PDCS 96

An Empirical Comparison of Area-Universal and Other Parallel Computing Networks

Ronald I. Greenberg
Mathematical and Computer Sciences, Loyola University

Lee Guan
Electrical Engineering, University of Maryland
OUTLINE

- Introduction
  - Area Universality and Theoretical Advantages
  - Objective of Current Research

- Networks and routing model

- Equalizing the hardware cost
  - Bisection Width and Pin-out Constraints
  - The Constant Layout-area Constraint

- Performance comparisons
  - Latency and Throughput Comparisons
  - Future Improvements
INTRODUCTION

• **Area Universality** General purpose computers are desired to achieve cost efficiency. Area universal computer networks - can simulate any other network with the same area with at most a polylogarithmic slow down.

• **Area Universal Networks** Fat-tree and fat-pyramid. Fat-pyramid promising for better performance in the context of non-constant wire-delay.)

• **Objective of current research** To address questions left open: comparison of the area universal networks with other popular networks (mesh and hypercube) in practice.
Two commonly used mesh layouts: (a) A $4 \times 4$ mesh without wrap-around links. (b) A folded $4 \times 4$ torus system. Unidirectional folded torus has the same layout area as bidirectional mesh without wrap-around.
The normal layout of a binary hypercube. This layout is known to produce minimum layout area.
A butterfly fat-tree. A set of N processors are placed at the leaves, represented by circles; the squares are switches. Each connection drawn between a pair of switches or a processor and a switch represents a pair of oppositely directed links, each capable of transmitting one flit in unit time.
A fat-pyramid. This network is obtained by superposing hierarchical mesh connections on a butterfly fat-tree. The original fat-tree connections are represented by thin lines and the mesh connections by thick lines.
Networks and Routing - Wormhole Routing

- Wormhole routing
  - Worms composed of flits (flow control digits)
  - Forwarding as soon as enough flits received to tell dest
  - Messages are blocked in-place.

- Dead lock and its prevention
  - Hypercube: e-cube routing (MSB to LSB).
  - Mesh w/o wraparound: first in one dimension then the other.
  - Mesh w. wraparound: using virtual channel.
A worm snakes through a network. (Address length = 2 flits, data length = 4 flits)
Latency of store and forward routing ($T_{SF}$) versus wormhole routing ($T_{WH}$). $W/L$ is the message aspect ratio. $D$ is the number of nodes to traverse. $N_0$, $N_1$, $N_2$ are node numbers.
Equalizing Hardware Cost - Bisection Width

- Bisection width as measure of cost (Dally). Favors low dimensionality.

- Bisection width for the four networks
  - Mesh $\sqrt{n}$
  - Hypercube $n/2$
  - Butterfly fat-tree $\sqrt{n}$
  - Fat-pyramid $\sqrt{n \log_{16} 4n}$
Equalizing Hardware Cost- Bisection Width

<table>
<thead>
<tr>
<th>n</th>
<th>mesh</th>
<th>hypercube</th>
<th>BFT</th>
<th>fat-pyramid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td>W</td>
<td>B</td>
<td>W</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>32</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>64</td>
<td>8</td>
<td>32</td>
<td>32</td>
<td>8</td>
</tr>
<tr>
<td>256</td>
<td>16</td>
<td>32</td>
<td>128</td>
<td>4</td>
</tr>
<tr>
<td>1024</td>
<td>32</td>
<td>32</td>
<td>512</td>
<td>2</td>
</tr>
<tr>
<td>4096</td>
<td>64</td>
<td>32</td>
<td>2048</td>
<td>1</td>
</tr>
</tbody>
</table>

The bisection width with channel width 1 and the channel width to maintain constant bisection width across different networks.
Equalizing Hardware Cost - Pin-out

- Pin-out as measure of cost (Abraham & Padmanabhan). Favors high dimensionality

- Total pin-outs for the four networks
  - Mesh: \(4(n - \sqrt{n})\)
  - Hypercube: \(n \log_2 n\)
  - Butterfly fat-tree: \((n + 3(n - \sqrt{n}))\)
  - Fat-pyramid: \((n + 5(n - \sqrt{n}) - \sqrt{n \log_2 n})\)
The pin-out with channel width 1 and the channel width to maintain constant pin-out across different networks.

<table>
<thead>
<tr>
<th>$n$</th>
<th>mesh</th>
<th>hypercube</th>
<th>BFT</th>
<th>fat-pyramid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PO</td>
<td>W</td>
<td>PO</td>
<td>W</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>32</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>32</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>256</td>
<td>256</td>
<td>32</td>
<td>256</td>
<td>32</td>
</tr>
<tr>
<td>1024</td>
<td>1024</td>
<td>32</td>
<td>1024</td>
<td>32</td>
</tr>
<tr>
<td>4096</td>
<td>4096</td>
<td>32</td>
<td>4096</td>
<td>32</td>
</tr>
</tbody>
</table>
Equalizing Hardware Cost - Layout Area

- Layout area as measure of cost. More accurate measurement of chip area.

- Side length $S(n) = \sqrt{n \cdot d \cdot W \cdot P}$, where $W$ is channel width, $P$ is wire pitch, and $d$ is determined by:
  - Mesh $1$
  - Hypercube $\frac{2^{k+2} - (-1)^k - 3}{6}$
  - Butterfly fat-tree $\log_4 n$
  - Fat-pyramid $\frac{3}{2} \log_4 n$
## Equalizing Hardware Cost - Layout Area

<table>
<thead>
<tr>
<th>$n$</th>
<th>mesh</th>
<th>hypercube</th>
<th>BFT</th>
<th>fat-pyramid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$d$</td>
<td>$W$</td>
<td>$d$</td>
<td>$W$</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>32</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>32</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>256</td>
<td>1</td>
<td>32</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>1024</td>
<td>1</td>
<td>32</td>
<td>21</td>
<td>2</td>
</tr>
<tr>
<td>4096</td>
<td>1</td>
<td>32</td>
<td>42</td>
<td>1</td>
</tr>
</tbody>
</table>

The wire density per row/column and the channel width under constant layout area constraints.
**Performance Comparison - Measures**

- **Low-load Latency** as primary performance measure; which tends to limit performance in practice in fine-grained parallel systems.

- **Max throughput** important for throughput-sensitive applications.

- **Loadrate** measured in bits per cycle per node. Total throughput goes up with network size yet throughput per node decreases due to
  - Bisection width limit (mesh, BFT, fat-pyramid)
  - Pin-out/area constraints to reduce channel width (hypercube)
  - Increasing blocking of available channels

- **Latency** stays rather flat until saturates then increase rapidly
  
  Networks should be designed to operate at the flat portion
Performance Comparison - Bisection Width

Comparison of packet routing latency under constraint of equal bisection width.
Performance Comparison - Pin-out

Comparison of packet routing latency under constraint of equal pin-out.
Performance Comparison - Layout Area

Comparison of packet routing latency under constraint of equal interconnect area.
Performance Comparison - Summary

• Constant bisection width
  – Mesh has the best max throughput.
  – Hypercube has the least throughput and largest latency.
  – BFT tends to achieve best latency as $N > 256$.
  – Fat-pyramid not far behind but poorer than BFT and mesh at least for moderate network sizes (unit wire delay).

• Constant pin-out
  – Hypercube shows best throughput and average latency close to mesh and fat-pyramid ($N < 4096$), agrees with Abraham and Padmanabhan’s result (pin-out favors high dimensionality).
Performance Comparison - Summary

• **Constant pin-out** (Continued)
  
  – BFT and fat-pyramid show lower latency than mesh for large N.
  
  – BFT shows a little better performance than fat-pyramid (const delay).

• **Constant layout-area**
  
  – Hypercube tends to have worst latency.
  
  – Mesh has the best latency and max throughput.
  
  – BFT and fat-pyramid show performances in between probably substantially attributable to the use of the simple, basic layout shown in previous figures. Further study of better layout is desired.
Performance Comparison - Future Works

• *Improved Network Architecture* for BFT and fat-pyramid. (Area in linear to number of processors, Ref: Greenberg 94)

• *Better cost model* considering packaging hierarchy.

• *Non-unit wire delay* - fat pyramid vs BFT. Fat-pyramid bypasses top-level up-links, holding promise for reducing long message path.

• *Real algorithms* vs. random message patterns. Random message gives a measure of general capacity. Real Algorithms show application-specific performances, e.g., sample sorting (throughout-sensitive) and sparse-matrix dense-vector multiplication (latency-sensitive).