Implementation and Performance Tuning of a Scalable Web Service: Design Document

Hamza Khalid
(hkhalid)

Coordinating the roles of the different server instances:

In my design, I have assigned the VM with id 1 as the main VM (or main server). This VM has managerial responsibilities. It keeps track of all the front tiers and middle tiers VMs in a Concurrent Hash Map. Additionally, it has a request-blocking queue for the whole system. The front-tier servers accept connections, parse requests, and put the parsed requests in the request-blocking queue (contained by the main VM) through Java RMI. The middle tiers then fetch these requests from the request-blocking queue and process them.

The front tiers drop the requests if the queue of a specific VM (request assigned to by the cloud to this VM) exceeds a certain threshold (3 in my case). Dropping these requests makes some clients unhappy; however, this approach makes several other clients happy (a trade-off).

In addition, while the initial VMs are booting, the main VM acts as both the front and the middle tier. It then switches its role and acts as only the front tier along with its managerial roles once the next VM started running.

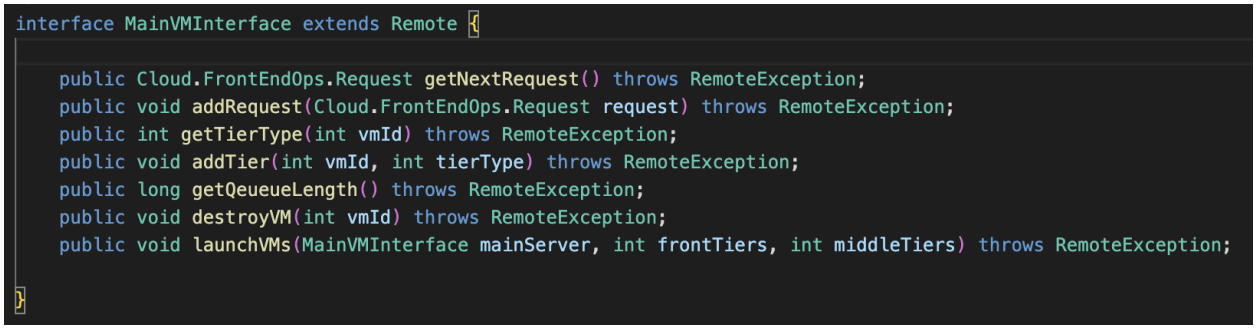
Adding and Removing Servers:

When the program is started, I launched the initial servers based on the initial interarrival rates by noticing when the number of happy clients gets saturated if we keep adding new servers.

Once, the program is running and the load changes, I checked the size of the system's central requests blocking queue. If it is above a certain threshold (7 in my case), I launched a new middle-tier server. In addition, if the request fetched by a middle-tier server is null (times out) three times, I scaled down that particular VM.

Using Java RMI:

Since the middle tier and front tier VMs need to communicate with the main VM to have information about all the other VMs and the request blocking queue, I used Java RMI with the following blueprint:



```
interface MainVMInterface extends Remote {

    public Cloud.FrontEndOps.Request getNextRequest() throws RemoteException;
    public void addRequest(Cloud.FrontEndOps.Request request) throws RemoteException;
    public int getTierType(int vmId) throws RemoteException;
    public void addTier(int vmId, int tierType) throws RemoteException;
    public long getQeueueLength() throws RemoteException;
    public void destroyVM(int vmId) throws RemoteException;
    public void launchVMs(MainVMInterface mainServer, int frontTiers, int middleTiers) throws RemoteException;

}
```