Design & Implementation

15-440 Spring 2018 Project 3
Yanning Mao (yanningm)

Role of Servers

My scaling system contains one master server, and multiple slave servers.

The master server works as a front-tier server, and always exists. The responsibilities of the master server contains:

- assign roles to the slave servers
- scales out
- maintains two central queues of requests polled by the front-tier servers (one for browse requests and one fro purchase requests)
- serve as a cache server

The slave servers can work as either a front-tier server, or a middle-tier server. The responsibilities of a slave server contains:

- ask the master server for its own role and work as that role
- front-tier slave servers polls requests from the Cloud and sends them to the central queue on the master server
- middle-tier slave servers polls requests from the central queue and sends them to the Cloud for processing
- determines whether itself should terminate and asks the master for approval of self termination

Scaling Policies

Initial Number of Servers

The initial number of servers to start with is determined via benchmarking. In CheckPoint 1, we've discovered the optimal number of single-tier servers for each hour.

Through benchmarking, the optimal number of front-tier and middle-tier servers to start with is

- initFrontTierServers = Math.max(numServer / 3, MIN_FRONT_TIER_SERVERS)
- initMiddleTierServers = 2 * Math.max(numServer / 3, MIN_MID_TIER_SERVERS)

where numServer is the optimal number of servers to start with from CheckPoint 1

Scale Out Policy

Scaling out is maintained by the master server. The master server checks for scale out every 1.5 seconds.

The benchmark result shows that scale out is needed when

    number of queued requests > 1.5 * total number of servers

The number of new middle-tier servers is

    number of queued requests - 1.5 * total number of servers

The number of new front-tier servers is

    number of new middle-tier servers / 4

Scale In Policy

Scaling in is maintained by each slave server itself. The slave server checks if itself needs to terminate based on its request arrival rate, and asks the master for approval of termination.

The slave server asks for termination when any of the following case is true:

1. The period since last request arrival is too long (greater than 1 second for front tier & greater than 2 seconds for middle tier)
2. There have been continuous slow request arrivals (two continuous inter-arrivals greater than 0.5 seconds for front tier & two continuous inter-arrivals greater than 0.7 seconds for middle tier)

The master refuses for termination when any of the following is true:

1. The termination of the slave server results in too few servers
2. The terminations of the slave servers are too frequent
3. The termination request is from the master server itself

Drop Policy

Dropping the requests is checked during each scale out.

Each request in the central queues on the master is equipped with a time stamp, which records the time it was added to the queue. The master examines from the head of each of the two central queues, and drops all the preceding requests that have waited too long in the queue (0.8 seconds for browse requests & 1.8 seconds for purchase requests).