Friend search for distributed social networks

Sebastian Probst Eide, St Edmunds College

Originator: Sebastian Probst Eide

4 October 2010

Special Resources Required

Machine with the Erlang VM installed
Virtual Servers running on Amazon EC2 and Cloudsigma
The use of my own laptop.

Project Supervisors: Dr David Eyers and Dr David Evans

Director of Studies: Dr Robert Harle

Project Overseers: Unknown

Introduction

It is hard to get started using distributed independent social networks as it is almost impossible to find and connect with your existing friends, not knowing where and in which social network they host their profiles. The purpose of this project is to lay the foundation for a decentralized and distributed friend search engine that can be hosted alongside installations of independent social networks allowing users to easily reconstruct their social graph in the social network of their choice. The focus of the project will be on the data storage layer of the search engine. I will compare and contrast different Distributed Hash Tables that I implement in Erlang. A front end, allowing basic searches to be performed, will also be created, but mainly to provide a way of rapidly testing the data storage layer. More user oriented functionality needed to allow the project to succeed on a larger scale will be left out to limit the scope of the project.

Work that has to be done

There are a great number of distributed hash tables available [1]. I have decided to implement and compare the following three: Chord, Kademilia and Pastry. They all support the same basic operations of setting and retrieving key-value pairs, but differ in the way they route traffic and which heuristics they use for chosing nodes to route traffic to.

The main parts of the project are to:

1. Implement the data structure layer in Chord, Kademilia and Pastry using a uniform API that allows the system to use any one of the three without additional changes.
2. Implement infrastructure that facilitates testing and monitoring of the system. More specifically it should allow:
    (a) Starting and stoping virtual servers accross the different service providers for testing purposes
    (b) Start and stop search nodes accross the physical servers
    (c) Display how many search nodes are available in the system, and how they are geographically distributed
    (d) Add and remove test data from the system
    (e) Perform repetable load testing on the system
    (f) Count the number of jumps and time a key lookup needs to find a data item, in addition to calculating average and median times of lookups for fixed larger key-value datasets

[1] Wikipedia currently lists 8 different protocols

(g) Being able to eliminate and add subsets of storage nodes in a repeateble fashion to test how the different Distributed Hash Tables cope with nodes dissapearing and appearing

3. Setup a Linux image that can be run accross Infrastructure as a Service providers[2]

4. Implement a web front end to allow users to perform basic searches accross the data storage layer.

Please note that the following aspects of the search server are secondary to the project and are not part of what will be implemented:

1. Fuzzy searches allowing the user to misspell names

2. Predictive searches

3. Searches taking social circles or other metadata into account

4. Protecting against spam or malicious attacks like denial of service attacks

Starting Point

I have a reasonable working knowledge of Erlang and linux and development of web based systems. The algorithms that will be implemented have all previously been implemented in other languages, and are used in production systems, so finding information about them should be possible. I have never used Amazon EC2 or any of the other Infrastructure as a Service (IaaS) providers I plan to use, but have used similar services in the past and don't see this as being a problem.

Success criterion

I regard the project as successful if I have working implementations of the three Distributed Hash Tables that allow me to set and retrieve values based on keys accross a distributed network of machines. Additionally I should have metrics for how the performance of key-lookup varies by node count and distributed hash table type.

[2]I will use Vagrant (http://vagrantup.com) and Chef for this

Difficulties to Overcome

The following main learning tasks will have to be undertaken before the project can be started:

- To learn and fully understand the Chord, Kademilia and Pastry algorithms.
- To learn how to do network communication in Erlang other than the built in message passing.
- To learn how to use the OverSim[3] simulation tool to test and verify my implementations of the Distributed Hash Tables.
- To learn how to use Amazon EC2 and similar Infrastructure as a Service providers.

Resources

Some aspects of this project (amongst other the heuristics in Pastry which take locality into account when routing) are more interesting to test in nodes that are in geographically distinct areas. For this reason I will use server instances running on commercial Infrastructure as a Service providers like Amazon AWS and Cloudsigma. For the majority of the development cycle, local testing will be just as interesting and can be done on my development machine.

This project requires no additional file space on University machines. I will be hosting the project source code and dissertation files in a repository on github[4]. If my machine breaks down, the development can be continued on any Unix based machine that has the Erlang VM installed.

Work Plan

Planned starting date is 15/10/2000.

Below follows a list of tasks that need to be done:

1. Work through the theory behind Chord, Kademilia and Pastry and other items listed under *difficulties to overcome* (2 weeks).
2. Implement initial version of Chord, Kademilia and Pastry in Erlang (4 weeks)
3. Implement test harness to perform testing of the system (4 weeks)

[3]http://en.wikipedia.org/wiki/OverSim
[4]http://github.com/sebastian/Part-2-project

4. Implement search server using one of the Data Storage Layers implemented (2 weeks)
5. Write dissertation (6 weeks).

Michaelmas Term

By the end of this term I intend to have completed the research and learning tasks and have finished the first implementations of the Distributed Hash Tables in Erlang. In the vacation that follows I intend to get a good start on the testing harness and do a little work on the search server.

Lent Term

In the first half of this term I intend to finish the test harness and search server and spend time testing the system and solving problems.

In the second half of this term I tend to get an initial draft of my dissertation written.

Easter Term

In this term I plan to polish the dissertation. The estimated completion date is the 15th of May, leaving a couple of days to let the dissertation rest before giving it a final read and correcting last minute mistakes befor the due date on the 20th of May.