Improved Parallel Java Cluster Middleware

Xi He
xxh2229@rit.edu

Committee:
Chair: Prof. Alan Kaminsky
Reader: Prof. Hans-Peter Bischof
Observer: Prof. Minseok Kwon
Agenda

- Introduction
- SSH
- Motivation
- Design
- Deploy, user manual and test
Introduction

Why Parallel Computing is important [1]?
Introduction (Cont.)

MPI: a standard application program interface for writing parallel programs.

Important implementation:
Sun MPI [2], MPICH2 [3], OpenMPI [4]
Parallel Java [5], mpiJava [6], MPJ [7]
Introduction (Cont.)

Parallel Java’s Architecture[5]
Introduction (Cont.)

Two goals in this master project

• Improve Job launching Scheme
• Improve Web Interface
SSH protocol layering

- Connection Layer Protocol
- User Authentication Layer Protocol
- Transport Layer Protocol
- TCP
- IP
SSH (Cont.)

Transport Layer Protocol
- Identification exchange
- Algorithm exchange
- Key exchange

User Authentication Layer Protocol
Public key, password, keyboard

Connection Layer Protocol
Channel open, data transfer
Motivation

SSH is too complicated
Replace SSH with MAC

More functionality in Web interface
Parallel Java’s New Architecture
Design Job Launching Scheme

- Modify the protocol for communication between Job Scheduler Daemon and Job Frontend Process

- Newly define the protocol for communication between Job Frontend Process and Job Backend Daemon

- Newly define the protocol for communication between Job Scheduler Daemon and Job Backend Daemon
## Design

### Job Launching Scheme (Cont.)

<table>
<thead>
<tr>
<th>Job frontend process</th>
<th>Job Scheduler Daemon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Job (username, Nn, Np, Nt)</td>
<td>Nn = Number of nodes</td>
</tr>
<tr>
<td></td>
<td>Np = Number of processes</td>
</tr>
<tr>
<td>Cancel Job (errmsg)</td>
<td>If insufficient resources</td>
</tr>
<tr>
<td>Assign Job number (errmsg)</td>
<td>If sufficient resources</td>
</tr>
<tr>
<td>Assign Backend Daemon</td>
<td>Assign required number of backend nodes. If a backend daemon is already running on the backend node, send “Assign Backend Daemon”. Else send “Create Backend Daemon.”</td>
</tr>
<tr>
<td>Create Backend Daemon</td>
<td>Periodically while job is in progress</td>
</tr>
<tr>
<td>Renew lease ()</td>
<td>When job finishes</td>
</tr>
<tr>
<td>Renew lease ()</td>
<td>If user cancels job</td>
</tr>
<tr>
<td>Job finished ()</td>
<td>If error in Job Scheduler, or admin cancels job</td>
</tr>
<tr>
<td>Cancel Job (errmsg)</td>
<td>If frontend detects backend has failed</td>
</tr>
<tr>
<td>Cancel Job (errmsg)</td>
<td></td>
</tr>
<tr>
<td>Backend failed (name)</td>
<td></td>
</tr>
</tbody>
</table>
Design

Job Launching Scheme (Cont.)

Job frontend process VS Job Backend Daemon

Challenge-response MAC based authentication
  • Both sides authentication
  • Data integrity
Design
Job Launching Scheme (Cont.)
Design
Job Launching Scheme (Cont.)

**Job frontend process**

- Challenge (random number A)
- Authenticate (random number B, MAC1)
- Authenticate (MAC3, Job execution parameter)

**Job Backend Daemon**

- Initiate authentication
- One side authentication
- The other side authentication
Design
Job Launching Scheme (Cont.)

<table>
<thead>
<tr>
<th>Job backend Daemon</th>
<th>Job Scheduler Daemon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready (username, ip, port)</td>
<td>When a job backend daemon is created</td>
</tr>
<tr>
<td>Renew Lease (username, ip, port)</td>
<td>Periodically while job is in progress</td>
</tr>
<tr>
<td>Terminate (username, ip, port)</td>
<td>When the job backend Daemon is about to terminate</td>
</tr>
</tbody>
</table>
Design Web Interface

Authentication Issues
Design
Web Interface (Cont.)
Possible Solution
Design
Web Interface (Cont.)
Web Server Design
Deploy

- Package the class files and copy to the cluster parallel computers
- Modify the configuration file
- Set up a certificate
User Manual

• Set up the user’s account

• Demo submit a job via command line

• Demo submit a job via Web interface
Test

• Run 10 Jobs on original Parallel Java, new Parallel Java, Web interface respectively.

• Each jobs require all of the available processors.

• Measure the time for these MPI implementations to launch the submitted jobs.
Test (Cont.)
Reference


Thank you

Any Questions?