Today’s Reminders

• Enroll in Piazza (31 have as of class time)
• Waitlist (33 currently enrolled)

Implementing Remote Procedure Calls

Phil Gibbons
15-712 F15
Lecture 3

Implementing Remote Procedure Calls

Andrew Birrell & Bruce Nelson 1984

• Andrew Birrell known for Grapevine (1981), with first distributed naming system (SigOps HoF paper); Dryad; was part of MSR-SV 2001-2014

• Bruce Nelson (CMU PhD’81), Dissertation “Remote Procedure Call”

• Awarded ACM Software System Award in 1994

SigOps HoF citation (2007): “This is THE paper on RPC, which has become the standard for remote communication in distributed systems and the Internet. The paper does an excellent job laying out the basic model for RPC and the implementation options.”

Remote Procedure Calls

• Procedure calls are a well-known & well-understood mechanism for transfer of program control and data

• Not a new idea: e.g., discussed in 1976 paper

• Major issues facing the designer of an RPC facility:
  - Precise semantics of a call in the presence of failures
  - Semantics of address-containing arguments in the absence of a shared address space
  - Integration of remote calls into existing programming systems
  - Binding (how caller determines location/identity of callee)
  - Suitable protocols for transfer of data & control between caller and callee
  - How to provide data integrity and security in an open communication network
Aims

• Make distributed computation easy

• “We wanted to make RPC communication highly efficient (within, say, a factor of five beyond the necessary transmission times of the network).”

• Make semantics of RPC package as powerful as possible, w/o loss of simplicity or efficiency

• Provide secure communication with RPC

Components of the System

Stubs

• User-stub and server-stub are automatically generated, using Mesa interface modules (basis for separate compilation)
  – Specification Interface: List of procedure names, together with the types of the arguments and results

• Lupine stub generator checks that user avoids specifying arguments or results that are incompatible with the lack of a shared address space

Binding

• Naming
  – Use Mesa interface module name + instance
Discussion of Binding

• Use Grapevine distributed database
  – Maps type to set of instances
  – Maps instance to network address

• Importing an interface has no effect on data structures

• Unique identifier means bindings broken on server crash

• Can only call explicitly exported procedures

• Grapevine enforces access control

• Options include importer specifies type and gets nearest

Packet-level Transport Protocol

• Up to factor of 10 faster than using general protocols
  – Not doing large data transfers
  – Goals: Minimize latency to get result & Server load under many users (minimize state info & handshaking costs)

• Guarantee: If call returns, procedure invoked exactly once

• Do not abort on server code deadlock or infinite loop

• When connection is idle, only a single machine-wide counter

• Rate of calls on ExportInterface in a single machine limited to an average rate of less than one per second

Packets Transmitted in Simple Call

• Arguments & Return result each fit in a single packet

• Caller periodically probes, and callee acks
  – Less work for server vs. pushing “I’m alive” messages
  – Callee can get “called failed” exception

Minimizing Process Swap Cost

• Maintain at each machine idle server processes ready to handle incoming packets (avoid process creates)

• Only 4 process swaps in each call

• Also, bypass SW layers for intranet RPCs
  – Modularity vs. performance trade-off
  – Today: RDMA
Performance

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Minimum</th>
<th>Median</th>
<th>Transmission</th>
<th>Local-only</th>
</tr>
</thead>
<tbody>
<tr>
<td>no args/results</td>
<td>1059</td>
<td>1097</td>
<td>131</td>
<td>9</td>
</tr>
<tr>
<td>1 arg/result</td>
<td>1070</td>
<td>1105</td>
<td>143</td>
<td>10</td>
</tr>
<tr>
<td>2 args/results</td>
<td>1077</td>
<td>1127</td>
<td>152</td>
<td>11</td>
</tr>
<tr>
<td>4 args/results</td>
<td>1115</td>
<td>1171</td>
<td>174</td>
<td>12</td>
</tr>
<tr>
<td>10 args/results</td>
<td>1222</td>
<td>1278</td>
<td>228</td>
<td>17</td>
</tr>
<tr>
<td>1 word array</td>
<td>1069</td>
<td>1111</td>
<td>131</td>
<td>10</td>
</tr>
<tr>
<td>4 word array</td>
<td>1106</td>
<td>1153</td>
<td>174</td>
<td>13</td>
</tr>
<tr>
<td>10 word array</td>
<td>1214</td>
<td>1250</td>
<td>229</td>
<td>16</td>
</tr>
<tr>
<td>40 word array</td>
<td>1643</td>
<td>1696</td>
<td>566</td>
<td>51</td>
</tr>
<tr>
<td>100 word array</td>
<td>2915</td>
<td>2936</td>
<td>1219</td>
<td>98</td>
</tr>
<tr>
<td>resume except’n</td>
<td>3574</td>
<td>3467</td>
<td>284</td>
<td>198</td>
</tr>
</tbody>
</table>

in microseconds

Alternatives

- Message passing – but procedure calls already in high-level languages
  - HPC community uses MPI (messages)
- Remote fork – would encounter same design problems
  - Also, what do you return, entire contents of memory??
- Distributed Shared Memory: need to extend address width, not as efficient, done at page granularity, not in their language (Mesa)
  - Long history of research into general DSM
  - Specialized DSM: key-value stores, parameter servers

Discussion

- Multicasting or broadcasting can be better than RPCs
- “At present it is hard to justify some of our insistence on good performance because we lack examples demonstrating the importance of such performance.”
- Today:
  - Remote Method invocation
  - Naming via key-value store containing hostname (or IP address) and port number
  - RPC over UDP/IP (unreliable, but more efficient); IP handles multi-packet arguments
  - Sun’s RPC (ONC RPC) system, with stub compiler rpcgen, is widely used, e.g., in Linux; provides XDR, a common data representation in messages
Using Threads in Interactive Systems: 
A Case Study

Carl Hauser, Christian Jacobi, Marvin Theimer, 
Brent Welch, Mark Weiser 1993