Report 5: Chordy

Peter Hamberg

October 7, 2011

1 Introduction

The epic tale of the construction of a distributed hashbased storage system.

When the storage requirements for a task becomes too big for one machine to handle, we have to branch outwards to find a solution. We create a ring network of nodes, where we balance the storage load between the many nodes in the network by wishing really hard that the fictitious hashes of the data we wish to store will map somewhat evenly over the distribution of IDs given to the nodes.

2 Main problems and solutions

The Chordy System, aptly named after the Chord Protocol, uses message passing to implement the details of the protocol. The data store is handled using the add, lookup and handover messages. Ring topology is maintained using the notify, request and status messages. Testing and ring integrity verification is provided by the probe messages. There is also the key and stabilize messages, used to set up nodes and initiate regular tests of ring topology respectively.

The data store messages are rather simple. The store is just a key-value list, and the add and lookup messages do, rather unsurprisingly, adding and retrieveing of messages - or if the key belongs to a different store; pass the message further down the ring. The handover, done when a new node injects itself between two already connected nodes, is handled by splitting the store in two parts using the lists:partition/2 function, sending to the new node the data it is now responsible for handling. The new node will simply merge the incoming data with its existing store.

The ring topology is a simple doubly linked list, with nodes ordered by id. The details of this algorithm have been both discussed and re-enacted during a previous lecture, and it will not be described in depth here. Suffice to say that the function names in the code became so confusing (i.e. "wait, is this the request I'm currently sending or the one I'm supposed to receive?", "The stabilize message doesn't trigger a call to the stabilize function? It

does?" and "Why, oh why is every function called notify all of a sudden?") that I applied some minor function renaming operations. Also I was bored. And tired.

The probe functions simply pings the successor with a timestamp and waits for the signal to be passed one lap around the ring. While this can appear to be simple, it triggered minor confusion on my part when the output seemed to list the nodes not-at-all in order. After a few seconds of cold sweat, and a gnawing sensation of fear that some elusive bug had escaped my automated tests, I realized that the ring was perfectly ordered, but that the start of the list didn't neccesarily have to be the smallest number. Such life.

The stabilize function is simply triggered once every few often. This is rather boring, and I mused for a while if we couldn't pass some sort of trigger signal instead. The programming-should-be-beautiful nerd within me simply doesn't like the polling side of the poll/notify-debate, but alas, time was short.

3 Conclusions

A storage system implemented over several nodes, very exciting. Unfortunately I could not test it over multiple computers at home, it will have to wait until the seminar.

One thing I'd like to examine further is what happens if selected messages are dropped instead of received. What can be done if a handover is lost, or what would be the result of a confused node continuously passing on its own lookup message?