Distributed Computing Systems (CSCI 4780/6780) – Project 4

Consistent Hashing-based Naming Service

Introduction

In this project, you are going to implement a consistent hashing (CH)-based flat naming system. As we studied in the class, consistent hashing provides a lookup service for key-value pairs. The CH system stores (key, value) pairs on a distributed set of servers. These servers collaboratively provide lookup service along with insertion and deletion. In this project, you will implement the CH name servers and a boot-strap CH server, which is a special CH server with certain extra functionalities. In order to simplify the implementation, several assumptions are made as follows.

1. The keys are assumed to be numbers in the range [0, 1023] (both inclusive). This mitigates the need for an extra hash-function. You can use the key value itself as the hash-value.
2. Values are assumed to be alphanumeric strings (no special characters).
3. The CH name server IDs are also numbers in [0, 1023] range.
4. The Bootstrap name server has the ID 0. This will be a permanent node in the system. In other words, the system starts up by having only this bootstrap name server in the system (i.e., starting up this Bootstrap name server starts up the system). The Boot strap name server does not exit the system (when it closes down the DHT system is assumed to have closed down).
5. The lookup, insertion and deletion operations are executed only at the Bootstrap name server. It provides a thread to interact with users and accepts lookup, insertion and deletion commands (see below for more details).
6. The name servers (including the bootstrap name server) interact with one another using sockets primitive.
7. Please note that you are implementing consistent hashing, which means that each name server only knows about it predecessor and successor name servers.
8. It is assumed that the bootstrap name server is known to all other name servers (i.e., it is provided at startup).
9. The entry and exit processes of name servers always start at the bootstrap nameserver.
10. Entry, lookup, insertion and deletion traverse the servers in clockwise direction (i.e., by following successors).
11. You can assume that there will not be any race conditions in the system (i.e., simultaneous lookups, insertions, deletions, entries and exits).

Bootstrap Name Server

As mentioned above the bootstrap name server is a permanent name server. Its startup signals the creation of the CH system and its close-down brings the entire system down. It is assumed that the Bootstrap name server is the first node in the system. In other words, at startup it is responsible for the entire [0, 1023] range.

The Bootstrap name server (bnserver) takes a single command line parameter – the name of the bootstrap name server configuration file (bnConfigFile). The bnConfigFile has the following format:

`ID of the server` (which is 0, as it is the Bootstrap name server)
`Port number on which the Bootstrap server will wait for connections from other servers.`
`A series of initial key value pairs.` One pair per line (key and value delimited by blank space).

An example `bnConfigFile` is available here (http://cobweb.cs.uga.edu/~laks/DCS-2021-Sp/pp4/bnconfig_example.txt)

Upon startup the Bootstrap name server will manage the entire [0, 1023] range until other servers join the system. The Bootstrap name server will also provide a user interaction thread. The user interaction thread will support the following commands.

`lookup key` → retrieves the value corresponding to the given key (if the key is in the system). If the given key is not in the system, “Key not found” should be printed. In addition to the value, this commands should also printout the sequence of server IDs that were contacted and the ID of the server from which the final response was obtained.

`Insert key value` → should insert the key value pair into the system. The command should print out the ID of the server into which the key value pair was inserted and the sequence of server IDS that were contacted.

`delete key` → should delete the key value pair corresponding to the given key. If successfully deleted, “Successful deletion” should be printed. If key was not in the system “Key not found” should be printed. In addition, the sequence of server IDs that were contacted in the deletion process should be printed.

Name Server

These are non-permanent servers in the system. Each name server has an ID (specified in the configuration file). Each name server is responsible for maintaining the key, value pairs for a specific range of keys as specified in the consistent hashing algorithm. Each name server knows its predecessor and successor name servers (i.e., it maintains the IP address and port number of its predecessor and successor name servers).

The Name server (nameserver) takes a single command line parameter – the name of the name server configuration file (nsConfigFile). The nsConfigFile has the following format:

`ID of this name server` (number in [1, 1023] range)
`Port number on which this server will wait for connections from other servers.`
`IP address and port number of the Bootstrap server (delimited by space).`

An example `nsConfigFile` is available here (http://cobweb.cs.uga.edu/~laks/DCS-2021-Sp/pp4/nsconfig_example.txt)

Each name server will also provide a user interaction thread. The user interaction thread will support the following commands.

`enter` → the name server will enter into the system. This process is initiated by first contacting the Bootstrap server. It will figure out the range of the keys that it should maintain by following the consistent hashing protocol (i.e., traversing the existing name servers in clock-wise direction). It will acquire the key value pairs corresponding to its range from the name server that will become its successor name server. Once the entire entry procedure is complete, "successful entry" message is printed out. In addition, the following information is printed out: (1) The range of keys that will be managed by this server; (2) the ID of the predecessor and successor name servers; (3) the IDs of the servers that were traversed during the entry process.

`exit` → the name server will gracefully exit the system. The name server will inform its successor and predecessor name servers. It will hand over the key value pairs that it was maintaining to the successor. Upon successful exit, the server will print "Successful exit" message. It will also print out the ID of the successor and the key range that was handed over.

Points to Note:

1. Several design decisions (such as messaging formats between name servers and data structures to be used in servers etc.) have been deliberately omitted. Coming up with a good design is a very important part of the project.
2. You are allowed to make reasonable assumptions (if need be). However, those additional assumptions should not be contrary to the original CH protocol (for example, you should NOT assume that every name server knows about every other name server in the system). It is a good idea to talk to the instructor before making additional assumptions. The assumptions should also be mentioned in the readme file.
3. Evaluation will be done through demos.

What to submit:

1. Your code for the entire system.
2. A readme file containing (a) names of the students in the project group; (b) any additional assumptions, special compilation or execution instructions; and (c) the following statement – if it is true – "`This project was done in its`

`entirety by <Project group member names>. We hereby state that we have not received unauthorized help of any form`". If this statement is not true, you should talk to the instructor before submission.

3. All submission will be done via ELC.