Multiple Processor Systems

Chapter 8
Multiple Processor Systems

8.1 Multiprocessors
8.2 Multicomputers
8.3 Distributed systems

• Continuous need for faster computers
  – shared memory model
  – message passing multiprocessor
  – wide area distributed system

Multiprocessors

Definition:
A computer system in which two or more CPUs share full access to a common RAM

Multiprocessor Hardware (1)

Bus-based multiprocessors
Multiprocessor Hardware (2)

- UMA Multiprocessor using a crossbar switch

Multiprocessor Hardware (3)

- UMA multiprocessors using multistage switching networks can be built from 2x2 switches

(a) 2x2 switch  (b) Message format

Multiprocessor Hardware (4)

- Omega Switching Network

Multiprocessor Hardware (5)

NUMA Multiprocessor Characteristics
1. Single address space visible to all CPUs
2. Access to remote memory via commands
   - LOAD
   - STORE
3. Access to remote memory slower than to local
Multiprocessor Hardware (6)

(a) 256-node directory based multiprocessor
(b) Fields of 32-bit memory address
(c) Directory at node 36

Multiprocessor OS Types (1)

Each CPU has its own operating system

Multiprocessor OS Types (2)

Master-Slave multiprocessors

Multiprocessor OS Types (3)

- Symmetric Multiprocessors
  – SMP multiprocessor model
Multiprocessor Synchronization (1)

TSL instruction can fail if bus already locked

Multiprocessor Synchronization (2)

Multiple locks used to avoid cache thrashing

Multiprocessor Synchronization (3)

Spinning versus Switching
- In some cases CPU must wait
  - waits to acquire ready list
- In other cases a choice exists
  - spinning wastes CPU cycles
  - switching uses up CPU cycles also
  - possible to make separate decision each time locked mutex encountered

Multiprocessor Scheduling (1)

- Timesharing
  - note use of single data structure for scheduling
**Multiprocessor Scheduling (2)**

- Space sharing
  - multiple threads at same time across multiple CPUs

---

**Multiprocessor Scheduling (3)**

- Problem with communication between two threads
  - both belong to process A
  - both running out of phase

---

**Multiprocessor Scheduling (4)**

- **Solution: Gang Scheduling**
  1. Groups of related threads scheduled as a unit (a gang)
  2. All members of gang run simultaneously
     - on different timeshared CPUs
  3. All gang members start and end time slices together

---

**Multiprocessor Scheduling (5)**

Gang Scheduling
Multicomputers

• Definition:
  
  Tightly-coupled CPUs that do not share memory

• Also known as
  – cluster computers
  – clusters of workstations (COWs)

Multicomputer Hardware (1)

• Interconnection topologies
  (a) single switch
  (b) ring
  (c) grid
  (d) double torus
  (e) cube
  (f) hypercube

Multicomputer Hardware (2)

• Switching scheme
  – store-and-forward packet switching

Multicomputer Hardware (3)

Network interface boards in a multicomputer
Low-Level Communication Software (1)

- If several processes running on node
  - need network access to send packets …
- Map interface board to all process that need it
- If kernel needs access to network …
- Use two network boards
  - one to user space, one to kernel

User Level Communication Software

- Minimum services provided
  - send and receive commands
- These are blocking (synchronous) calls

Remote Procedure Call (1)

- Steps in making a remote procedure call
  - the stubs are shaded gray

Node to Network Interface Communication

- Use send & receive rings
- coordinates main CPU with on-board CPU
Remote Procedure Call (2)

Implementation Issues
- Cannot pass pointers
  - call by reference becomes copy-restore (but might fail)
- Weakly typed languages
  - client stub cannot determine size
- Not always possible to determine parameter types
- Cannot use global variables
  - may get moved to remote machine

Distributed Shared Memory (1)

- Note layers where it can be implemented
  - hardware
  - operating system
  - user-level software

Distributed Shared Memory (2)

Replication
(a) Pages distributed on 4 machines
(b) CPU 0 reads page 10
(c) CPU 1 reads page 10

Distributed Shared Memory (3)

- False Sharing
- Must also achieve sequential consistency
Multicomputer Scheduling
Load Balancing (1)

- Graph-theoretic deterministic algorithm

Load Balancing (2)

- Sender-initiated distributed heuristic algorithm
  - overloaded sender

Load Balancing (3)

- Receiver-initiated distributed heuristic algorithm
  - under loaded receiver

Comparison of three kinds of multiple CPU systems

<table>
<thead>
<tr>
<th>Item</th>
<th>Multiprocessor</th>
<th>Multicomputer</th>
<th>Distributed System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node configuration</td>
<td>CPU</td>
<td>CPU, RAM, net</td>
<td>Complete computer</td>
</tr>
<tr>
<td>Node peripherals</td>
<td>All shared</td>
<td>Shared exc. maybe disk</td>
<td>Full set per node</td>
</tr>
<tr>
<td>Location</td>
<td>Same rack</td>
<td>Same mm</td>
<td>Possibly workable</td>
</tr>
<tr>
<td>Internode communication</td>
<td>Shared RAM</td>
<td>Dedicated interconnect</td>
<td>Traditional network</td>
</tr>
<tr>
<td>Operating systems</td>
<td>One, shared</td>
<td>Multiple, same</td>
<td>Possibly all different</td>
</tr>
<tr>
<td>File systems</td>
<td>One, shared</td>
<td>One, shared</td>
<td>Each node has own</td>
</tr>
<tr>
<td>Administration</td>
<td>One organization</td>
<td>One organization</td>
<td>Many organizations</td>
</tr>
</tbody>
</table>
Distributed Systems (2)

- Common base for applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Middleware</th>
<th>Windows</th>
<th>Pentium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Middleware</td>
<td>Linux</td>
<td>Pentium</td>
</tr>
<tr>
<td>Application</td>
<td>Middleware</td>
<td>Solaris</td>
<td>SPARC</td>
</tr>
<tr>
<td>Application</td>
<td>Middleware</td>
<td>Mac OS</td>
<td>Macintosh</td>
</tr>
</tbody>
</table>

Achieving uniformity with middleware

Network Hardware (1)

- Ethernet
  - (a) classic Ethernet
  - (b) switched Ethernet

Network Hardware (2)

The Internet

Network Services and Protocols (1)

<table>
<thead>
<tr>
<th>Service</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliable message stream</td>
<td>Sequence of pages of a book</td>
</tr>
<tr>
<td>Reliable byte stream</td>
<td>Remote login</td>
</tr>
<tr>
<td>Unreliable connection</td>
<td>Digitized voice</td>
</tr>
<tr>
<td>Unreliable datagram</td>
<td>Network test packets</td>
</tr>
<tr>
<td>Acknowledged datagram</td>
<td>Registered mail</td>
</tr>
<tr>
<td>Request-reply</td>
<td>Database query</td>
</tr>
</tbody>
</table>

Network Services
Network Services and Protocols (2)

- Internet Protocol
- Transmission Control Protocol
- Interaction of protocols

Document-Based Middleware (1)

- The Web
  - a big directed graph of documents

Document-Based Middleware (2)

How the browser gets a page

1. Asks DNS for IP address
2. DNS replies with IP address
3. Browser makes connection
4. Sends request for specified page
5. Server sends file
6. TCP connection released
7. Browser displays text
8. Browser fetches, displays images

File System-Based Middleware (1)

- Transfer Models
  - (a) upload/download model
  - (b) remote access model
**File System-Based Middleware (2)**

- **Naming Transparency**
  - (b) Clients have same view of file system
  - (c) Alternatively, clients with different view

**File System-Based Middleware (3)**

- **Semantics of File sharing**
  - (a) single processor gives sequential consistency
  - (b) distributed system may return obsolete value

**File System-Based Middleware (4)**

- **AFS – Andrew File System**
  - workstations grouped into cells
  - note position of venus and vice

**Shared Object-Based Middleware (1)**

- **Main elements of CORBA based system**
  - Common Object Request Broker Architecture
Shared Object-Based Middleware (2)

- Scaling to large systems
  - replicated objects
  - flexibility
- Globe
  - designed to scale to a billion users
  - a trillion objects around the world

Globe structured object

Shared Object-Based Middleware (4)

- A distributed shared object in Globe
  - can have its state copied on multiple computers at once

Internal structure of a Globe object
Coordination-Based Middleware (1)

- Linda
  - independent processes
  - communicate via abstract tuple space
- Tuple
  - like a structure in C, record in Pascal

1. Operations: out, in, read, eval

Coordination-Based Middleware (2)

Publish-Subscribe architecture

Coordination-Based Middleware (3)

- Jini - based on Linda model
  - devices plugged into a network
  - offer, use services
- Jini Methods
  1. read
  2. write
  3. take
  4. notify