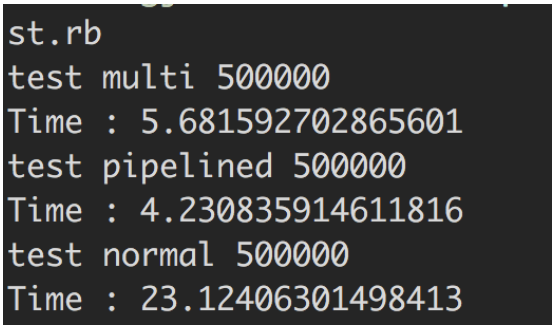
Nano Twitter 0.6 Update

1. Experiment with Redis

   Before implementing caching layer with Redis, I did some small experiments.

   There are two things that I think is important. The first is the connection time and the second is atomicity. Multi method can deal with atomicity, and pipeline is the way to send several commands to the Redis server as a batch. The test result is as following.



2. What have we done with Redis currently

   We have cached the following and follower list of each user into Redis.

   The current business logic for home time line query is to get the following users first, and then query all the tweets with this following list. And now we can get following list from cache, so if say that we have n tweets altogether, the time complexity for this is O(n).

3. What we want to do with Redis in the future

   We also want to cache the timeline and home timeline in cache, but we only want to store tweet ids in cache. So we can imagine that there are altogether k tweets in one user' s home line, the time complexity is O(klogn). So, if have a lot of tweets in the database and k < logn, it is better to do this in this way.

4. The current architecture of Redis.

   When we need to query follow relationships, the application sever don't interact with the database directly. It only get data from the caching layer. If the cache doesn't hit, it will import data from the database automatically and then return the data back to application server.

5. Some problems because of multi-thread and multi-process.

   There will be slight possibility that the caching layer will do redundant data import and sometimes the return result may be empty due to data expiration. But it will not affect the following requests. To completely fix this, a queue is necessary to keep operations synchronous.

   Comments in "lib/datacache.rb" explains this more detially.