Remote Procedure Calls (RPC)

Xin Zheng – CS 614 Fall '07
(Some slides borrowed from Fall '05)

Outline

- What's RPC
- Cedar and Firefly RPC designs
- Performance measurements
- Conclusion

Network/Distributed Programs

- Computers/processes want to talk to each other
- Might be different machines in a network, or different address spaces in same machine

Communication

- Can design your own protocol

Custom Protocol

```java
public class Message {
    public Message(String type, String args) { /* */ }
}
```

Custom Protocol

```java
public class Communicator {
    public void send(Message m) { /* */ }
    public Message recv() { /* */ }
}
```
**Custom Protocol**

```java
public void joinGame(String gameName) {
    Message m = join(gameName, playerName);
    game_comm.send(m);
    Message reply = game_comm.recv();
    if (reply.getType() == SUCCESS) {
        ...
    } else if (reply.getType() == ERROR) {
        ...
    } else {...}
}
```

**Communication**

- Can design your own protocol
- OR...
- Realize that intra-process communication already happens all the time, wa procedure calls!

**A Very Simple Idea**

- Retain local procedure call semantics, but let procedures reside on different machines
- ...And you get RPC!

**A Very Simple Idea**

- Retain local procedure call semantics, but let procedures reside on different machines
- ...And you get RPC!

**Why RPC?**

- For programmers, nothing new to learn
- Distributed applications don’t have to look all that different from local programs
- Reducing extralinguistic clutter is always good... Pretty much all languages already support procedures/functions/methods
How It Works

Caller

```
foo("Hi there");
```

```
foo(s) {
    // stub
    return;
```

Callee

```
dispatcher() {
    // finds procedure
    foo(s);
    actual proc
    foo(s) {
        return "Howdy";
    }
    return;
```

Web Applications

- Server and client (browser) side components
- Client provides UI
- Server provides data, computation

Google Web Toolkit

- Java to JavaScript compiler
- Most of Gmail, Google Maps, etc written in GWT
- Provides higher level abstractions in Java compared to native JavaScript

Google Web Toolkit

```java
public interface ContactsService {
    public String[] getContacts(String s);
}
```

```java
public class ContactsServiceImpl implements ContactsService {
    public String[] getContacts(String s) {
        return new String[];
    }
}
```
Google Web Toolkit

```java
public interface ContactsService {
    // send_to.onchange event handler
    public void onChange(String s) {
        ContactsService svc = GWT.create(ContactsService);
        String[] contacts = svc.getContacts();
        ...
    }
}
```

Making RPC Fast

- Both Cedar and Firefly use custom protocols
- Skips traditional network stack
- Very platform specific
  - Firefly has some (a lot?) hand-tuned assembly code
  - Also relies a bit on multi-processors for performance

Cedar

- Minimize time between call and getting result
- Minimize load on servers
- Assume a large number of call with small amounts of data transfer
- Protocol defined at the packet level
- Implemented in Mesa

Simple Calls

- Client retransmits until ack received
  - Result acts as an ack
  - Similar for the callee: next call packet is a sufficient ack
- Callee maintains table for last call ID
  - Duplicate call packets can be discarded
  - This shared state acts as connection

Advantages

- No special connection establishment
- Low state requirements
  - Callee: only call ID table stored
  - Caller: single counter sufficient (for sequence number)
- No concern for state of connection – ping packets not required
- No explicit connection termination
**Large Calls**

**Firefly**
- Goes even further than Cedar in optimizations
- Protocol defined at the sub-packet level
  - Allows procedure stubs to access packets directly
  - Packet are reused when possible
- Written in Modula-2+/assembly

**Reusing Packets**
- Server stub can retain call packet for result
- Waiting thread contains packet buffer – this packet can be used for retransmission
- Packet buffers reside in memory shared by everyone
  - Security can be an issue

**Performance of Cedar**

<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 arg/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>1100</td>
<td>142</td>
<td>10</td>
</tr>
<tr>
<td>2 arg/results</td>
<td>1077</td>
<td>1100</td>
<td>142</td>
<td>11</td>
</tr>
<tr>
<td>4 arg/results</td>
<td>1115</td>
<td>1130</td>
<td>153</td>
<td>12</td>
</tr>
<tr>
<td>10 arg/results</td>
<td>1222</td>
<td>1250</td>
<td>157</td>
<td>17</td>
</tr>
<tr>
<td>1 word array</td>
<td>1008</td>
<td>1034</td>
<td>120</td>
<td>10</td>
</tr>
<tr>
<td>4 word array</td>
<td>1106</td>
<td>1125</td>
<td>132</td>
<td>13</td>
</tr>
<tr>
<td>10 word array</td>
<td>1214</td>
<td>1250</td>
<td>138</td>
<td>16</td>
</tr>
<tr>
<td>40 word array</td>
<td>1643</td>
<td>1685</td>
<td>566</td>
<td>51</td>
</tr>
<tr>
<td>100 word array</td>
<td>2915</td>
<td>2926</td>
<td>1219</td>
<td>98</td>
</tr>
<tr>
<td>resume except’n</td>
<td>2555</td>
<td>2637</td>
<td>284</td>
<td>134</td>
</tr>
<tr>
<td>unwind except’n</td>
<td>3574</td>
<td>3467</td>
<td>284</td>
<td>196</td>
</tr>
</tbody>
</table>

**Performance of Firefly**

<table>
<thead>
<tr>
<th># of caller threads</th>
<th>Calls to Null()</th>
<th>Calls to Mediator()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>seconds</td>
<td>RPC's/sec</td>
</tr>
<tr>
<td>1</td>
<td>26.61</td>
<td>355</td>
</tr>
<tr>
<td>2</td>
<td>16.80</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>27.28</td>
</tr>
<tr>
<td>4</td>
<td>7.90</td>
<td>24.68</td>
</tr>
<tr>
<td>5</td>
<td>6.70</td>
<td>24.65</td>
</tr>
<tr>
<td>6</td>
<td>6.30</td>
<td>24.65</td>
</tr>
<tr>
<td>7</td>
<td>5.40</td>
<td>24.69</td>
</tr>
<tr>
<td>8</td>
<td>4.70</td>
<td>24.67</td>
</tr>
</tbody>
</table>

Send=Receive

**Firefly Stubs+Runtime**

<table>
<thead>
<tr>
<th>Machine</th>
<th>Procedure</th>
<th>Microseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caller</td>
<td>Calling program (loop to report call)</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Calling stub (call &amp; return)</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>Server</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>Transporter (send call pkt)</td>
<td>27</td>
</tr>
<tr>
<td>Server</td>
<td>Receiver (receive call pkt)</td>
<td>158</td>
</tr>
<tr>
<td></td>
<td>Server stub (call &amp; return)</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Null (the server procedure)</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Receiver (send result pkt)</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Transporter (receive result pkt)</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>Enter</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>TOTAL</td>
<td>606</td>
</tr>
</tbody>
</table>

= 11 ms per call to "Null"
Firefly Send+Receive

Table VII: Latency of steps in the send-receive operation

Replace unneeded buffers and process outstanding packets

Assembly Code

Table IX: Execution time for main path of the Ethernet interrupt routine

<table>
<thead>
<tr>
<th>Version</th>
<th>Time in microseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Modula-2+</td>
<td>758</td>
</tr>
<tr>
<td>Final Modula-2+</td>
<td>547</td>
</tr>
<tr>
<td>Assembly language</td>
<td>177</td>
</tr>
</tbody>
</table>

Processors

Table X: Calls to Nul1() with varying numbers of processors

Threads

Table XI: Throughput in megabits/second of MaxResult(b) with varying numbers of processors

<table>
<thead>
<tr>
<th>caller processors</th>
<th>server processors</th>
<th>1 caller thread</th>
<th>2 caller threads</th>
<th>3 caller threads</th>
<th>4 caller threads</th>
<th>5 caller threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>2.0</td>
<td>1.5</td>
<td>1.3</td>
<td>1.3</td>
<td>1.3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3.4</td>
<td>2.3</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4.6</td>
<td>2.7</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4.7</td>
<td>2.7</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4.7</td>
<td>2.7</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Comparison

Table XII: Performance of remote RPC in other systems

Comparison

- RPC, as an abstraction, is popular
  - Both inter- and intra-machine
- Asynchronous versions now common
- Tension between interoperability and performance
  - Java RMI's default implementation is HTTP
  - As is GWT