Homework Assignment 4

Introduction

Structured DHTs

In structured distributed hash tables, such as Chord, Pastry, Tapestry and others, O(log(N)) lookup performance can be achieved. For latency-sensitive applications in distributed environments, this logarithmic performance translates into high latencies.

Proactive replication in DHTs can be used to achieve constant lookup performance O(1), in case that the query distributions of popular applications are known upfront. When a power law distribution can be assumed, the pro-active replication strategy can be modeled into a closed-form solution.

Replication framework

The proposed general replication framework operates on top of any DHT that uses prefix-routing, such as Chord, Pastry, Tapestry and Kademlia. In such DHTs, each node has a unique randomly assigned identifier in a circular identifier space. Each object also has a unique randomly selected identifier, and is stored at the node whose identifier is closest to its own, called the home node. Routing is performed by successively matching a prefix of the object identifier against node identifiers. Generally, each step in routing takes the query to a node that has one more matching prefix than the previous node. A query traveling k hops reaches a node that has k matching prefixes. Since the search space is reduced exponentially, this query routing approach provides $O(\log_b(N))$ lookup performance on average, where N is the number of nodes in the DHT and b is the base, or fanout, used in the system.

In the proposed framework, the length of the average query path will be reduced by one hop when an object is proactively replicated at all nodes logically preceding that node on all query paths. For example, replicating the object at all nodes one hop prior to the home-node decreases the lookup latency by one hop. We can apply this iteratively to disseminate objects widely throughout the system. Replicating an object at all nodes k hops or less from the home node will reduce the lookup latency by k hops.

The extent of replication in the system is controlled by assigning a replication level to each object. An object at level i is replicated on all nodes that have at least i matching prefixes with the object. Queries to objects replicated at level i incur a lookup latency of at most i hops. Objects stored only at their home nodes are at level $\log_b(N)$, while objects replicated at level 0 are cached at all the nodes in the system. Figure 1 illustrates the concept of replication levels.

The goal of the replication strategy is to find the minimal replication level for each object such that the average lookup performance for the system is a constant C number of hops. Naturally, the optimal strategy involves replicating more popular objects at lower levels (on more nodes) and less popular objects at higher levels. By judiciously choosing the replication level for each object, we can achieve constant lookup time with minimal storage and bandwidth overhead.

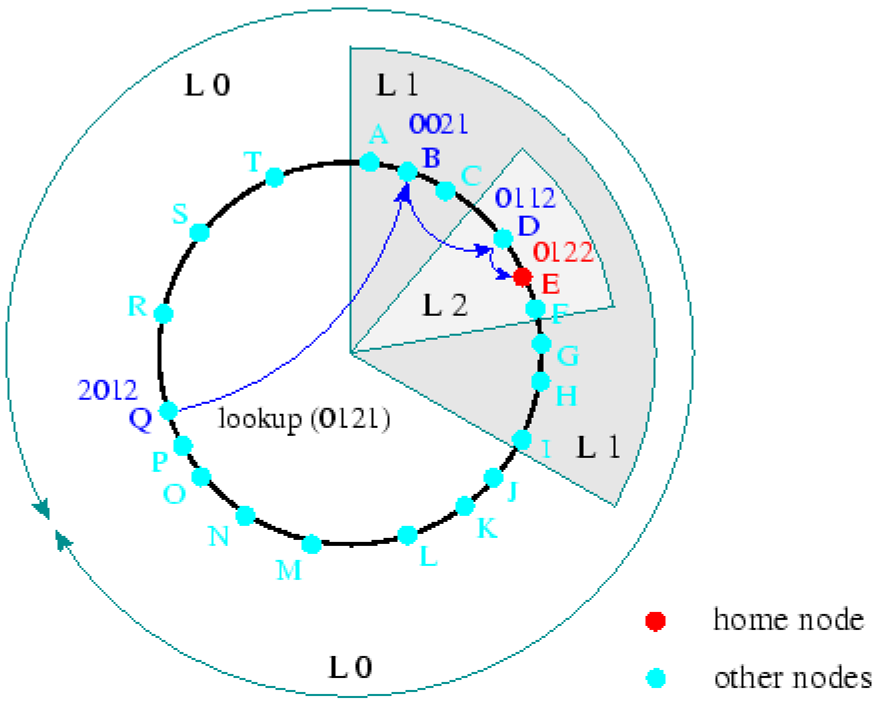

Figure 1: This figure illustrates the levels of replication in the model. A query for object 0121 takes three hops from node Q to node E, the home node of the object. By replicating the object at level 2, that is at D and F, the query latency can be reduced to two hops. In general, an object replicated at level i incurs at most i hops for a lookup.

Analytical model

An analytical model[1] can be derived for a Zipf-like query distribution, where the number of queries to the i^{th} most popular object is proportional to $i^{-\alpha}$, where α is the parameter of the distribution. The query distribution has a heavier tail for smaller values of the parameter α. A Zipf distribution with parameter 0 corresponds to a uniform distribution. The total number of queries to the most popular m objects, Q(m), is approximately $(m^{1-\alpha} -1)/(1-\alpha)$ for $\alpha\neq 1$. Using the above estimate for the number of queries received by objects, the aim is to minimize the total number of replicas with the constraint that the average lookup latency is a constant C.

Let b be the base of the underlying DHT system, M the number of objects, and N the number of nodes in the system, Initially, all M objects in the system are stored only at their home nodes, that is, they are replicated at level $k=\log_b N$. The model[1] then leads to the formulas below. Important: only the results necessary for the assignment are listed below. For the complete derivation, we refer to the URL in footnote below.

The average per node storage requirement for replication is:

[1] http://www.cs.cornell.edu/people/egs/beehive/beehive-nsdi04/node3.html

$$M[(1-\frac{1}{b})(x_0+\frac{x_1}{b}+\cdots+\frac{x_{k-1}}{b^{k-1}})+\frac{1}{b^k}]$$

The optimal solution is reached for the following values of x_i:

$$x_i^* = [\frac{d^i(k'-C')}{1+d+\cdots+d^{k'-1}}]^{\frac{1}{1-\alpha}}, \forall 0 \le i < k'$$

$$x_i^* = 1, \forall k' \le i \le k$$

$$\text{where } d = b^{\frac{1-\alpha}{\alpha}}$$

$$\text{where } C' = C(1-\frac{1}{M^{1-\alpha}})$$

We can derive the value of k' by satisfying the condition that $x_{k'-1}<1$, that is:

$$\frac{d^{k'-1}(k'-C')}{1+d+\cdots+d^{k'-1}} < 1$$

Assignment

Provide two graphs where the average storage cost per node is depicted.

- In the first graph, the Zipf paramter α varies from 0.5 to 0.9 in steps of 0.1, while C=1.
- In the second graph, the average lookup latency C varies from 1 to 2, in steps of 0.2, while α=0.9.

The other parameters remain invariant: N=10000, M=1000000, b=32.

Use an excel file to calculate the required formulas and present the graphs.

Report

This assignment should be completed **individually**. While it is OK to discuss the solution with fellow students, it is strictly prohibited to exchange and/or copy solutions.

You should hand in an Excel file, named “assignment4_firstname_lastname.xlsx”, containing two sheets.

The first sheet calculates the required formulas from the model, as well as the answer to the question on the average lookup.

The second sheet provides the requested graphs.

Use the Minerva dropbox of Filip De Turck to submit your solutions. Please do not zip nor pack this file in another way!

The deadline is Tuesday, December 9th 23:59 CET.