UiT

THE ARCTIC UNIVERSITY OF NORWAY

INF-3200 Assignment 2

Department of Computer Science

Peter Munch-Ellingsen

November 9, 2015

This report contains 5 pages including this cover page

1 Introduction

This report outlines the design and implementation of an expansion to the previously discussed hash-map. This version includes the ability to add and remove nodes as well as systems for electing a leader of the network.

2 Technical background

2.1 Distributed hash-map implementation

As mentioned above this is an extension to a previously implemented distributed hash-map. The map is structured in such a way that each node knows each other node in the map. This kept costs of adding and retrieving data from the map relatively low. As discussed in the earlier report this system is not designed for being dynamic. A dynamic system in this sense is a system which can have nodes leave and join on runtime. A couple of ideas were however outlined in the report and for this report the system will be expanded to include joining and leaving.

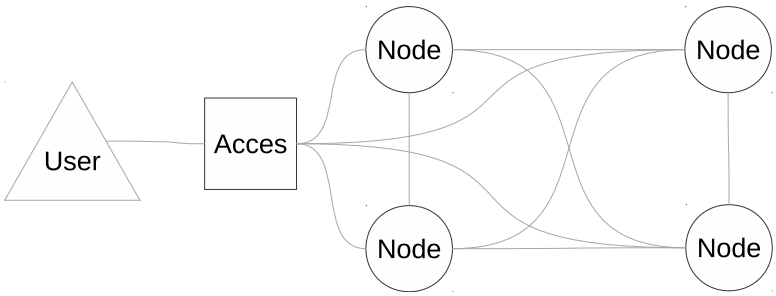

Figure 1: Sample system structure

3 Design and implementation

As noted the previous implementation is not designed for being dynamic. As each node keeps a list of all the other nodes each node in the network will have to update their lists when a new node comes or goes. In addition each node must ensure that the list they keep is the exact same list as the other nodes, otherwise there will be confusion in regards to the hashing algorithm.

To recap, the previous system kept a numbered list of all nodes. Then when a value entered the system it would hash it's key and modulate it over the number of nodes to arrive at the node which should have that value. This hashing system would not work for a dynamic system. During testing it was shown that as many as 99.8% of the values in the map would need to change node when a node was added or removed. Seeing as the bottleneck in a distributed system typically is communication this would be unacceptable. Therefore a new hashing algorithm has been chosen. The way it works is by generating the same hash as before but instead of modulating the value it uses bit-mask to map it into a smaller field of values. If is still doesn't fit the value is fed back into the system and re-hashed. What this means is that as long as the number of nodes does not cross a bit-mask boundary (e.g. 8, 16, 32, 64) then the amount of values which has to re-locate is equal to the amount of values that the new node should have. This is a tremendous benefit as the communications are now limited to only the joining node and not causing disturbance in the network as a whole. In addition it is also possible by using the algorithm to know which values will have to relocate. This means that only the few values which needs relocation would actually have to be hashed again. This is however not currently implemented.

In order to ensure that the network stays sane when adding and removing nodes it is paramount that a scheme is developed such that no node may see a different list than the other. The current system works by polling a random node for the list of nodes in the network. Since all nodes would have to be informed of the joining or leaving node the approach taken is simply to start at the top of the list and informing downwards. When combining this with communicating to the nodes that should be informed of what the network is expected to look like after the join it is possible to ensure that the network stays sane. Since all leaving and joining nodes starts at the same node it is easy to see how only legal states should be possible. It is however worth noting that the network might be unable to properly account for all values in the network as they are passed around. This is however also the case for the previous implementation. And as each node makes sure that a key belongs with it when adding information the network should never actually lose the data.

When a node leaves the network it would typically shift all the nodes. This would mean that all nodes which comes after the leaving node would need to retransmit all their data. This is circumvented by a clever trick. Whenever a node wants to leave the network it switches places with the

last node in the list. This means that only those two nodes would need to exchange data apart from the actual repositioning of the data in the last node.

The task also has a goal of being able to determine a leader in the network. This is simply predetermined to be the node at the top of the list. By using this system the cost of appointing a leader is zero, and the leader position is also automatically filled and communicated to all nodes automatically by the joining and leaving algorithm.

4 Discussion

The network discussed here is quite efficient for smaller networks. It should however be quite trivial to put multiple such network structures together in a tree-like structure. This would make sure that the regular costs of joining and leaving would be kept fairly low while still supporting a large amount of nodes and values. Image for example creating a simple three-layer 10 node network, it would lead to good speeds as the parts which sees each other are quite small while still supporting 1000 nodes.

The drawback of crossing the bit-mask boundary is quite substantial and leads to about half of the total data to need relocation. This could however be mitigated by keeping virtual nodes running on the same hard-ware. For instance if a node would leave the network and cause a bit-mask boundary change then another node could spawn a virtual node on the same machine which would keep the number of nodes above the boundary limit while still letting the initial node leave. Such virtual nodes could also be shed when new nodes would join the network if they cause a boundary crossing in the other direction. By implementing such a system the performance would be made quite stable as far as the initial size bracket had been chosen well.

The current implementation attached with this report only incorporates the basic functionality of the possible design. It also has some trouble dealing with redistributing values when nodes leaves. As this was not the focus of this assignment it was however left unresolved.

Another, more serious, flaw is the lack of HTTP support. The initial design had a translator server which translated from HTTP to the internal format. As the specification was not sufficiently studied before implementation this was kept in the design to help remove some of the unnecessary bloat of the HTTP protocol. This means that the network still operates

with a single entry-point which then randomly forwards the commands to the underlying network and thus making it incompatible with the test suite intended for validation.

5 Conclusion

The network demonstrated here fulfills the purpose of a dynamic network and incorporates simple mechanics by which to elect a leader. It does also incorporate features not outlined in the specification and unfortunately misses some points due to the legacy code base. The design is however overall fairly solid.