Team 4
EJBay: A Fault Tolerant Distributed Auctioning System
Final Implementation

Bryan Murawski Joe Trepasso Mike Mishkin
Megan Hyland Jon Gray Prameet Shah

Procedures to start EJBay:

Setup:

Environmental variables -

```
JAVA_HOME="/usr/local/j2sdk1.4.2_02"
JBOSS_HOME="/afs/ece/class/ece749/ejb/jboss-3.2.3"
JBOSS_SERVER="/afs/ece.cmu.edu/usr/jtrapass/749Team4/"
```

Note: Due to the fact that we have changed many of the XML configuration files this setupd will not run on a fresh JBoss install. Also, our JNDI server is run from the team4.jtrapass deploy dir while our main code deploys in team4 so both of these are necessary.

Scripts:

```
runJNDI
runClient <jndi server name>
runServer <afs path to launch server from> <jndi server> <optional: number of replicas>
```

Steps to Run:

1. First, you must start the JNDI server. This can be done as follows

   ```
   ./runJNDI
   ```

2. Next, you must start the replication manager on a server as follows:

   ```
   ./runServer . <jndi server name> <optional: number of replicas>
   ```

   NOTE: If the '.' is omitted, the server will attempt to load from the deploy directory, but as we have not deployed a jar file there this process will fail.

3. To run the EJBay program run the ./runClient command with the machine that the JNDI server is running on as its parameter. Make sure you are in the project folder. Assuming that JNDI is running on othello, the runClient command would take the following form:

   ```
   ./runClient othello.lab.ece.cmu.local
   ```

   Once the program is running, a list of user commands and a user prompt appears on the screen where the client can either login or create an account. Once a user is logged in they can view or edit their user information, they can view or post auctions, and they can view or post bids.

   There is also an automatic client which will run N operations of AccountView on the specified auto-client's account. This was mainly used for testing. To run this alter the experimentList file to contain the number of clients and invocations that you want. Then, manually start the JNDI and server as described above, but do not launch a client. By now you can execute the autoClient.pl script and clients will be created with the proper parameters. (This is not really as automated as would be desireable, however, it's difficult to make this truly automated and still allow you to inject faults using our code and perl scripts)

Comments

- As we noted in our presentation, our final HP application is not exactly passive and a client will fail over to a random server whenever a crash occurs until the replication manager has correctly updated everything. The client will report every time it attempts to talk with a new server, but be aware that there is a brief period where not talking with the primary is expected behavior.
- Log files used for performance evaluation will be written to the `./temp` directory. If this directory does not exist for some reason, the client will print an error message when exiting. This error only means the logs were not written, however, the code ran just fine.
- If you compile (run ant) while a server is running you will cause the servers to die and must restart the replication manager.
- Every node crash causes a server to get banned so when we run out of servers the replication manager will say this and exit.
- A client link failure will not cause failover, however, that client will be able to operate by communicating with one of the backup replicas. This client will, however, have a noticeably performance decrease.
- If the replication manager crashes or you CTRL-C it, server replicas will continue running. To clean these up, you can simply run `./killAllGamesJava`, this script will rsh into all of the games machines except mahjongg and kill java things of yours that appear to be our server.