CSCE 855 PA1 Design Document

Dongpu Jin

2/23/2013

In this programming assignment, an interactive distributed banking system is implemented with client-server architecture using Java RMI library via RPC. The system runs on Amazon EC2 instances. The goal of this banking system is to allow users to manage their bank accounts via an ATM. The ATM serves as the client and the bank runs on the server. The working flow can be viewed as be described as the following: users enter desired operation to the ATM. The ATM sends user requests to the bank server. The bank server checks the validity of the requests and processes valid requests. After finishing the processing, feedback are sent back to the client and displayed to the user.

To be more specific on the implementation details, the core mechanism behind this client-server architecture is RPC, which allows client to invoke the methods on the server via interfaces. The bank management methods declared in the interface are actually defined on the server. The client simply calls the method stubs and the execution transfers to the server to execute the actual method body. In order to make the services provided by the server become available to the public, the server needs to create a registry on a specific port (the default port is 1099). All the clients will be communicating with the server through this registry port. In addition, the server also needs to binds itself to a specific RMI name, such as *rmi://localhost:1099/Bank*, as its identifier. Therefore, clients can look up for a particular server based on its identifier. Note that the server is a "localhost" relative to itself, but when the clients tries to look up this particular server, it needs to provided the full name of the host on which the server runs. For example, the client can obtain a reference to the server by looking up the identifier *rmi://ec2-54-234-217-109.compute-1.amazonaws.com:1099/Bank*. Java RMI library provides many easy-to-use APIs to abstract out the underlying complexity behind this process. When using Java RMI APIs, make sure all the exceptions are well handled. Otherwise, it may cause system crashes when creating registry fails or server lookup fails.

Implemented the banking system in a distributed fashion may not be as fast as implementing everything on a single machine, but the advantage we gain are enormous. First, by separating client and server, it makes the data easy to access. The same bank account can be queried by different ATM client at different locations. If there are multiple servers, much larger amount of data can be stored. This also makes the data more reliable, since data can be duplicated on multiple servers. If one crashes, we will not lose all the data. In addition, it allows multiple ATM to query the server simultaneously. This can be one of the extensions for this system. To achieve this goal, the server may need to create multiple registry ports during launching. Different ATM client can communicate with the server through different port.