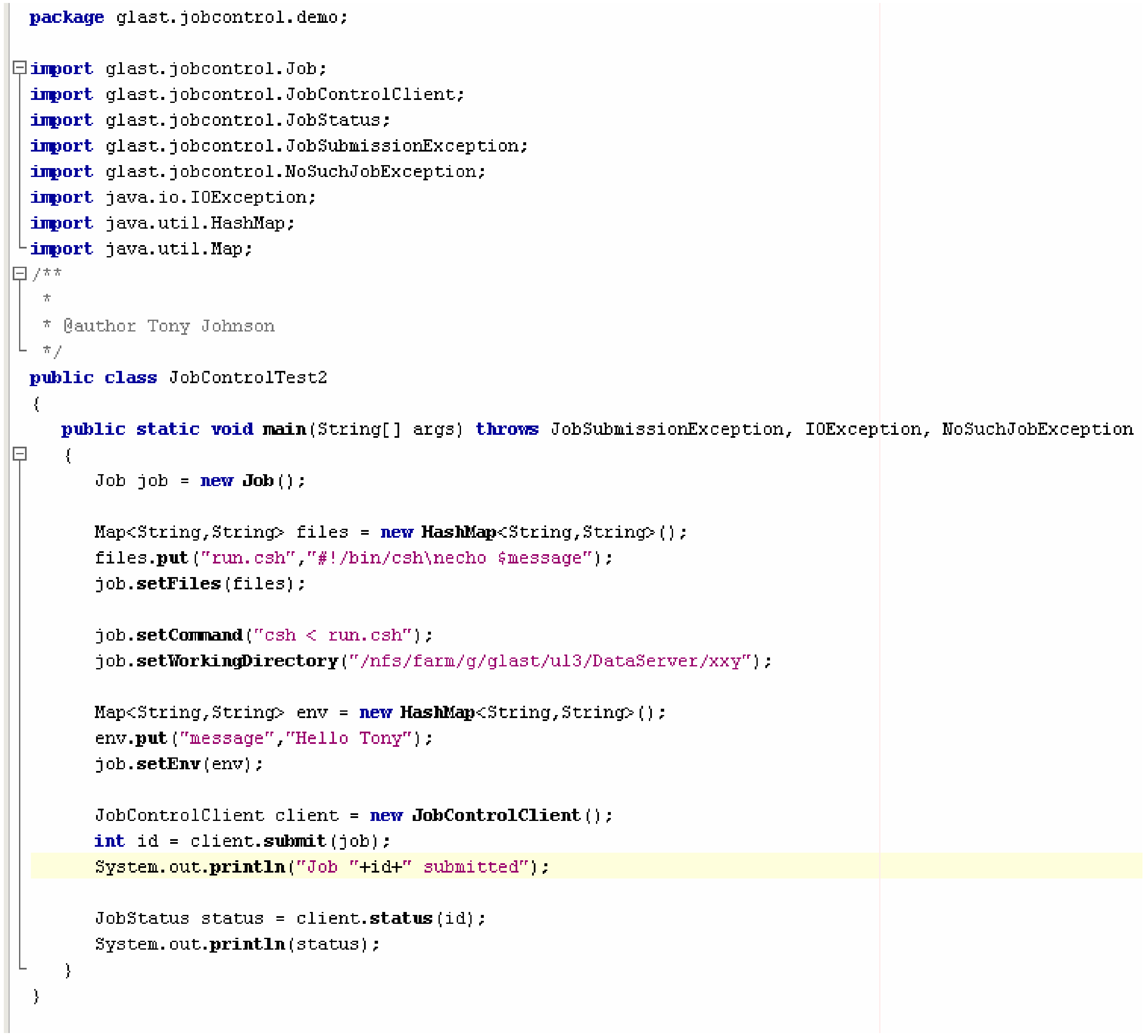
org-glast-bsub

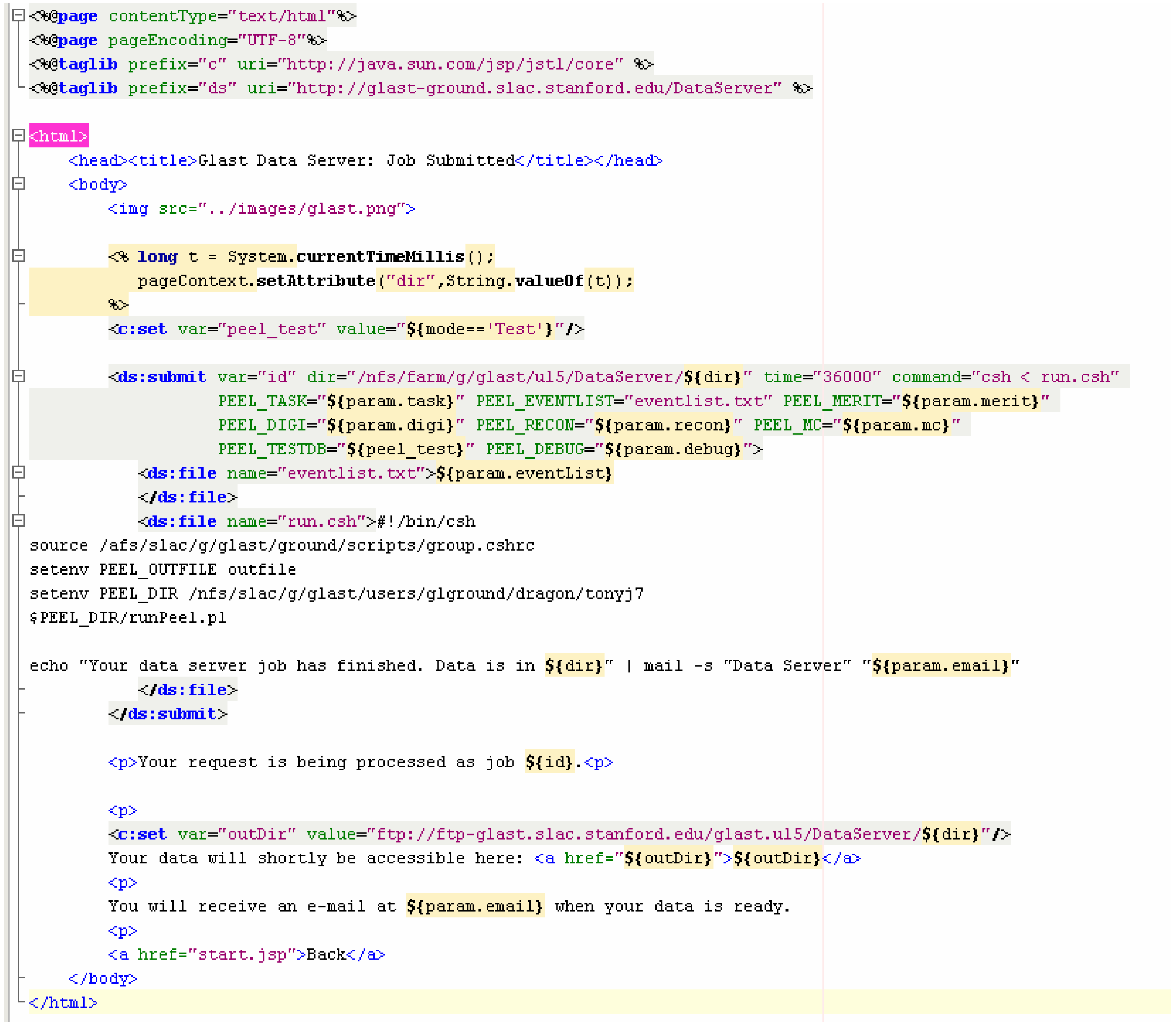
- Batch submission system based on design discussed at Developer's Workshop (in March 2005)
  - Very simple standalone server (started by cron) interfaces to LSF
  - Simple java client can be run anywhere to submit job and query status

Java Usage



```java
package glast.jobcontrol.demo;

import glast.jobcontrol.Job;
import glast.jobcontrol.JobControlClient;
import glast.jobcontrol.JobStatus;
import glast.jobcontrol.JobSubmissionException;
import glast.jobcontrol.NoSuchJobException;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
/**
 *
 * @author Tony Johnson
 */
public class JobControlTest2
{
   public static void main(String[] args) throws JobSubmissionException, IOException, NoSuchJobException
   {
      Job job = new Job();

      Map<String,String> files = new HashMap<String,String>();
      files.put("run.csh","#!/bin/csh\necho $message");
      job.setFiles(files);

      job.setCommand("csh < run.csh");
      job.setWorkingDirectory("/nfs/farm/g/glast/u13/DataServer/xxy");

      Map<String,String> env = new HashMap<String,String>();
      env.put("message","Hello Tony");
      job.setEnv(env);

      JobControlClient client = new JobControlClient();
      int id = client.submit(job);
      System.out.println("Job "+id+" submitted");

      JobStatus status = client.status(id);
      System.out.println(status);
   }
}
```

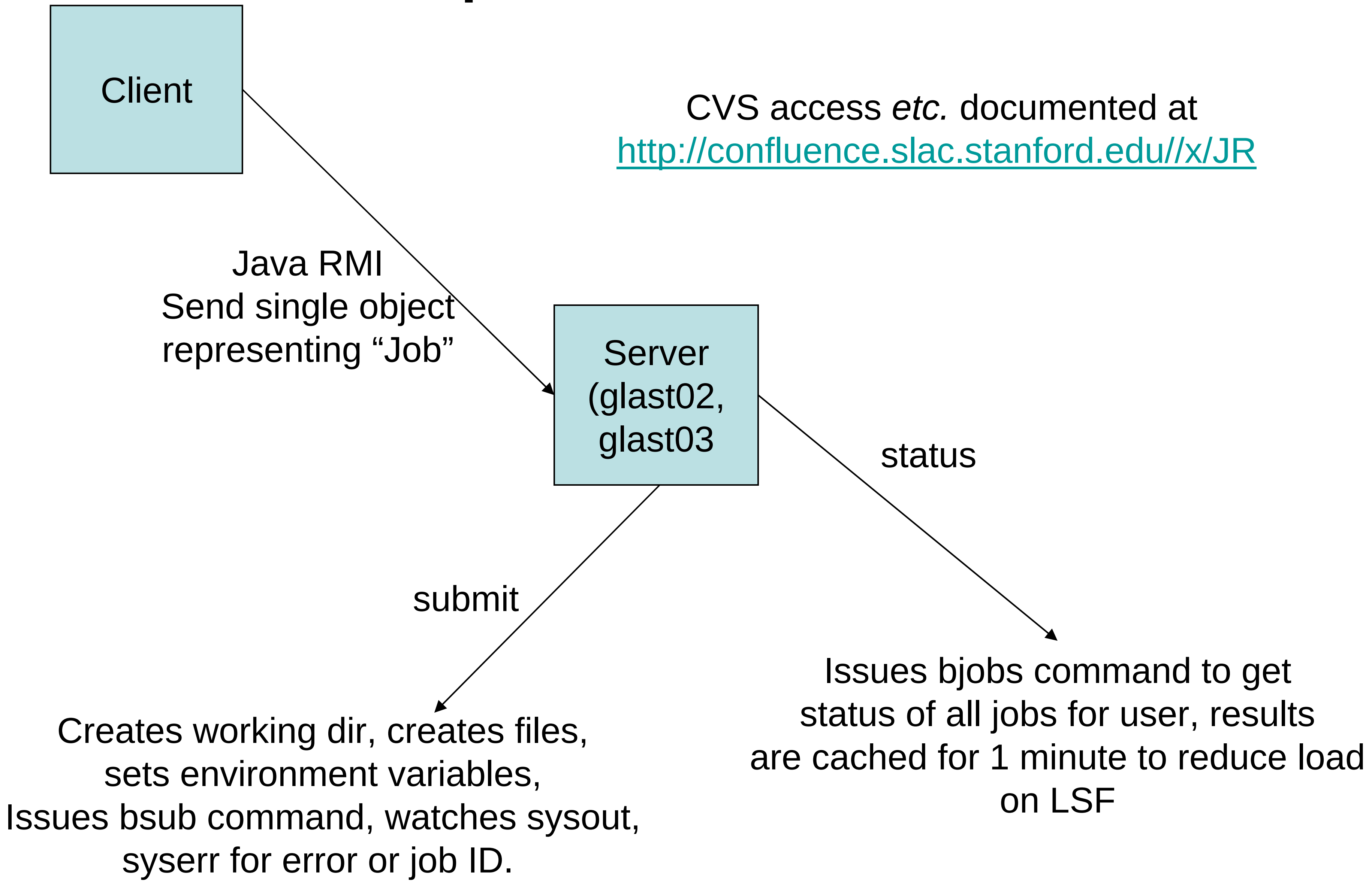
JSP Usage



```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib prefix="ds" uri="http://glast-ground.slac.stanford.edu/DataServer" %>

<html>
    <head><title>Glast Data Server: Job Submitted</title></head>
    <body>
        <img src="../images/glast.png">

        <% long t = System.currentTimeMillis();
           pageContext.setAttribute("dir",String.valueOf(t));
        %>
        <c:set var="peel_test" value="${mode=='Test'}"/>

        <ds:submit var="id" dir="/nfs/farm/g/glast/u15/DataServer/${dir}" time="36000" command="csh < run.csh"
                   PEEL_TASK="${param.task}" PEEL_EVENTLIST="eventlist.txt" PEEL_MERIT="${param.merit}"
                   PEEL_DIGI="${param.digi}" PEEL_RECON="${param.recon}" PEEL_MC="${param.mc}"
                   PEEL_TESTDB="${peel_test}" PEEL_DEBUG="${param.debug}">
           <ds:file name="eventlist.txt">${param.eventList}
           </ds:file>
           <ds:file name="run.csh">#!/bin/csh
source /afs/slac/g/glast/ground/scripts/group.cshrc
setenv PEEL_OUTFILE outfile
setenv PEEL_DIR /nfs/slac/g/glast/users/glground/dragon/tonyj7
$PEEL_DIR/runPeel.pl

echo "Your data server job has finished. Data is in ${dir}" | mail -s "Data Server" "${param.email}"
           </ds:file>
        </ds:submit>

        <p>Your request is being processed as job ${id}.<p>

        <p>
        <c:set var="outDir" value="ftp://ftp-glast.slac.stanford.edu/glast.u15/DataServer/${dir}"/>
        Your data will shortly be accessible here: <a href="${outDir}">${outDir}</a>
        <p>
        You will receive an e-mail at ${param.email} when your data is ready.
        <p>
        <a href="start.jsp">Back</a>
    </body>
</html>
```

Implementation

CVS access *etc.* documented at
http://confluence.slac.stanford.edu//x/JR



Creates working dir, creates files,
sets environment variables,
Issues bsub command, watches sysout,
syserr for error or job ID.

Issues bjobs command to get
status of all jobs for user, results
are cached for 1 minute to reduce load
on LSF

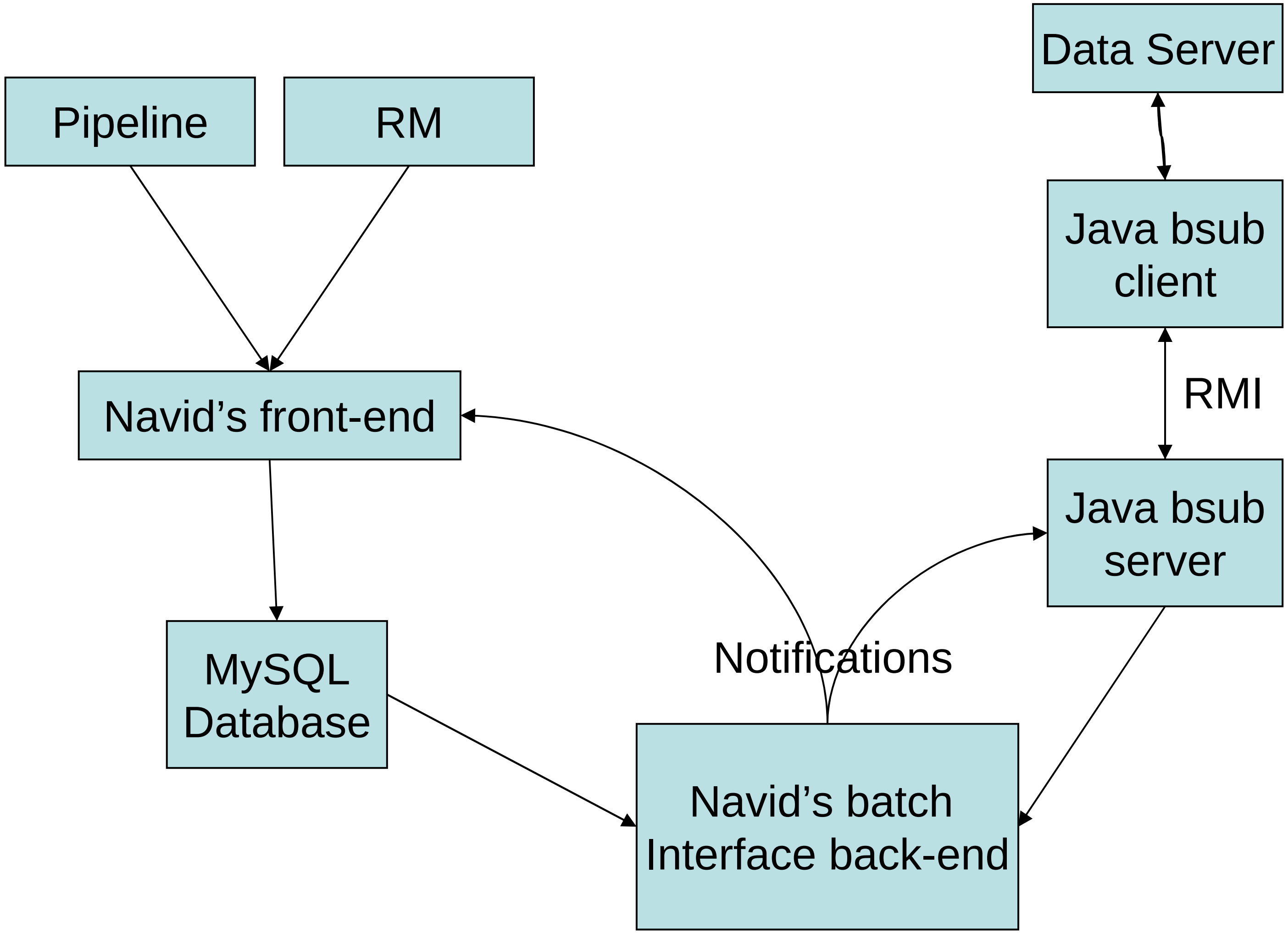
Features

- Allows jobs to be submitted and queried
- Job can include
  - Arbitrary set of files (including main script)
  - Arbitrary arguments (but may be mangled by shell)
  - Arbitrary set of environment variables
  - Working directory
  - Log file location
  - Max CPU
  - Max Memory
  - Priority
  - Arbitrary batch options (use with caution, ties to LSF)
- Status queries are cached to comply with SLAC’s wishes
- Interface is synchronous
  - Submission will either succeed and return LSF id, fail, or timeout.

Missing Features

- No notifications back to client about job status changes (started, ended etc)
- Currently only easily callable from Java
  - (Could add simple web services interface)
- Can only submit under userid Glast
- Client hardwired to use glast02/03
  - (Last two points could be fixed with multiple servers, and Jini lookup to find servers)

Combining Navid + Java batch submission?


Data Server
Pipeline
RM
Java bsub client
RMI
Navid's front-end
Java bsub server
MySQL Database
Notifications
Navid's batch Interface back-end