Project 3 Report
Sanjay Nair
Preethu Thomas

After implementing and testing a version of the Chord distributed hash table in Scala using the Akka Actor framework, we were interested in testing if our implementation matched the logarithmic performance present in the full implementation. When testing our program, we were interested in increasing the number of nodes in the system linearly or exponentially and see if the number of hops required to lookup a value maintained a log(n) relationship with n nodes in the system.

We began with a simple test, increasing the number of nodes from 10 to 1000 by steps of 10 and having each node complete when they sent 10 messages (for the interest of time). Of course, the average number of hops computed at the end of execution could be considered to be more accurate if a larger number of messages were sent by each node. However, with each node sending one message a second and the time required to build the Chord ring, we decided that 10 messages was sufficient to observe the trend of increasing number of hops with increasing number of nodes. Figure 1 shows the results of this experiment.

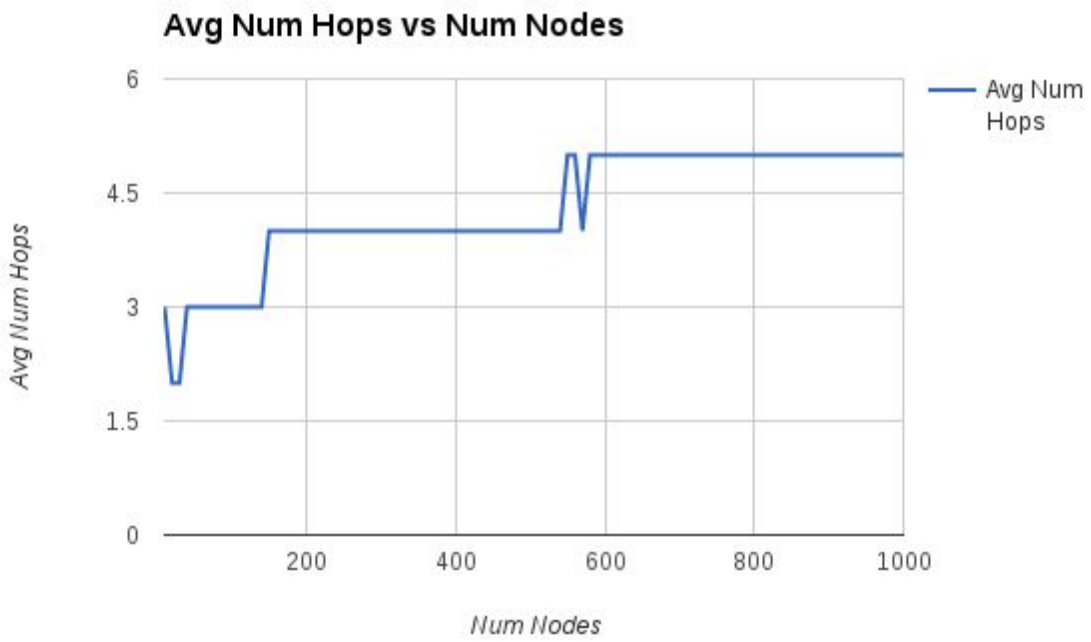

Figure 1

As we hoped, the number of hops required to lookup a value seemed to be increasing logarithmically with respect to the number of nodes. We continued the experiment by running the system with 10 to 262144(2^18) nodes, 10-1000 using an increment of 10, 1010-9010 using an increment of 1000, and between 32-262144 using increments of powers of 2. On each run to monitor performance in a more quickly growing system. The results, including a fit with a logarithmic trend line is shown in Figure 2.

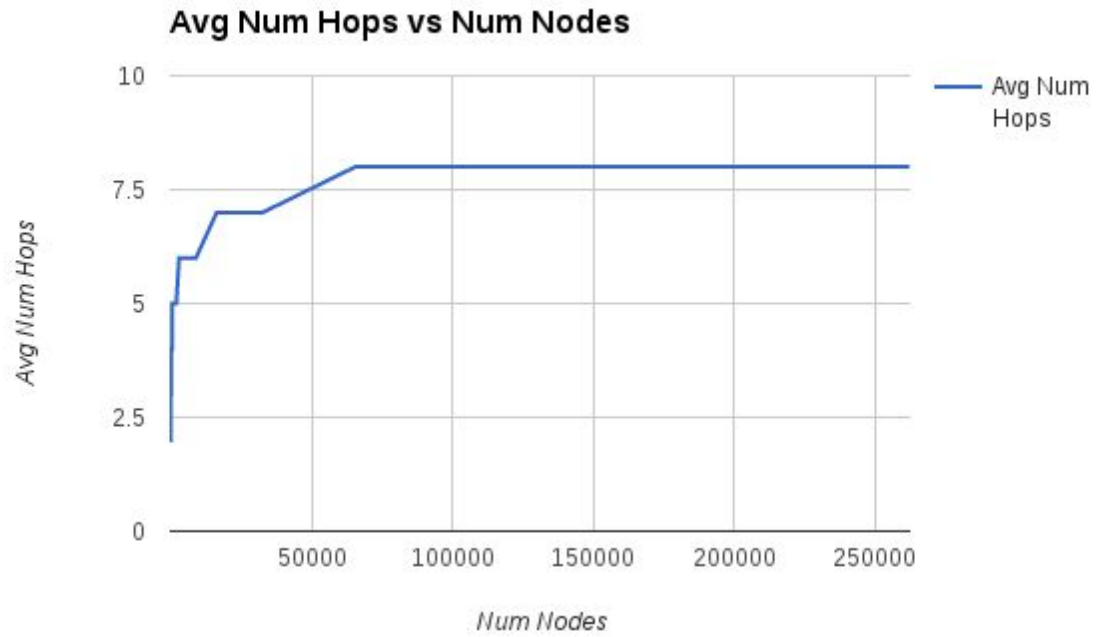

Figure 2

We can conclude that our system does meet the logarithmic performance that follows the behavior presented in the Chord publication. The main pitfall of the method we implemented was the amount of time required to build a Chord ring of increasing number of nodes before starting the requesting process from each node. Our implementation currently inserts one node into the system every 100 milliseconds to allow for each node to have enough time to update all other relevant nodes when joining. Due to the asynchronous nature of the Akka framework, this was necessary to provide some assurance that each node join would not cause any inconsistency in the Chord ring as a whole. The side effect was that setting up a ring of 100,000 nodes took approximately 2.7 hours, and 262144 nodes took approximately 7.2 hours before requests could even begin to propagate through the system.