Project 3 Report

Lu Zhang
anderwID: lzhang3

1. Role coordination

There are two kinds of vms in my implementation, one is Forwarder (Front-end vms) and the other is Processor/Application Server (Middle-layer vms). Forwarder is responsible for taking requests from the its requests queue using SL.getNextRequest() and put it into a global request queue. Processor is responsible for poll request from the global queue and process it.

Among forwarders, there exists a unique role known as master, who is responsible for storing the global queue so that other vms can put and pull requests by using RPC calls.

Master also take the responsibility to consider scalein and scaleout, which will be detailed afterwards.

2. Star vm numbers

In this implementation, the numbers of forwarder and processors are decided dynamically according to the interval between first two requests (detected by master). For example, when the interval is larger than 650ms, the start processor number is 1 and forwarder number is 0.

3. Scaling strategy

As said before, master is responsible for considering scaling. In its basic routine, it will check if the length of its request queue is larger than 1.5 times of current number of processors, if yes, scaling out will be triggered. Otherwise, master will detect the interval between two requests, if the interval is much less than previous detected interval, than scaling in will be triggered. **It is notable that, to ensure requests in forwards' local queue will not fail, scaling uses notify-and-kill mechanism**, which is to tell the vm it's time to die, and the notified vm will first finish its stuff and tell master "you can end me". In this way instead of directly killing vms, about 10~20 timeout or failing will be saved.

4. Caching

Caching is on master vm. And caching is done by using a ConcurrentHashMap. When Get is invoded, it first check if the cache has the entry, if yes,

return the value; if not, fetch from default DB, write to cache and return value. Transaction is implemented as writing through the caching, directly interacting with DB, if return value is true, then write in cache.