Lightweight Remote Procedure Call

Brian N. Berhad, Thomas E. Andreson, Edward D. Lazowska, and Henry M. Levy

Presented by Brent Dombrowski
The Observation

- Remote Procedure Calls are not going to remote machines
  - V System: 97% to same machine.
  - Taos: 94.5% to same machine.
  - Unix + NFS: Not much RPC at all.
Details

- Taos OS in Detail
  - Million plus cross domain RPC calls
    - 112 procedures called.
    - 95% to just 10 procedures.
    - 75% to just 3 procedures.
Details

• Data being transferred is not large.

Fig. 1. RPC size distribution.
RPC Common Usage

- To the same machine.
- Across domains.
- Small amounts of data.
RPC Overhead

- Stub overhead.
- Message buffer.
- Access validation.
- Message transfer.
- Scheduling.
- Context switch.
- Dispatch.
RPC Optimizations

- DASH eliminates kernel copying.
- Mach and Taos use handoff scheduling.
- Others eliminate buffer copying.
- SRC RPC most ambitious.
  - Trades safety for performance.
  - Globally shared buffers with single lock.
LRPC

• Design communication facility for the common case.
  • Between protection domains on the same machine.
• Model borrowed from protected procedure call.
Techniques

- Simple control transfer.
- Simple data transfer.
- Simple stubs.
- Design for concurrency.
Control Transfer

- Call made to kernel trap.
- Kernel:
  - Validates call.
  - Creates a link.
  - Dispatches client’s thread to server domain.
- Control and results return through kernel.
Data Transfer

• Normal RPC arguments are copied 4 times.
  • Stub to RPC message.
  • Client domain to kernel domain.
  • Kernel domain to server domain.
  • Message to server stack.
Data Transfer

- LRPC uses pair wise argument stacks (A-Stacks) for argument passing.
  - Allows for copying as needed.
  - Provides private channel.
  - Allows asynchronous changes.
Stubs

- Bridge procedure call and domain transfer.
- Stubs blur boundaries to reduce cost of crossing them.
- Auto generated in assembly.
Stubs

• Simple LRPC only needs 3 calls.
  • Into client stub.
  • Out of server.
  • Out of stub.
• Complex LRPC reverts to traditional RPC.
Concurrency

- Designed with shared memory multiprocessors in mind.
- Throughput increase by minimizing use of shared locks.
- Cache domain contexts on idle processors.
- Track domain switches and prod processors to spin in most used.
Performance

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>
Performance

Fig. 2. Call throughput on a multiprocessor.

Calls per Second

Number of Processors

LRPC Optimal
LRPC Measured

RPC Optimal
RPC Measured
# RPC vs LRPC

<table>
<thead>
<tr>
<th></th>
<th>RPC</th>
<th>LRPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call Stub</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Buffer Message</td>
<td>Yes</td>
<td>No - in A-stack</td>
</tr>
<tr>
<td>Validate Access</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Transfer Message</td>
<td>Yes</td>
<td>No - in A-stack</td>
</tr>
<tr>
<td>Schedule Server</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Context Switch</td>
<td>Yes</td>
<td>No - Reuse current</td>
</tr>
<tr>
<td>Sever Stub</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Transfer Message</td>
<td>Yes</td>
<td>No - in A-stack</td>
</tr>
<tr>
<td>Validate Access</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
LRPC Shortcuts

- Both the client and server have access to the same memory and can use that in place of message passing.

- The client will block waiting for the server so the clients thread can be used to run the server.

- Some access validation still needed due to multiple threads and exception handling.
Summary

• Adopts common case approach.
• Uses the following when possible:
  • Simple control transfer.
  • Simple data transfer.
  • Simple stubs.
  • Multiprocessors.
Mystery

• Why use RPC to a local machine?
  • Why not a function call?
  • What was wrong with the protected procedure call?