Faster!

Presenter: Saikat Guha

Cornell University

CS 614, Fall 2005
Now: Networks of Workstations
- Aggregate DRAM
- Multiple CPUs
- Network as I/O backplane

Cluster Computing: Commodity supercomputing

Gigabit network interconnects
- Ethernet, ARP, IP ... solved problem. Right?
Down with IP!

- Cluster computing
  - Few (thousands of) hosts
  - Simple, small topology
  - Network packet = function call
- IP solves a different problem
  - Global inter network
  - Planetary scale, multi-hop
  - IP data generally interactive, or bulk
Down with IP!

- Baked into the kernel
  - Death by contention (Ethernet)
  - Death by congestion (ARP)
  - Death by latency (IP)
  - Death by processing overhead (Kernel)

- ATM to the rescue
  - Circuit switched
  - Low maximum overhead (high minimum overhead)
    - ATM: 10%
    - Ethernet: 30%
  - Supported by kernels ... as IP over ATM. D’oh!
Look Ma, no kernel!

- By the power of: $\mu$-Kernel
  - sans user-space FS
  - sans user-space VM
  - sans all but user-space networking

**U-Net: A User-Level Network Interface for Parallel and Distributed Computing**

Thorsten von Eicken, Anindya Basu, Vineet Buch and Werner Vogels, Cornell University
Back to the Future

- Zero-copy, *true* Zero-copy
  - Shared buffer (IO-Lite ’99)
- Multiplex Network Interface (Exokernel ’95)
- Input and Output queues (SEDA ’01)
- Save on context switches (L$^4$ ’97)
U-Net is born

- User app makes syscall, creates endpoint
- Setup (ATM-like) channels to demultiplex
- Get a user-kernel (or user-hardware) shared buffer
- Compose data in buffer, send scatter-gather descriptor to Tx queue
- Trap to kernel
- For receive, poll or register upcall
U-Net, fantastic! Fore, not so much.

Figure 8: TCP bandwidth as a function of data generation by the application.

Figure 9: UDP and TCP round-trip latencies as a function of message size.
Long live U-Net

- Restricts user application
  - U-Net with buffer management ’97. Welsh et al.
- Scalable?
  - Connections
  - Nodes
  - Interfaces
- Reinvent the wheel
  - Naming, Routing, Discovery
  - Reliability, QoS
Does it really matter?
  • Cross-machine RPC: 0.6% – 5.3%
  • Are nodes still slower than networks?

LRPC saves the world
  • Exploit machine-local RPC (> 94%)
  • Reduce message copies
  • Reduce scheduling lag

**Lightweight Remote Procedure Call**

Brian Bershad, Thomas Anderson, Edward Lazowska, Henry Levy, UWash
Copy-happy RPC

- Stub generation
- Buffer Overhead
- Context Switch $\times 2$
- Scheduling $\times 2$

Faster! CS 614, Fall 2005
LRPC. Or perhaps just, $PC$

No scheduling required, just switch.
Context Switch be Gone

- Optimization for multiprocessors
  - Cache contexts on idle processor
  - Instead of context switch, run cached proc.
  - Saves on TLB misses, cache misses etc

- No pessimization for remote calls
  - Fallback to real RPC
  - for complex local calls too
Proof by Numbers

### Table IV. LRPC Performance of Four Tests (in microseconds)

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
<th>LRPC/MP</th>
<th>LRPC</th>
<th>Taos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>The Null cross-domain call</td>
<td>125</td>
<td>157</td>
<td>464</td>
</tr>
<tr>
<td>Add</td>
<td>A procedure taking two 4-byte arguments and returning one 4-byte argument</td>
<td>130</td>
<td>164</td>
<td>480</td>
</tr>
<tr>
<td>BigIn</td>
<td>A procedure taking one 200-byte argument</td>
<td>173</td>
<td>192</td>
<td>539</td>
</tr>
<tr>
<td>BigInOut</td>
<td>A procedure taking and returning one 200-byte argument</td>
<td>219</td>
<td>227</td>
<td>636</td>
</tr>
</tbody>
</table>

### Table V. Breakdown of Time (in microseconds) for Single-Processor Null LRPC

<table>
<thead>
<tr>
<th>Operation</th>
<th>Minimum</th>
<th>LRPC overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modula2+ procedure call</td>
<td>7</td>
<td>—</td>
</tr>
<tr>
<td>Two kernel traps</td>
<td>36</td>
<td>—</td>
</tr>
<tr>
<td>Two context switches</td>
<td>66</td>
<td>—</td>
</tr>
<tr>
<td>Stubs</td>
<td>—</td>
<td>21</td>
</tr>
<tr>
<td>Kernel transfer</td>
<td>—</td>
<td>27</td>
</tr>
<tr>
<td>Total</td>
<td>109</td>
<td>48</td>
</tr>
</tbody>
</table>
Under the rug

- Memory management costs
  - Allocate A-stack at bind time
- Resource migration
- Server control of degree of concurrency