Name: Ting-Yu Lan
AndrewID: tingyula

15-440 Distributed System
Project 3: Tuning of a Scalable Web Service

Roles of different server instances

Master
- Master is also a front-tier, but it has more things to do than an ordinary from-tier
- Estimated system workload
- Drop requests if a system has too heavy workloads
- Calculate how many new servers needed according to workloads and the number of existing servers
- Scale out servers
- Maintain a cache
- Maintain information shared between different VMs

Front-tier
- Get requests from the load balancer and put them into a queue, called requestList, located on the master

Middle-tier
- Get requests from master's requestList and handle them by using cache
- Calculate the time interval between processing a request. End itself if it has been idle for too long

Cache
- Located on master
- Cache everything in a ConcurrentHashMap. If can not find a key in the map, ask the database for that key
- Pass set and transaction to database, because this is a simple write through cache

How many in each tier to run initially

Initially, I launch 2 front-tiers and 2 middle tiers. When these four VMs start running, I estimated the workloads of the system. My system will launch VM according to the estimated workloads. It can directly launch a maximum number of middle-tier, which is 11 if the workloads are estimated to be really heavy.

When to add or remove servers

I use a queue length to estimate workload. If the estimated workload exceeds a threshold for three sample times, I scale out the middle-tier. I set many different thresholds to determine how many middle-tier do I need to scale out. As to scale in, each middle tier would calculate the interval between processing requests. It would terminate itself if it has been idle for too long.

From the experiment, I found that front-tier do not need to scale out. It would not become the bottleneck in this system. The bottleneck is the database if I launch over 11 middle-tier. Otherwise, the bottleneck is middle-tier.

Name: Ting-Yu Lan
AndrewID: tingyula

Implementation of the database cache
Since it is a write-through cache, it would directly pass sets and transactions to the database. If the middle-tier asks for a key not existing in the cache, it would also pass that request to the database, and cache it. The cache would cache everything since it has unlimited memory space.

Other Design
- I put the cache on master instead of putting it on a VM alone. This can not only save VM time but also save RMI connecting time. The middle-tier only needs to connect to a server, master.
- The master would directly drop requests get from the load balancer if there are lots of requests in requestList. If the master does not drop them, they will timeout in the requestList and hurt the system's performance.

What I have learned
- Use the constant time to test different workloads first before testing on mix distribution.
- It is better to tune only one parameter once and fix all the others. This allows me to clearly understand how a parameter affects the system.