Optimized Distributed Hashing Table with Improved Availability and Better Balanced Load

Junghoon Kang Chaoren Liu

April 4, 2016

1 Problem Description

The distributed hashing table (DHT) is a key-value store where key-value pairs are spread among Internet-connected nodes. Each node can retrieve the value given a key from this network. DHT is a convenient data structure, and interesting applications can be built on top of it.

Our project has two phases:

- Implement and test the basic of a DHT. The redistribution of key-value pairs in case of node join-in and node leave (failure and delay) will be considered.
- On top of the basic DHT above, we will add features like high availability (by key-value pairs replication). When one node fails, the key-value pairs on the failing nodes will be redistributed to the successor node, which congests the communication between the successor and other nodes. On the other hand, the successor may be overloaded. We will solve this problem by mapping a node to more than one images on the circle and redistributing key-value pairs on a failing node to more than one nodes. Similarly, when a new node join in, the new node also pull key-value pairs from several nodes (the image number of a node depends on its capability).

2 Related Work

This project requires a deep understanding on DHT and an efficient look-up algorithm. We will read relevant literatures on these topics before starting the implementation. To complete the add-on features in phase 2, we need to learn from available distributed key-value stores, such as Dynamo from Amazon.

3 Plan

- April 2 - April 10: Finish the phase 1.
- April 11 - April 17: Finish the phase 2.
- April 18 - April 20: Finish the writing and the presentation.

4 Resource

We will work on our laptops. Software we will use is Scala/Akka. Scala/Akka is a convenient tool for distributed programming.