# Part VI

**Implementation Aspects**

<table>
<thead>
<tr>
<th>Part</th>
<th>Background and Motivation</th>
<th>Complexity and Models</th>
<th>Architectural Variations</th>
</tr>
</thead>
<tbody>
<tr>
<td>I:</td>
<td>1. Introduction to Parallelism</td>
<td>2. A Taste of Parallel Algorithms</td>
<td><strong>Part II</strong>: Extreme Models</td>
</tr>
<tr>
<td></td>
<td>3. Parallel Algorithm Complexity</td>
<td>4. Models of Parallel Processing</td>
<td>5. PRAM and Basic Algorithms</td>
</tr>
<tr>
<td>III:</td>
<td>9. Sorting on a 2D Mesh or Torus</td>
<td>10. Routing on a 2D Mesh or Torus</td>
<td><strong>Part III</strong>: Mesh-Based Architectures</td>
</tr>
<tr>
<td></td>
<td>11. Numerical 2D Mesh Algorithms</td>
<td>12. Other Mesh-Related Architectures</td>
<td>Data Movement on 2D Arrays</td>
</tr>
<tr>
<td></td>
<td>15. Other Hypercubic Architectures</td>
<td>16. A Sampler of Other Networks</td>
<td>The Hypercube Architecture</td>
</tr>
<tr>
<td>V:</td>
<td>17. Emulation and Scheduling</td>
<td>18. Data Storage, Input, and Output</td>
<td><strong>Part IV</strong>: Low-Diameter Architectures</td>
</tr>
<tr>
<td></td>
<td>19. Reliable Parallel Processing</td>
<td>20. System and Software Issues</td>
<td>Hypercubic and Other Networks</td>
</tr>
<tr>
<td></td>
<td>23. Data-Parallel SIMD Machines</td>
<td>24. Past, Present, and Future</td>
<td>Coordination and Data Access</td>
</tr>
<tr>
<td></td>
<td>25. Control-Parallel Systems and Conclusion</td>
<td>Robustness and Ease of Use</td>
<td><strong>Part VI</strong>: Implementation Aspects</td>
</tr>
</tbody>
</table>
About This Presentation

This presentation is intended to support the use of the textbook *Introduction to Parallel Processing: Algorithms and Architectures* (Plenum Press, 1999, ISBN 0-306-45970-1). It was prepared by the author in connection with teaching the graduate-level course ECE 254B: Advanced Computer Architecture: Parallel Processing, at the University of California, Santa Barbara. Instructors can use these slides in classroom teaching and for other educational purposes. Any other use is strictly prohibited. © Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
<th>Revised</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>Spring 2005</td>
<td>Fall 2008*</td>
<td></td>
</tr>
</tbody>
</table>

* Very limited update
VI Implementation Aspects

Study real parallel machines in MIMD and SIMD classes:
• Examine parallel computers of historical significance
• Learn from modern systems and implementation ideas
• Bracket our knowledge with history and forecasts

Topics in This Part

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Shared-Memory MIMD Machines</td>
</tr>
<tr>
<td>22</td>
<td>Message-Passing MIMD Machines</td>
</tr>
<tr>
<td>23</td>
<td>Data-Parallel SIMD Machines</td>
</tr>
<tr>
<td>24</td>
<td>Past, Present, and Future</td>
</tr>
</tbody>
</table>
21 Shared-Memory MIMD Machines

Learn about practical shared-variable parallel architectures:
  • Contrast centralized and distributed shared memories
  • Examine research prototypes and production machines

Topics in This Chapter

21.1 Variations in Shared Memory
21.2 MIN-Based BBN Butterfly
21.3 Vector-Parallel Cray Y-MP
21.4 Latency-Tolerant Tera MTA
21.5 CC-NUMA Stanford DASH
21.6 SCI-Based Sequent NUMA-Q
# 21.1 Variations in Shared Memory

<table>
<thead>
<tr>
<th>Central Main Memory</th>
<th>Single Copy of Modifiable Data</th>
<th>Multiple Copies of Modifiable Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>UMA</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BBN Butterfly</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cray Y-MP</td>
<td></td>
</tr>
<tr>
<td>Distributed Main Memory</td>
<td><strong>COMA</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>CC-NUMA</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tera MTA</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Sequent NUMA-Q</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 21.1** Classification of shared-memory hardware architectures and example systems that will be studied in the rest of this chapter.
C.mmp: A Multiprocessor of Historical Significance

Fig. 21.2 Organization of the C.mmp multiprocessor.
Shared Memory Consistency Models

Varying latencies makes each processor’s view of the memory different.

**Sequential consistency** (strictest and most intuitive); mandates that the interleaving of reads and writes be the same from the viewpoint of all processors. This provides the illusion of a FCFS single-port memory.

**Processor consistency** (laxer); only mandates that writes be observed in the same order by all processors. This allows reads to overtake writes, providing better performance due to out-of-order execution.

**Weak consistency** separates ordinary memory accesses from synch accesses that require memory to become consistent. Ordinary read and write accesses can proceed as long as there is no pending synch access, but the latter must wait for all preceding accesses to complete.

**Release consistency** is similar to weak consistency, but recognizes two synch accesses, called “acquire” and “release”, that sandwich protected shared accesses. Ordinary read/write accesses can proceed only when there is no pending acquire access from the same processor and a release access must wait for all reads and writes to be completed.
21.2 MIN-Based BBN Butterfly

Fig. 21.3 Structure of a processing node in the BBN Butterfly.

Fig. 21.4 A small 16-node version of the multistage interconnection network of the BBN Butterfly.
21.3 Vector-Parallel Cray Y-MP

Fig. 21.5 Key elements of the Cray Y-MP processor. Address registers, address function units, instruction buffers, and control not shown.
Cray Y-MP’s Interconnection Network

Fig. 21.6 The processor-to-memory interconnection network of Cray Y-MP.
21.4 Latency-Tolerant Tera MTA

Fig. 21.7
The instruction execution pipelines of Tera MTA.

Memory Internal Pipeline

Interconnection Network

Memory Internal Pipeline

Write Pool

Retry Pool

Write Pool

Issue Pool

Instr. Fetch

Write Pool

Memory Pool

Interconnection Network

Memory Pool

Write Pool

Write Pool

Write Pool

Write Pool

Write Pool

Write Pool

Write Pool

Write Pool
21.5 CC-NUMA Stanford DASH

Fig. 21.8 The architecture of Stanford DASH.
21.6 SCI-Based Sequent NUMA-Q

Fig. 21.9 The physical placement of Sequent’s quad components on a rackmount baseboard (not to scale)

Fig. 21.10 The architecture of Sequent NUMA-Q 2000.
Details of the IQ-Link in Sequent NUMA-Q

**Fig. 21.11**
Block diagram of the IQ-Link board.

**Fig. 21.12**
Block diagram of IQ-Link’s interconnect controller.
22 Message-Passing MIMD Machines

Learn about practical message-passing parallel architectures:
- Study mechanisms that support message passing
- Examine research prototypes and production machines

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.1 Mechanisms for Message Passing</td>
</tr>
<tr>
<td>22.2 Reliable Bus-Based Tandem NonStop</td>
</tr>
<tr>
<td>22.3 Hypercube-Based nCUBE3</td>
</tr>
<tr>
<td>22.4 Fat-Tree-Based Connection Machine 5</td>
</tr>
<tr>
<td>22.5 Omega-Network-Based IBM SP2</td>
</tr>
<tr>
<td>22.6 Commodity-Driven Berkeley NOW</td>
</tr>
</tbody>
</table>
22.1 Mechanisms for Message Passing

1. **Shared-Medium networks** (buses, LANs; e.g., Ethernet, token-based)
2. **Router-based networks** (aka direct networks; e.g., Cray T3E, Chaos)
3. **Switch-based networks** (crossbars, MINs; e.g., Myrinet, IBM SP2)

Fig. 22.1 The structure of a generic router.

Fig. 22.2 Example 4 × 4 and 2 × 2 switches used as building blocks for larger networks.
Router-Based Networks

Fig. 22.1 The structure of a generic router.
### Case Studies of Message-Passing Machines

**Fig. 22.3** Classification of message-passing hardware architectures and example systems that will be studied in this chapter.

<table>
<thead>
<tr>
<th>Coarse-Grain</th>
<th>Shared-Medium Network</th>
<th>Router-Based Network</th>
<th>Switch-Based Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tandem NonStop (Bus)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Berkeley NOW (LAN)</td>
<td></td>
<td>nCUBE3</td>
<td>TMC CM-5 IBM SP2</td>
</tr>
</tbody>
</table>

**Fine-Grain**

| Medium-Grain |                  |                      |                     |

| Router-Based Network |                      |                      |                     |

| Switch-Based Network |                      |                      |                     |
22.2 Reliable Bus-Based Tandem NonStop

Fig. 22.4 One section of the Tandem NonStop Cyclone system.
High-Level Structure of the Tandem NonStop

Fig. 22.5 Four four-processor sections interconnected by Dynabus+. 
22.3 Hypercube-Based nCUBE3

Fig. 22.6 An eight-node nCUBE architecture.
22.4 Fat-Tree-Based Connection Machine 5

Fig. 22.7 The overall structure of CM-5.
Processing Nodes in CM-5

Fig. 22.8
The components of a processing node in CM-5.
The Interconnection Network in CM-5

Fig. 22.9 The fat-tree (hyper-tree) data network of CM-5.
22.5 Omega-Network-Based IBM SP2

![Diagram of IBM SP series architecture]

High-performance switch

Input/output nodes

Host nodes

Fig. 22.10 The architecture of IBM SP series of systems.
The IBM SP2 Network Interface

Fig. 22.11 The network interface controller of IBM SP2.
The Interconnection Network of IBM SP2

Fig. 22.12  A section of the high-performance switch network of IBM SP2.
22.6 Commodity-Driven Berkeley NOW

NOW aims for building a distributed supercomputer from commercial workstations (high end PCs), linked via switch-based networks

Components and challenges for successful deployment of NOWs (aka clusters of workstations, or COWs)

Network interface hardware: System-area network (SAN), middle ground between LANs and specialized interconnections

Fast communication protocols: Must provide competitive parameters for the parameters in the LogP model (see Section 4.5)

Distributed file systems: Data files must be distributed among nodes, with no central warehouse or arbitration authority

Global resource management: Berkeley uses GLUnix (global-layer Unix); other systems of comparable functionalities include Argonne Globus, NSCP metacomputing system, Princeton SHRIMP, Rice TreadMarks, Syracuse WWVM, Virginia Legion, Wisconsin Wind Tunnel
23 Data-Parallel SIMD Machines

Learn about practical implementation of SIMD architectures:
• Assess the successes, failures, and potentials of SIMD
• Examine various SIMD parallel computers, old and new

### Topics in This Chapter

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>23.1</strong></td>
<td>Where Have All the SIMDs Gone?</td>
</tr>
<tr>
<td><strong>23.2</strong></td>
<td>The First Supercomputer: ILLIAC IV</td>
</tr>
<tr>
<td><strong>23.3</strong></td>
<td>Massively Parallel Goodyear MPP</td>
</tr>
<tr>
<td><strong>23.4</strong></td>
<td>Distributed Array Processor (DAP)</td>
</tr>
<tr>
<td><strong>23.5</strong></td>
<td>Hypercubic Connection Machine 2</td>
</tr>
<tr>
<td><strong>23.6</strong></td>
<td>Multiconnected MasPar MP-2</td>
</tr>
</tbody>
</table>
23.1 Where Have All the SIMDs Gone?

Associative or content-addressable memories/processors constituted early forms of SIMD parallel processing.

Fig. 23.1 Functional view of an associative memory/processor.
Search Functions in Associative Devices

*Exact match:* Locating data based on partial knowledge of contents

*Inexact match:* Finding numerically or logically proximate values

*Membership:* Identifying all members of a specified set

*Relational:* Determining values that are less than, less than or equal, etc.

*Interval:* Marking items that are between or outside given limits

*Extrema:* Finding the maximum, minimum, next higher, or next lower

*Rank-based:* Selecting $k$th or $k$ largest/smallest elements

*Ordered retrieval:* Repeated max- or min-finding with elimination (sorting)
Mixed-Mode SIMD/MIMD Parallelism

Fig. 23.2 The architecture of Purdue PASM.
23.2 The First Supercomputer: ILLIAC IV

Fig. 23.3 The ILLIAC IV computer (the inter-processor routing network is only partially shown).
23.3 Massively Parallel Goodyear MPP

Fig. 23.4 The architecture of Goodyear MPP.
Fig. 23.5
The single-bit processor of MPP.
23.4 Distributed Array Processor (DAP)

Fig. 23.6 The bit-serial processor of DAP.
DAP’s High-Level Structure

Fig. 23.7 The high-level architecture of DAP system.
23.5 Hypercubic Connection Machine 2

Fig. 23.8
The architecture of CM-2.
The Simple Processing Elements of CM-2

Fig. 23.9 The bit-serial ALU of CM-2.
23.6 Multiconnected MasPar MP-2

Fig. 23.10
The architecture of MasPar MP-2.
The Interconnection Network of MasPar MP-2

Fig. 23.11 The physical packaging of processor clusters and the 3-stage global router in MasPar MP-2.
The Processing Nodes of MasPar MP-2

Fig. 23.12 Processor architecture in MasPar MP-2.
24 Past, Present, and Future

Put the state of the art in context of history and future trends:
• Quest for higher performance, and where it is leading us
• Interplay between technology progress and architecture

Topics in This Chapter

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>24.1 Milestones in Parallel Processing</td>
<td>24.1</td>
</tr>
<tr>
<td>24.2 Current Status, Issues, and Debates</td>
<td>24.2</td>
</tr>
<tr>
<td>24.3 TFLOPS, PFLOPS, and Beyond</td>
<td>24.3</td>
</tr>
<tr>
<td>24.4 Processor and Memory Technologies</td>
<td>24.4</td>
</tr>
<tr>
<td>24.5 Interconnection Technologies</td>
<td>24.5</td>
</tr>
<tr>
<td>24.6 The Future of Parallel Processing</td>
<td>24.6</td>
</tr>
</tbody>
</table>
24.1 Milestones in Parallel Processing

1840s  Desirability of computing multiple values at the same time was noted in connection with Babbage’s Analytical Engine

1950s  von Neumann’s and Holland’s cellular computers were proposed

1960s  The ILLIAC IV SIMD machine was designed at U Illinois based on the earlier SOLOMON computer

1970s  Vector machines, with deeply pipelined function units, emerged and quickly dominated the supercomputer market; early experiments with shared memory multiprocessors, message-passing multicomputers, and gracefully degrading systems paved the way for further progress

1980s  Fairly low-cost and massively parallel supercomputers, using hypercube and related interconnection schemes, became available

1990s  Massively parallel computers with mesh or torus interconnection, and using wormhole routing, became dominant

2000s  Commodity and network-based parallel computing took hold, offering speed, flexibility, and reliability for server farms and other areas
24.2 Current Status, Issues, and Debates

Design choices and key debates:

**Architecture**
- General- or special-purpose system?
- SIMD, MIMD, or hybrid?

**Interconnection**
- Shared-medium, direct, or multistage?
- Custom or commodity?

**Routing**
- Oblivious or adaptive?
- Wormhole, packet, or virtual cut-through?

**Programming**
- Shared memory or message passing?
- New languages or libraries?

Related user concerns:

- Cost per MIPS
- Scalability, longevity
- Cost per MB/s
- Expandability, reliability
- Speed, efficiency
- Overhead, saturation limit
- Ease of programming
- Software portability
Fig. 24.1 Milestones in the Accelerated Strategic Computing Initiative (ASCI) program, sponsored by the US Department of Energy, with extrapolation up to the PFLOPS level.
24.4 Processor and Memory Technologies

Fig. 24.2 Key parts of the CPU in the Intel Pentium Pro microprocessor.
24.5 Interconnection Technologies

Fig. 24.3 Changes in the ratio of a 1-cm wire delay to device switching time as the feature size is reduced.
Intermodule and Intersystem Connection Options

Fig. 24.4 Various types of intermodule and intersystem connections.
System Interconnection Media

Fig. 24.5  The three commonly used media for computer and network connections.
24.6 The Future of Parallel Processing

In the early 1990s, it was unclear whether the TFLOPS performance milestone would be reached with a thousand or so GFLOPS nodes or with a million or so simpler MFLOPS processors.

The eventual answer was closer to $1K \times \text{GFLOPS}$ than $1M \times \text{MFLOPS}$.

PFLOPS performance was achieved in more or less the same manner.

**Asynchronous design** for its speed and greater energy economy: Because pure asynchronous circuits create some difficulties, designs that are locally synchronous and globally asynchronous are flourishing.

**Intelligent memories** to help remove the memory wall: Memory chips have high internal bandwidths, but are limited by pins in how much data they can handle per clock cycle.

**Reconfigurable systems** for adapting hardware to application needs: Raw hardware that can be configured via “programming” to adapt to application needs may offer the benefits of special-purpose processing.

**Network and grid computing** to allow flexible, ubiquitous access: Resource of many small and large computers can be pooled together via a network, offering supercomputing capability to anyone.
Parallel Processing for Low-Power Systems

Figure 25.1 of Parhami's computer architecture textbook.
Peak Performance of Supercomputers

Evolution of Computer Performance/Cost

Two Approaches to Parallel Processing

Architecture (topology) and algorithm considered in tandem

+ Potentially very high performance
  – Large effort to develop efficient algorithms

Users interact with topology-independent platforms

+ Less steep learning curve
+ Program portability
  – Sacrifice some performance for ease of use

Main examples:
  Virtual shared memory
  Message-passing interface

The topology-independent approach is currently dominant
**ABCs of Parallel Processing in One Slide**

**A  Amdahl’s Law (Speedup Formula)**

*Bad news* – Sequential overhead will kill you, because:

\[
\text{Speedup} = \frac{T_1}{T_p} \leq \frac{1}{f + \frac{(1 - f)}{p}} \leq \min(1/f, p)
\]

**Morale:** For \( f = 0.1 \), speedup is at best 10, regardless of peak OPS.

**B  Brent’s Scheduling Theorem**

*Good news* – Optimal scheduling is very difficult, but even a naive scheduling algorithm can ensure:

\[
\frac{T_1}{p} \leq T_p < \frac{T_1}{p} + T_{\infty} = \left(\frac{T_1}{p}\right)[1 + \frac{p}{(T_1/T_{\infty})}]
\]

**Result:** For a reasonably parallel task (large \( T_1/T_{\infty} \)), or for a suitably small \( p \) (say, \( p < T_1/T_{\infty} \)), good speedup and efficiency are possible.

**C  Cost-Effectiveness Adage**

*Real news* – The most cost-effective parallel solution may not be the one with highest peak OPS (communication?), greatest speed-up (at what cost?), or best utilization (hardware busy doing what?).

**Analogy:** Mass transit might be more cost-effective than private cars even if it is slower and leads to many empty seats.
Gordon Bell Prize for High-Performance Computing

Established in 1988 by Gordon Bell, one of the pioneers in the field

Prizes are given annually in several categories (there are small cash awards, but the prestige of winning is the main motivator)

The two key categories are as follows; others have varied over time

**Peak performance:** The entrant must convince the judges that the submitted program runs faster than any other comparable application

**Price/Performance:** The entrant must show that performance of the application divided by the list price of the smallest system needed to achieve the reported performance is better than that of any other entry

Different hardware/software combinations have won the competition

Supercomputer performance improvement has outpaced Moore’s law; however, progress shows signs of slowing down due to lack of funding

In past few years, improvements have been due to faster uniprocessors (2004 NRC Report *Getting up to Speed: The Future of Supercomputing*)
Recent Gordon Bell Prizes, November 2008

Performance in a Scientific Application: superconductor simulation
Oak Ridge National Lab: Cray XT Jaguar, 150,000 cores, 1.35 PFLOPS


Special Prize for Algorithm Innovation
Lawrence Berkeley National Lab: harnessing potential of nanostructures