Parallel Computing Trends: from MPPs to NoWs
(from Massively Parallel Processors to Networks of Workstations)

Fall Research Forum
Oct 18th, 1994

Thorsten von Eicken
Department of Computer Science
Cornell University

Email: tve@cs.cornell.edu
Use of Parallel Computing

To provide CPU cycles and I/O in a distributed environment

Why not use Parallel Machines (MPPs)?

Parallel Machines have:
- high-bandwidth interconnect
- low communication latencies
- use fast commodity processors
- parallel languages (well... soon)

parallel machines use cheap commodity parts and package them in a richly interconnected box
MPPs: A Near Miss

“Near commodity” μprocs, DRAMs, and boards delays shipment

<table>
<thead>
<tr>
<th>MPP</th>
<th>Proc</th>
<th>Year</th>
<th>=WS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T3D</td>
<td>150Mhz Alpha</td>
<td>‘93/’94</td>
<td>‘92/’93</td>
</tr>
<tr>
<td>Paragon</td>
<td>50Mhz i860</td>
<td>‘92/’93</td>
<td>~ ‘91</td>
</tr>
<tr>
<td>CM-5</td>
<td>32Mhz SS-2</td>
<td>‘91/’92</td>
<td>‘89/’90</td>
</tr>
</tbody>
</table>

μproc performance improves 50%/yr (4%/month)
- 1 year lag: WS = 1.5x MPP node performance
- 2 year lag: WS = 2.25x MPP node performance

No economy of scale in 100s
=> + $

SW incompatibility (OS & apps)
=> + $$$$$
Price of 128 Sparcs

Includes:

- 128 50Mhz SuperSparc w. 1Mb cache
- 4GB DRAM (32Mb/proc)
- 134Gb Disk (1Gb/proc)
- 128 screens (native or xterms)
- fast switched network

**Xterms for MPP cost more than ATM for NoW**
Using the Server: Communications Spectrum

Log Latency

key strokes

voice

video

I/O bus

processor bus

human perception
~100ms

disk access
~10ms

context switch
~1ms

3-5x DRAM access
<1μs

Log Bandwidth

NATURAL I/O

FILE I/O

OBJECT SHARING

key strokes voice video I/O

bus processor

perception disk access context switch

~100ms ~10ms ~1ms

© Greg Papadopoulos @ Thinking Machines

Fall Research Forum — From MPPs to NoWs 5
Two Design Fixpoints

Globally sharable disk
- perform block I/O operations on other node's I/O bus
- bandwidth tracks disk (I/O bus) bandwidth
- latency dominated by transmission time for big blocks

Globally sharable DRAM
- perform cache-line size operations on other node's memory banks
- bandwidth tracks memory bank (processor bus) bandwidth
- latency tracks DRAM access times (3-5x)
Unfortunately, the Reality is...

Communication performance

<table>
<thead>
<tr>
<th></th>
<th>MPP</th>
<th>WS cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>bandwidth</td>
<td>10MB/sec</td>
<td>1MB/sec</td>
</tr>
<tr>
<td>round-trip</td>
<td>10μsec</td>
<td>5ms</td>
</tr>
<tr>
<td>message ovhd</td>
<td>1μs</td>
<td>1ms</td>
</tr>
</tbody>
</table>

— The End —
Is it possible to do better?

Raw performance
- cpu:
  - 100 SPECint, 100 SPECfp
  - >50MB/s I/O bus
- network:
  - 15MB/s switched net
  - 48-byte cell = 3.2μs latency

Off-the-shelf handicaps
- workstation
  - network interface is on I/O bus
  - no protected user-level access to NI
- operating system
  - no coordination among nodes
- ATM network
  - unreliable communication
  - no flow-control

Expectation: close to MPP performance but a small difference will remain

Trick: cut out the layers of software (without compromising security)
### Active Messages: an MPP Communication Architecture

<table>
<thead>
<tr>
<th>Message Passing</th>
<th>Global Name Space</th>
<th>Shared Memory</th>
<th>Data Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>compiler</td>
<td>run-time libraries</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Communication Architecture
- nCUBE
- WS Cluster
- CM5
- CS-2
- Paragon
- SP-1

- **Communication architecture functionality**
  - check validity of messages sent
  - demultiplex arriving messages
  - provide access to message data
  - notify destination processes on message arrival

- **Composition and Optimization**
  - Few+Simple+Fast Primitives
  - Compose primitives to realize progr. model
  - Optimize frequently occurring simple cases
  - Concentrate on performance
  - Provide clear cost model
**SSAM: SparcStation Active Messages Prototype**

**SEND:**
- select connection
- move data into buf
- add flow-control
- trap to kernel
- move data to NI

**SEND trap:**
- check valid connection
- xlate connection to VCI
- move data to NI

**POLL:**
- trap to kernel per message:
  - check flow-control
  - call handler
  - send ack (if needed)

**RECV trap:**
- map VCI to process
- move data from NI
SSAM Trap Cost

On SPARCstation-20 (60Mhz)

Note: null system call: 6.9μs on SS-20
Split-C remote read

- global pointers
  - struct node { ... } N;
  - struct node *global P = toglobal(&N);
- remote read
  - struct node my_N;
  - my_N = *P;
- implementation
  - send request Active Message to remote read-handler
  - read-handler sends reply Active Message with data to read-reply-handler
  - read-reply-handler stores data local and signals completion

Remote read timing
Comparison to Related Work

MPP performance
- SPARCstation-20 cluster vs CM-5
  - 2/3 the bandwidth
  - 4x the round-trip latency (incl. one switch)

Univ. Washington Remote Memory Access
- C.A. Thekkath, et.al, UW TR93-04-03
- remote read/write built into the kernel
- very similar hardware set-up
- similar level of engineering (kernel traps)
- basically same performance
- differences in flexibility, assumptions, requirements, etc...

<table>
<thead>
<tr>
<th>machine</th>
<th>peak bandwidth</th>
<th>round-trip latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP-1 + MPL/p</td>
<td>8.3MB/s</td>
<td>56μs</td>
</tr>
<tr>
<td>Paragon + NX</td>
<td>73MB/s</td>
<td>44μs</td>
</tr>
<tr>
<td>CM-5 + CMAML</td>
<td>10MB/s</td>
<td>12μs</td>
</tr>
<tr>
<td>SS-20 + SSAM</td>
<td>5.6MB/s</td>
<td>52μs†</td>
</tr>
</tbody>
</table>

† incl 1 switch
Summary & Conclusions

ATM workstation clusters show promise
- raw performance numbers look competitive with MPPs
- standard networking software looses most of the potential
- **but:** requires MPP-style communication architecture to deliver performance

Active Messages prototype demonstrates MPP-level performance
- must start from raw hardware and build-up
- must use direct traps to kernel to achieve protection at minimal cost
- must carefully orchestrate buffering to minimize data movement
- **result:** competitive with MPPs,
  difference to fastest MPPs can be explained by hardware limitations

Bottlenecks
- network interface on the I/O bus
- slow network switches (10µs/hop)
- no protection, no flow-control, no error detection in hardware