</tr>
<tr>
<td>Maximum bandwidth</td>
<td>fixed</td>
</tr>
<tr>
<td>Interface to comm. medium</td>
<td>memory interf.</td>
</tr>
<tr>
<td>Global order</td>
<td>arbitration</td>
</tr>
<tr>
<td>Protection</td>
<td>Virtual → physical</td>
</tr>
<tr>
<td>Trust</td>
<td>total</td>
</tr>
<tr>
<td>OS</td>
<td>single</td>
</tr>
<tr>
<td>Comm. abstraction</td>
<td>HW</td>
</tr>
</tbody>
</table>

In a bus-based configuration, true, processors can be added.

At the other end of the spectrum, consider a LAN:
<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Bus</th>
<th>LAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical length</td>
<td>≈ 1 ft</td>
<td>KM</td>
</tr>
<tr>
<td>Number of connections</td>
<td>fixed</td>
<td>many</td>
</tr>
<tr>
<td>Maximum bandwidth</td>
<td>fixed</td>
<td>?</td>
</tr>
<tr>
<td>Interface to comm. medium</td>
<td>memory interf.</td>
<td>peripheral</td>
</tr>
<tr>
<td>Global order</td>
<td>arbitration</td>
<td>?</td>
</tr>
<tr>
<td>Protection</td>
<td>Virt -&gt; physical</td>
<td>OS</td>
</tr>
<tr>
<td>Trust</td>
<td>total</td>
<td>none</td>
</tr>
<tr>
<td>OS</td>
<td>single</td>
<td>independent</td>
</tr>
<tr>
<td>Comm. abstraction</td>
<td>HW</td>
<td>SW</td>
</tr>
</tbody>
</table>

No clear limit to physical scaling, little trust, no global order, consensus difficult to achieve.

Independent failure and restart.

What are the design trade-offs for the spectrum of machines between?

- specialized or commodity nodes
- capability of node-to-network interface
- supporting programming models

What does scalability mean?

- Avoids inherent design limits on resources.
- Bandwidth increases with $p$.
- Latency does not.
- Cost increases slowly with $p$.

**Bandwidth scalability**

[§7.1.1] If a large number of processors are to exchange information simultaneously with each other, or with memories, a large number of wires must connect them.

Scalable machines must be organized in the general fashion illustrated below.
- A large number of independent processor modules and memory modules
- are connected together by independent wires (or links) through a large number of switches.

In a bus-based multiprocessor, the address must be broadcast, and all nodes must be consulted to determine which output is to participate in the transaction.

In a network switch, however, the switch controller determines the proper output (from the input).

The most common structure for scalable machines is illustrated below.
Processors and memories are packaged together with a “communication assist” in a replicable unit called a *node*.

Alternatively, the dance-hall organization can be used.

Here, processing nodes are separated from memories by the network. In this case, bandwidth requirement scales with the number of processors. However, access latencies scale with the number of processors.

However, latencies to local memory don’t increase in a node-based design.

To achieve scalable bandwidth, we need a large number of concurrent transactions *over different wires*.
This means the transactions are not globally ordered.

**Latency scaling**

[§7.1.2] How does latency scale as the number of nodes increases?

The time to transfer $n$ bytes between two nodes is given by

$$T(n) = \text{overhead} + \text{channel time} + \text{routing delay}$$

where

- **overhead** = time in initiating or completing the transfer,
- **channel time** = $n/B$, where $B$ is bw. of the “thinnest” channel, and
- **routing delay** = $f(H, n)$, a function of the number of hops in the transfer.

Traditional data-communications are “store and forward” networks, where the data must arrive at one node before it can be forwarded to the next.

In parallel machines, the message “cuts through” several switches, without stopping at any of them.

It is like a train whose locomotive can traverse one switch before the last cars have finished traversing the previous switch.

What is the impact of cut-through routing?

Consider a network where the—

- max distance between any two nodes is $\log n$,
- total number of switches is $\alpha n \log n$ for a small constant $\alpha$, and
- overhead = 1 $\mu$s, bandwidth = 64 MB/s, routing delay = 200 ns per hop.

What is the channel time for a 128B transfer?

**Cut-through routing**
\[ T_{64}(128) = 1.0 \, \mu s + 2.0 \, \mu s + \_ \text{hops} \times \_ \, \mu s/\text{hop} = \_ \, \mu s \]

\[ T_{1024}(128) = 1.0 \, \mu s + 2.0 \, \mu s + \_ \text{hops} \times \_ \, \mu s/\text{hop} = \_ \, \mu s \]

**Store-and-forward routing**

\[ T^{sf}_{64}(128) = 1.0 \, \mu s + 6 \text{ hops} \times (2.0 + 0.2) \, \mu s /\text{hop} = 14.2 \, \mu s \]

\[ T^{sf}_{1024}(128) = 1.0 \, \mu s + 10 \text{ hops} \times (2.0 + 0.2) \, \mu s /\text{hop} = 23 \, \mu s \]

Actually, the situation is even worse than this, due to contention. Explain.

**Cost scaling**

[§7.1.3] The cost of a system can be divided into fixed and incremental components:

\[ \text{cost}(p, m) = \text{fixed cost} + \text{incremental cost}(p, m) \]

The fixed costs puts small systems at a cost/performance disadvantage.

Let us define \( \text{costup}(p) = \text{cost}(p) \div \text{cost}(1) \).

A system is cost-effective if \( \text{speedup}(p) > \text{costup}(p) \).
2048 processors, 475-fold speedup at 206 times the cost.

**Physical scaling**

[§7.1.2] How compact do nodes need to be?

There is really no consensus. Some machines have nodes that are little bigger than microprocessors; others have nodes that are a workstation chassis.

Technologies that support long distances (e.g., optical fiber) have a much larger fixed cost per link.

The nCUBE/2 is a good example of chip-level integration.
Notice that the network interface was integrated along with the rest of the processor. This was a custom processor, at 500K transistors, large for its time.

An example of board-level integration is the Thinking Machines CM-5 (ca. 1993).

The node comprised essentially the components of a 33MHz Sparc-based workstation.

The network interface was an additional ASIC on the Sparc MBUS.
The IBM Scalable Parallel System (SP-1 and SP-2) is a good example of a design employing system-level integration.

Its racks consist of several almost complete RS6000 workstations.

Large configurations are built by cabling several racks together, complete with additional switch racks.

Disks on compute nodes are typically used only for swap space and temporary storage.

Most I/O devices are concentrated on dedicated I/O nodes.