PARALLEL PROCESSING

- Numerical problems and algorithms
- Impact of Amdahl’s law
- Parallel computer architectures
  - Memory architectures
  - Interconnection networks
- Flynn’s taxonomy (SIMD, MIMD, etc.)
- Shared-memory multiprocessors
- Distributed-memory, message-passing multicomputers
COMPUTATIONAL PROBLEMS

- Numerical simulation has become as important as experimentation
  - Permits exploration of a much larger region in parameter space
  - All-numerical designs

- The most CPU-intensive application areas include:
  - Computational fluid dynamics
  - Computational electromagnetics
  - Molecular modeling

- Typical problem sizes (in double-precision floating-point operations) and execution times (in CPU-days @ 10^9 FLOPS):

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Execution time (CPU-days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{14}</td>
<td>1.16</td>
</tr>
<tr>
<td>10^{15}</td>
<td>11.57</td>
</tr>
<tr>
<td>10^{16}</td>
<td>115.7</td>
</tr>
<tr>
<td>10^{17}</td>
<td>1157</td>
</tr>
</tbody>
</table>
Most of the important CPU-intensive applications involve the solution of a system of coupled partial differential equations.

- **Initial-value problems**
  - Time-dependent wave propagation (electromagnetics, geophysics, optical networking)
  - Time-dependent dynamics (fluid flow, mechanical engineering)

- **Boundary-value problems**
  - Typically require solution of very large systems of linear equations
  - Quasi-static electromagnetic problems
  - Static mechanical engineering (finite element analysis)
  - Eigenvalue problems (molecular energy surfaces)
PROPAGATION EQUATIONS FOR 4-WAVE MIXING

- Paraxial wave equation:

\[
\left[ \frac{\partial}{\partial z'} + \left( i \frac{\beta_2}{2} \frac{\partial^2}{\partial t'^2} - \frac{\beta_3}{6} \frac{\partial^3}{\partial t'^3} \right) \right] \mathcal{F}_n = -\frac{\alpha}{2} \mathcal{F}_n + i \frac{16\pi^2 \omega_n \chi^{(3)}}{n_0^2 A_e c^2} \mu_{nklm} D_{|k|,|l|,|m|} \\
\times \mathcal{F}_k \mathcal{F}_l \mathcal{F}_m e^{i(\Delta \beta_{nklm})z'}
\]

- Two regimes to study:
  - Three strong waves \((\mathcal{F}_k, \mathcal{F}_l, \mathcal{F}_m)\) generate nine weak waves \((\mathcal{F}_n)\)
    - Useful for estimating crosstalk among channels
  - Parametric coupling of three strong waves
    - Leads to coherent amplification of some channels and de-amplification of others
PULSE PROPAGATION IN AN OPTICAL FIBER (1)
A centralized, shared-memory multiprocessor
Layers where shared memory can be implemented

1. Application
2. Language run-time system
3. Operating system
4. Hardware

Shared memory
A distributed-memory multicomputer
Interconnection topologies

star

full interconnection

tree

ring

grid

double torus

cube

4-D hypercube
Interconnection network consisting of a 4-switch square grid
Deadlock in a circuit-switched interconnection network
Two-Hypernode Convex Exemplar System
### Typical access times to retrieve a word from a remote memory

<table>
<thead>
<tr>
<th>Machine</th>
<th>Communication mechanism</th>
<th>Interconnection network</th>
<th>Processor count</th>
<th>Typical remote memory access time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARCCenter</td>
<td>Shared memory</td>
<td>Bus</td>
<td>(\leq 20)</td>
<td>1 (\mu)s</td>
</tr>
<tr>
<td>SGI Challenge</td>
<td>Shared memory</td>
<td>Bus</td>
<td>(\leq 36)</td>
<td>1 (\mu)s</td>
</tr>
<tr>
<td>Cray T3D</td>
<td>Shared memory</td>
<td>3D torus</td>
<td>32–2048</td>
<td>1 (\mu)s</td>
</tr>
<tr>
<td>Convex Exemplar</td>
<td>Shared memory</td>
<td>Crossbar + ring</td>
<td>8–64</td>
<td>2 (\mu)s</td>
</tr>
<tr>
<td>KSR-1</td>
<td>Shared memory</td>
<td>Hierarchical ring</td>
<td>32–256</td>
<td>2–6 (\mu)s</td>
</tr>
<tr>
<td>CM-5</td>
<td>Message passing</td>
<td>Fat tree</td>
<td>32–1024</td>
<td>10 (\mu)s</td>
</tr>
<tr>
<td>Intel Paragon</td>
<td>Message passing</td>
<td>2D mesh</td>
<td>32–2048</td>
<td>10–30 (\mu)s</td>
</tr>
<tr>
<td>IBM SP-2</td>
<td>Message passing</td>
<td>Multistage switch</td>
<td>2–512</td>
<td>30–100 (\mu)s</td>
</tr>
<tr>
<td>Application</td>
<td>Scaling of computation</td>
<td>Scaling of communication</td>
<td>Scaling of computation-to-communication</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------</td>
<td>--------------------------</td>
<td>-----------------------------------------</td>
<td></td>
</tr>
<tr>
<td>FFT</td>
<td>$\frac{n \log n}{p}$</td>
<td>$\frac{n}{p}$</td>
<td>$\log n$</td>
<td></td>
</tr>
<tr>
<td>LU</td>
<td>$\frac{n}{p}$</td>
<td>$\sqrt[4]{n}$</td>
<td>$\sqrt[4]{n}$</td>
<td></td>
</tr>
<tr>
<td>Barnes</td>
<td>$\frac{n \log n}{p}$</td>
<td>Approximately $\sqrt[4]{n (\log n)}$</td>
<td>Approximately $\sqrt[4]{n}$ $\sqrt[4]{p}$</td>
<td></td>
</tr>
<tr>
<td>Ocean</td>
<td>$\frac{n}{p}$</td>
<td>$\sqrt[4]{n}$</td>
<td>$\sqrt[4]{n}$</td>
<td></td>
</tr>
</tbody>
</table>

Scaling of computation, of communication, and of the ratio are critical factors in determining performance on parallel machines.
Scalable vs. non-scalable systems
Amdahl’s law for parallel processing
AMDAHL’S LAW FOR PARALLEL COMPUTING

$f = \text{parallelizable fraction}$

$ \log_2 n$

($n = \text{number of processors}$)

efficiency

0

0.2

0.4

0.6

0.8

1.0

0

2

4

6

8

10

0.5
• Threads are the software equivalent of hardware functional units

• Properties of threads:
  ▶ Exist in user process space or kernel space
    ○ User threads are faster to launch than processes
  ▶ Mappings required for execution:
    user thread → lightweight process → kernel thread → CPU

• Reference: Bil Lewis and Daniel J. Berg, *Threads Primer* (SunSoft/Prentice Hall, 1996)
Computational paradigms

![Diagram of computational paradigms with nodes labeled P1, P2, P3, P4, P5, P6, P7, P8, P9, and various synchronization points. The diagram includes a work queue with nodes labeled P1, P2, and P3.]
Cache state transitions using signals from the processor:

- **Invalid (not valid cache block)**:
  - Processor read miss:
    - Read Only (clean)
  - Processor write miss:
    - Read/Write (dirty)
  - Processor read miss:
    - Write back dirty block to memory
    - Read/Write (dirty)

- **Read/Write (dirty)**:
  - Processor write:
    - Read/Write (dirty)
  - Processor write miss:
    - Invalid (not valid cache block)
  - Processor write:
    - Read Only (clean)

- **Read Only (clean)**:
  - Processor write:
    - Read Only (clean)

- **Another processor has a read miss or a write miss for this block (seen on bus); write back old block**

Cache state transitions using signals from the bus:

- **Invalid (not valid cache block)**
  - Invalid (not valid cache block)
  - Invalidate or another processor has a write miss for this block (seen on bus); write back old block

- **Read/Write (dirty)**
  - Read/Write (dirty)

- **Read Only (clean)**
  - Read Only (clean)

- **Another processor has a read miss or a write miss for this block (seen on bus)**
The MESI cache coherence protocol

(a) CPU 1
   A
   Exclusive
   Cache
   Bus

   CPU 2
   A
   Shared

   CPU 3

   Memory

   CPU 1 reads block A

(b) CPU 1
   A
   Shared
   Bus

   CPU 2
   A
   Shared

   CPU 3

   Memory

   CPU 2 reads block A

(c) CPU 1
   A
   Modified
   Bus

   CPU 2

   A

   CPU 3

   Memory

   CPU 2 writes block A

(d) CPU 1
   A
   Shared
   Bus

   CPU 2

   A

   CPU 3

   A

   Memory

   CPU 3 reads block A

(e) CPU 1
   A
   Modified
   Bus

   CPU 2

   A

   CPU 3

   Memory

   CPU 2 writes block A

(f) CPU 1
   A
   Modified
   Bus

   CPU 2

   A

   CPU 3

   Memory

   CPU 1 writes block A
Try to lock variable using swap: read lock variable and then set variable to locked value (1)

Begin update of shared data

Finish update of shared data

Unlock: set lock variable to 0
### Flynn’s taxonomy of parallel computers

<table>
<thead>
<tr>
<th>Instruction streams</th>
<th>Data streams</th>
<th>Name</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>SISD</td>
<td>Classical Von Neumann machine</td>
</tr>
<tr>
<td>1</td>
<td>Multiple</td>
<td>SIMD</td>
<td>Vector supercomputer, array processor</td>
</tr>
<tr>
<td>Multiple</td>
<td>1</td>
<td>MISD</td>
<td>Arguably none</td>
</tr>
<tr>
<td>Multiple</td>
<td>Multiple</td>
<td>MIMD</td>
<td>Multiprocessor, multicompiler</td>
</tr>
</tbody>
</table>
A taxonomy of parallel computers

Parallel computer architectures

- SISD
  - (Von Neumann)
  - Vector processor
- SIMD
  - Array processor
- MISD
  - Multi-processors
- MIMD
- Multi-computers
  - Multi-computers
  - UMA
    - Bus
  - COMA
    - Switched
  - NUMA
    - CC-NUMA
    - NC-NUMA
  - MPP
  - COW
    - Grid
    - Hypercube

Shared memory
Message passing