University of Tromsø

Assignment 2

Distributed Key-Value Store

Authors:
Nilsen Andreas and Einshøj Alexander

November 5, 2017

Contents

Introduction

This paper describes how to implement distributed key-value store operations upon the peer-to-peer protocol chord. The motivation with a distributed key-value store compared to a single machine is scalability and fault-tolerance and performance.

1 Design

Nodes are assigned a key space that range from it's own id to its successor's id. Whenever a new key-value pair is inserted into to the system, the key k is hashed with *SHA1*. The node that owns the key space where k lies can be fetched by using the method $findsucessor$[1, p.6] with k.

There are several ways to store a key-value pair for a node, but a trivial data structure to use is a dictionary.

2 Discussion

The figure below shows the throughput per second of PUT and GET requests, running on a different number of nodes, ranging from 1 to 16.

The complexity of the current linear lookup is O(n). This can be seen from the graph, which is slightly skewed by the overhead of the application. By implementing fingertables, a logarithmic lookup of O(log n) would be achieved, but this is not currently implemented.

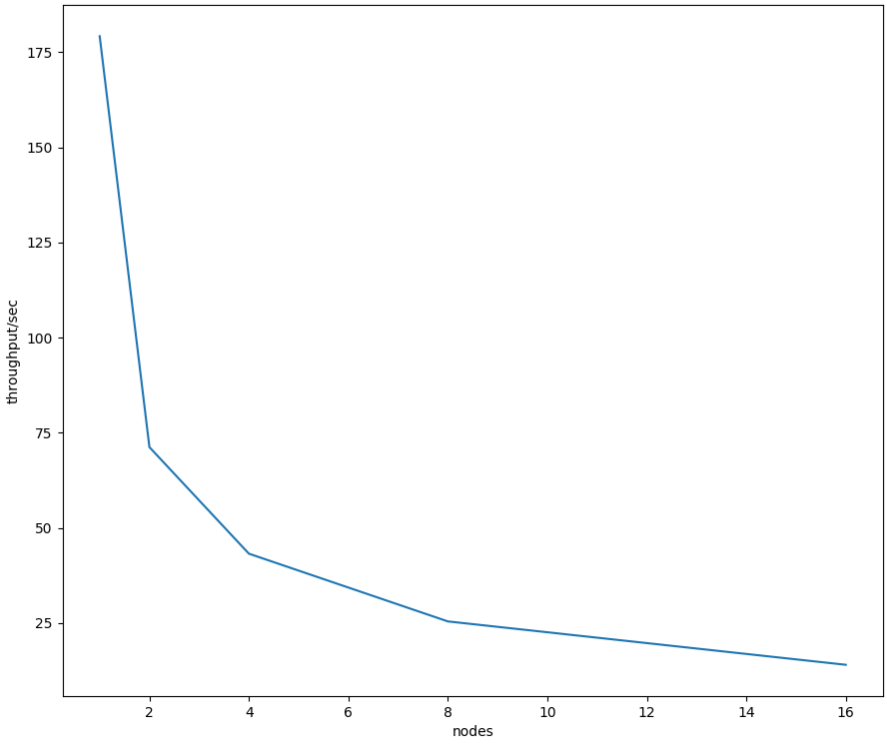

Figure 1: graph

3 Implementation

The chord protocol and key-value store operations are implemented in the programming language GO, throughput-testing and graphing of the results were done in python.

References

[1] D. L.-N. Ion Stoica Robert Morris, *Chord: A scalable peer-to-peer lookup protocol for internet applications.*