# HW2 Twitter Data With Redis

Wesley Jiang

On average with broadcasting the insertion speed is 4.418 milliseconds per insert. Without broadcasting insertion time is 0.079851 milliseconds per insert. The speed is significantly faster without broadcasting. Inserting a million tweets would have taken a whopping 1.22 hour inserting time whereas inserting without broadcasting took only 78 seconds. The difference is 55 times faster inserts without broadcasting.

On average, it took 38066 milliseconds for retrieving a user's timeline without broadcasting and 52 milliseconds for retrieving a user's timeline with broadcasting. Retrieving all users inserts would have taken 106 hours without broadcasting, whereas a loading time of 8.6 minutes with broadcasting. It is 732 times faster to retrieve a prebuilt timeline than to build one on the fly.

The two strategy have trade-offs. By performing broadcasting, the insertion time is 55 times slower than inserting without updating any timelines. However, with pre-updated timelines, the retrieval of a timeline is 732 times faster than building one on the fly. The broadcasting strategy saves much more time. More importantly users care much more about viewing their timeline than the time it takes to post a tweet, which is why it makes sense for twitter to use this strategy.

The timeline request for MySQL was 5 requests per second. With broadcasting, Redis was about 20 requests per second, however without broadcasting, it would take up to  38 second per request.
The insertion speed for Redis without broadcasting is 12k inserts per second, and with broadcasting is 226 inserts per second. MySQL insertion speed was 7k inserts per second. Overall MySQL is much slower than Redis at requesting timelines with the implementation of broadcasting. Redis only has a faster insertion speed if broadcasting is not implemented. However, the pervious MySQL homework was done in Python instead of Java. The two languages might also affect performance of the applications.  The exact implementations might also delayed the performance of Redis.

#Class/DS4300