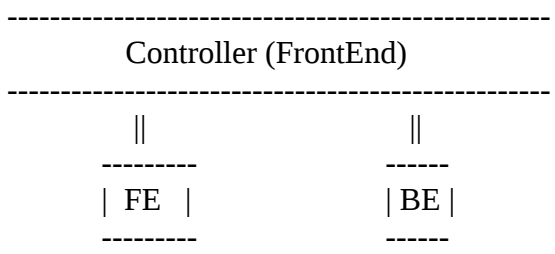
Scalable Service:

Design:



At the start up of the system, there are three Vms in total. One instance is used as the controller, one as a dedicated frontend, one as the backend. During the intial five seconds from the start up, the controller will drop a request until a single backend server is running. Depending on the number of dropped requests during this initial “boot up” period, a number of backends are launched accordingly inorder to service future requests. The number of dropped requests to backends ratio was found out by using trial and error.

Any vm when launched will try to bind itself as the controller with the cloud rmi registry. If there is a controller that is already present, it will register itself as a slave. The controller on launching a new vm will put that vm's id in a seperate list for front ends and backends. The slave vm on succesful booting, will check what role it plays in the system by making an rmi call. The slave's role is determined by the list in which its id is present.

A concurrent queue is maintained in the controller, and each front end (including the controller) will add the requests to the queue as and when they arrive. The backends will fetch the requests from this queue and service them. The communication between the slaves and the controller is carried out through RMI calls.

After the initial bootup time, the controller, in addition to adding requests to the concurrent queue, will also perform the scale out and scale in operations of the backends. It will also perform dropping of the requests in order to prevent the requests from timing out. The scaling and dropping logic are explained below.

Scaling out: When the difference between the global concurrent queue length and the number of 'acitve' backends is greater than zero and the all the backends that were launched are active, the difference number of backends is launched. An active backend is one which registers itself as running with the controller through an rmi call. Seperate lists for the number of total backends and total active backends is maintained. Hence comparing the size of these two lists would suffice.

Scaling in: When a particular backend has not processed a request for a certain time, it unregisters itself with the controller and shuts itself down, automatically.

Dropping: When the ratio of the global queue size at any point of time to the total number of backend vms is greater than a certain threshold, the next request is dropped from the system. The numbers used as threshold were found using trial and error method.

There are only two frontends including the controller and they are not scaled out dynamically.

Cache get: The cache is essentially a concurrent hashmap. In case of a cache miss,during a get operation, the entry is fetched from the main database and stored in the cache

Set / transact operations: These operations are passed on to the main database. The cache is updated accordingly depending on the result of the operation in the main database.