Supercomputing 2007, www.top500.org

#1: IBM Blue Gene
500 TFlops/sec
Main memory = 74 GB
213,000 processors.

Vendor:
- IBM 46%
- HP 33%

- Multicore processors (Intel Clovertown quad-core most popular)
  - 71% Intel
  - 16% AMD Opteron
  - 12% IBM power processors
- 81% cluster architecture
- Interconnects: Gigabit Ethernet, Infiniband

LINPACK, HPC benchmarks

(HPL) → LINPACK, linear system of equations
(DFGEMM) Floating pt rate for double precision matrix-matrix mult.
(STREAM) Sustainable memory BW for simple vector kernel
(PTRANS) Parallel matrix transpose
(Random Access) integer random updates of memory
(FFT) FFT
(B_EFF) latency & BW of simultaneous communication patterns
• CPU speed limitations $\rightarrow$ instruction execution rate
  \[ CPU \leftrightarrow \text{memory rate} \]
  $\rightarrow$ memory interleaving, cache
  $\rightarrow$ instruction and execution pipelining
  $\rightarrow$ superscalar execution: data/resource/branch dependencies
  $\rightarrow$ VLIW processors & IA-64
  Instructions that can be concurrently executed are packed.

---

**Figure 2.1** The evolution of a typical sequential computer: (a) a simple sequential computer; (b) a sequential computer with memory interleaving; (c) a sequential computer with memory interleaving and cache; and (d) a pipelined processor with $d$ stages.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

---

in a single long instruction word & executed on multiple functional units at same time

requires extensive compiler support
  - loop unrolling, branch prediction, speculative execution

$\rightarrow$ VLIW & superscalar processors:
  - exploit implicit parallelism
  - small scale of concurrency
SIMD

ILLiac IV, MPP, DAP,
CM-2, MasPar
MP-1 & MP-2

MIMD

Cosmic Cube, nCube,
IPSC, Symmetry,
FX-series
TC-2000, CM-5
KSR-1, Paragon XP/S

Figure 2.2  A typical SIMD architecture (a) and a typical MIMD architecture (b).

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

SIMD: Single Instruction stream  Multiple Data stream

MIMD: Multiple

SISD: Single  Single

MISD: Multiple

FLYNN's taxonomy.
if (B == 0)
    C = A;
else
    C = A/B;

(a)

Processor 0
A 5
B 0
C 0

Processor 1
A 4
B 2
C 0

Processor 2
A 1
B 1
C 0

Processor 3
A 0
B 0
C 0

Initial values

Processor 0
A 5
B 0
C 5

Processor 1
A 4
B 2
C 0

Processor 2
A 1
B 1
C 0

Processor 3
A 0
B 0
C 0

Step 1

Processor 0
A 5
B 0
C 5

Processor 1
A 4
B 2
C 2

Processor 2
A 1
B 1
C 1

Processor 3
A 0
B 0
C 0

Step 2

(b)

Figure 2.3 Executing a conditional statement on an SIMD computer with four processors: (a) The conditional statement; (b) The execution of the statement in two steps.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.

different processors cannot execute different instances in the same clock cycle.
distributed memory or private memory architecture

→ NUMA like

Figure 2.4  A typical message-passing architecture.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
- BW of interconnect must be substantial
  → conflicts
  → multiple stages of interconnect

- UMA vs. NUMA

- Cache coherence

![Diagram of shared-address-space architectures]

Figure 2.5 Typical shared-address-space architectures: (a) Uniform-memory-access shared-address-space computer; (b) Non-uniform-memory-access shared-address-space computer with local and global memories; (c) Non-uniform-memory-access shared-address-space computer with local memory only.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- Shared memory NUMA vs. msg-passing

  → NUMA provides HW support for R/W to remote memories
  
  msg-passing: remote access emulated by explicit msg-passing

- Easy to emulate msg-passing arch. by shared-memory arch.

- Reverse is difficult
• nonblocking network

• \( b \geq p \)

Figure 2.6 A completely nonblocking crossbar switch connecting \( p \) processors to \( b \) memory banks.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
Figure 2.7 A typical bus-based architecture with no cache (a) and with cache memory at each processor (b).

Copyright (r) 1994 Benjamin/Cummings Publishing Co.
- Crossbar: scalable to performance; not scalable to cost
- Shared bus: Not scalable to cost; scalable to cost

![Diagram](image)

**Figure 2.8** The schematic of a typical multistage interconnection network:

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

\[
\begin{aligned}
2 \times 2 \\
& \text{crossbars;} \\
\frac{p}{2} & \text{such}
\end{aligned}
\]

\[
\begin{aligned}
n = \log p (= m)
\end{aligned}
\]
Input \(i\), output \(j\)

\[
j = \begin{cases} 
2i & 0 \leq i \leq p/2 - 1 \\
2i + 1 - p & p/2 \leq i \leq p - 1 
\end{cases}
\]

Left-rotation on binary representation of \(i\)

Omega network: \(\frac{p}{2} \times \log p\) switching elements

![Diagram of Omega network](image)

Figure 2.10 A perfect shuffle interconnection for eight inputs and outputs.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
\( S_i \rightarrow S_j \text{ iff } j = i \oplus (2 \log p - l) \)

at level \( l \)

\[ j = i \oplus 2 \]

\[ j = i \oplus 100 \]

\[ j = i \oplus 010 \]

\[ j = i \oplus 001 \]

"Butterfly network" \( \rho = \# \text{ processors} \)

- Other networks
  -> banyan, Benes,

\( M = \frac{n}{2} \text{ switches/stage} \)

switch \( \langle x, s \rangle \), where \( x \in [0, M-1] \), stage \( s \in [0, \log_2 n - 1] \)

\[ \text{Figure 2.11 Two switching configurations of the} \]
\[ 2 \times 2 \text{ switch: (a) Pass-through; (b) Cross-over.} \]

Copyright \((c)\) 1994 Benjamin/Cummings Publishing Co.

- Benes
- Routing in stage \( i \) uses \( i \text{th} \) bit of destination.
- \( N \) network used in BBN Butterfly, IBM RP-3, NYU Ultracomputer.

\( e.g.: 001 \) to \( 010 \)

**Figure 2.12** A complete omega network connecting eight inputs and eight outputs.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.
Figure 2.13  An example of blocking in omega network: one of the messages (010 to 111 or 110 to 100) is blocked at link AB.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.
Baseline Network

\[ s = 1 \quad s = 2 \quad s = 3 \]

000
001
010
011
100
101
110
111

\[ j_1 = \sum_{i \in \mathcal{F}} \frac{i}{2^k} \]

\[ j_2 = \sum_{i \in \mathcal{F}} \frac{i}{2^k} + 2 \log p - s \]

\[ K = \log p - (s - 1) \]

\[ J_1 = \sum_{i \in \mathcal{F}} \frac{i}{2^k} \cdot 2^k + \left[ \frac{i \mod 2^k}{2} \right] \]

\[ J_2 = J_1 + 2 \]

\[ \bar{z} \in \]
Figure 2.14  A completely-connected network of eight processors (a), and a star-connected network of nine processors (b).
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Figure 2.15  A four-processor linear array (a) and a four-processor ring (b).
Copyright (r) 1994 Benjamin/Cummings Publishing Co.
- 2-D meshes: DAP, Paragon XP/S
- 3-D meshes: Gray T3D, J-machine

Figure 2.16 (a) A two-dimensional mesh with an illustration of routing a message from processor $P_1$ to processor $P_d$; (b) a two-dimensional wraparound mesh with an illustration of routing a message from processor $P_1$ to processor $P_d$; (c) a three-dimensional mesh.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.
communication bottleneck at higher levels.

Figure 2.17 Complete binary tree networks and message routing in them.
Copyright (c) 1994 Benjamin/Cummings Publishing Co.

Figure 2.18 A fat tree network of 16 processors.
Copyright (c) 1994 Benjamin/Cummings Publishing Co.
(1) $P_i = P_j$ if binary rep. differ in 1 bit position
(2) each proc. connected to $d$ other procs
(3) $d$-dim hypercube split into $2^{(d-1)}$-dim hypercubes in $d$ ways
(4) By fixing $k/d$ bits, we have $(d-k)$-dim HC
How many such HCs? $2^k$

2D cube

$s = 01$

$d = 10$

$\oplus 11$

$\oplus 10$

0-D hypercube

1-D hypercube

2-D hypercube

3-D hypercube

4-D hypercube

$d$-dim HC

2 Proc.

$d$ bit address

$d'$ links

Figure 2.19 Hypercube-connected architectures of zero, one, two, three, and four dimensions. The figure also illustrates routing of a message from processor 0101 to processor 1011 in a four-dimensional hypercube.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.

(5) # communic links in shortest path = Hamming distance.

$s + t$: route along dimensions where the corr. bit position = 1

e.g. nCUBE 2, Cosmic Cube, iPSC
Figure 2.20  Three distinct partitions of a three-dimensional hypercube into two two-dimensional cubes. Links connecting processors within a partition are indicated by bold lines.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- $k$-ary $d$-cube networks
  - radix
    - $r$: # processors along each dimension
- $d$-dim hypercube $\equiv$ binary $d$-cube
- ring of $p$ processors $\equiv p$-ary $1$-cube
- 2D wraparound mesh of $p$ proce $\equiv k$-ary $2$-cube
- $k$-ary $d$-cube $\equiv$ constructed from $k$ $k$-ary $(d-1)$-cubes by connecting the processors that occupy identical positions in the cubes into rings.
Figure 2.21 The two-dimensional subcubes of a four-dimensional hypercube formed by fixing the two most significant label bits (a) and the two least significant bits (b). Processors within a subcube are connected by bold lines.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.
Table 2.1 A summary of the characteristics of various static network topologies connecting \( p \) processors.

<table>
<thead>
<tr>
<th>Network</th>
<th>Diameter</th>
<th>Bisection Width</th>
<th>Arc Connectivity</th>
<th>Cost (No. of links)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completely-connected</td>
<td>1</td>
<td>( p^2/4 )</td>
<td>( p - 1 )</td>
<td>( p(p - 1)/2 )</td>
</tr>
<tr>
<td>Star</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>( p - 1 )</td>
</tr>
<tr>
<td>Complete binary tree</td>
<td>( 2 \log((p + 1)/2) )</td>
<td>1</td>
<td>1</td>
<td>( p - 1 )</td>
</tr>
<tr>
<td>Linear array</td>
<td>( p - 1 )</td>
<td>1</td>
<td>1</td>
<td>( p - 1 )</td>
</tr>
<tr>
<td>Ring</td>
<td>( \lfloor p/2 \rfloor )</td>
<td>2</td>
<td>2</td>
<td>( p )</td>
</tr>
<tr>
<td>2-D mesh without wraparound</td>
<td>( 2(p - \sqrt{p}) )</td>
<td>( \sqrt{p} )</td>
<td>2</td>
<td>( 2(p - \sqrt{p}) )</td>
</tr>
<tr>
<td>2-D wraparound mesh</td>
<td>( 2\sqrt{p}/2 )</td>
<td>( 2\sqrt{p} )</td>
<td>4</td>
<td>( 2p )</td>
</tr>
<tr>
<td>Hypercube</td>
<td>( \log p )</td>
<td>( p/2 )</td>
<td>( \log p )</td>
<td>( (p \log p)/2 )</td>
</tr>
<tr>
<td>Wraparound k-ary d-cube</td>
<td>( d \lfloor k/2 \rfloor )</td>
<td>( 2k^{d-1} )</td>
<td>2( d )</td>
<td>( dp )</td>
</tr>
</tbody>
</table>

Connectivity = measure of multiplicity of paths between any two processes. Arc connectivity = min. # arcs that must be removed to break network into 2 parts. Bisection width = min. # links that have to be removed to partition network into 2 equal halves.

\[ \text{bisection BW} = \text{bisection width} \times \text{channel BW} \]

Diameter: \( \max \left[ \min_{ij} (i,j) \right] \)

Embedding networks into a hypercube:

- Without embedding, an algorithm designed for a specific graph may have to be adapted to another graph.

\((V, E) \rightarrow (V', E') : \max \# \text{ edges mapped to any edge in } E \equiv \text{congestion} \)

\[ \max \# \text{ links in } E' \text{ that any edge in } E \text{ is mapped onto} \]

\[ \frac{|V'|}{|V|} = \text{expansion} \]
Embedding Linear Array into Hypercube

processor \(i\) of linear array \(\rightarrow\) processor \(G(i, d)\) of HC

\[
\begin{align*}
G(0, i) &= 0 \\
G(1, i) &= 1 \\
G(i, x + 1) &= \begin{cases} 
G(i, x) & i < 2^x \\
2^x + G(2^x + 1 - i, x) & i \geq 2^x 
\end{cases}
\end{align*}
\]

1-bit Gray code
2-bit Gray code
3-bit Gray code
3-D hypercube
8-processor ring

Reflect along this line

(a)

Figure 2.22 A three-bit reflected Gray code ring (a) and its embedding into a three-dimensional hypercube (b).

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

\(G\) = binary reflected Gray code

\(G(i, d)\) = \(i^{th}\) entry in seq. of Gray codes of \(d\) bits

- mapping dilation = 1
- expansion = 1
\[ \text{GRAY} = BINARY \oplus (BINARY/2) \]

\[ \begin{align*}
\text{BINARY} &= 101 \\
\text{LSR} \quad &101 \\
\oplus &0101 \\
\hline
\text{GRAY} &= 111 \\
\end{align*} \]

\[ \begin{align*}
011 \oplus &0001 \\
\hline
&010
\end{align*} \]
Array $\rightarrow$ 2-D grid

If $\left\lfloor \frac{i}{\sqrt{p}} \right\rfloor$ is even:

$$i \rightarrow i$$

If $\left\lfloor \frac{i}{\sqrt{p}} \right\rfloor$ is odd:

$$i \rightarrow \left\lfloor \frac{i}{\sqrt{p}} \right\rfloor \sqrt{p} + \left( \left\lfloor \frac{i}{\sqrt{p}} \right\rfloor + 1 \right) \sqrt{p} - i$$

(red edges) 2-D grid $\rightarrow$ (blue edges) array

Expansion = 1
Dialation = $2\sqrt{p} - 1$
Congestion = $\sqrt{p} + 1$
Embedding a Mesh into a Hypercube

- \(2^r \times 2^s\) wraparound mesh \(\rightarrow\) \(2^{r+s}\) HC

- \((i,j)_{\text{mesh}} \rightarrow G(i,r) \parallel G(j,s)_{\text{HC}}\)

- dilatation = 1, congestion = 1

![Diagram of mesh and hypercube](image)

Figure 2.23 (a) A 4 x 4 mesh illustrating the mapping of mesh processors to processors in a four-dimensional hypercube; and (b) a 2 x 4 processor mesh embedded into a three-dimensional hypercube.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

- proc in same row mesh \(\rightarrow\) proc with labels having \(r\) identical MSBs in HC

- row mesh \(\rightarrow\) distinct subcube

- column mesh \(\rightarrow\) distinct subcube
Embedding Binary Tree into HC

- Proc only at leaf nodes

1. root $\rightarrow$ any proc $^\text{HC}$
2. $\forall$ node $m$ at depth $j$
   - $C_{\text{LEFT}}(m) \rightarrow$ same proc $^\text{HC}$ to which $m$ is mapped [let this be proc $i$]
   - $C_{\text{RIGHT}}(m) \rightarrow$ proc $^\text{HC}$ we invert bit $j$ of $i$
     
     $$= \text{proc } i \oplus 2^{(j-1)}$$

(a) ![Tree Diagram](image1.png)
(b) ![Cube Diagram](image2.png)

Figure 2.24 A tree rooted at processor 011 ($=3$) and embedded into a three-dimensional hypercube: (a) the organization of the tree rooted at processor 011, and (b) the tree embedded into a three-dimensional hypercube.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

- In above mapping, expansion $= 1$
Routing Mechanisms

- minimal vs. non-minimal
- deterministic vs. adaptive
- dimension-ordered routing vs. XY routing, E-cube routing

\( P_s \rightarrow P_d \).

At any intermediate proc \( P_i \):

\( \rightarrow \) find msg. along dimension corresponding least significant non-zero bit in \( P_i + P_d \).

---

Figure 2.25  Routing a message from processor \( P_s \) (010) to processor \( P_d \) (111) in a three-dimensional hypercube using E-cube routing.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

---

Communication Costs in Static I.N.

1) \( t_s \) (startup time): once per msg.

2) \( t_h \) (per hop time) = node latency \( \times \) latency in switch to determine which off buffer/channel to use.

3) \( t_w \) (per word xfer time) = \( \frac{1}{r} \), where \( r = \text{channel BW} \).
- Store-and-forward: \( t_{\text{comm}} = t_s + (m t_w + t_h) l = \Theta(ml) \)

- Cut-through routing: msg advanced from incoming \rightarrow\ outgoing link as it arrives

  * eg: wormhole routing
  * flow-control digits (flits) are pipelined

---

![Diagram](image.png)

Figure 2.26 Passing a message from processor \( P_0 \) to \( P_3 \) (a) through a store-and-forward communication network; (b) and (c) extending the concept to cut-through routing. The shaded regions represent the time that the message is in transit. The startup time associated with this message transfer is assumed to be zero.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

- Susceptibility to deadlock
  - How avoided?
    - E-cube routing & XY routing are deadlock-free

Cut-through: \( t_{\text{comm}} = t_s + l t_h + m t_w = \Theta(m + l) \)
Figure 2.27  An example of deadlock in a wormhole-routing network.
Copyright (c) 1994 Benjamin/Cummings Publishing Co.
Cost-Performance Tradeoffs

- p-proc + C has $\frac{\text{plog} p}{2}$ edges $\rightarrow \frac{\text{plog} p}{2} \text{ wires}$

- p-proc wraparound mesh has $\frac{4p}{2}$ edges $\rightarrow \frac{\text{plog} p}{2}$

- If cost $\propto$ wires, mesh $\omega(\frac{\log p}{4})$ wires/channel, costs $\frac{1}{\omega}$

  HC $\omega/1$ wire/channel

- Avg dist: mesh $= \frac{\sqrt{p}}{2}$; HC $= \frac{\log p}{2}$

- With cut-through, $\text{latency} = t_s + t_h \frac{\sqrt{p}}{2} + tw m$

  Mesh $= t_s + t_h \frac{\sqrt{p}}{2} + 4mt_w / \sqrt{p}$

  HC $= t_s + t_h \frac{(\log p)}{2} + tw m$

- For light load, mesh is better; heavy load, HC excels

[ANALYZE]

- If cost $\propto$ bisection width, repeat above analysis

  mesh: $t_s + t_h \frac{\sqrt{p}}{2} + 4mt_w / \sqrt{p}$

  HC: $t_s + t_h \frac{(\log p)}{2} + tw m$

$\Rightarrow$ For $p > 16$ & sufficiently large message sizes

  mesh outperforms HC $\rightarrow$ at light loads

  mesh comparable HC $\rightarrow$ at high loads