Project Title: Web Caching

Richard Andreas, Patrick Carpanedo, Igor Solomatin, Sean Brady

Github link: https://github.com/randreas/CS655_GENI_Project
Project on GENI: Web Caching

Introduction / Problem Statement

As the modern internet has grown, being connected to the internet has become a normal part of life. Many users expected quick and reliable access to their email, news websites, and other distributed applications at home, school and work. Oftentimes, connections to origin servers that host these vital services can experience high latency, either due to network constraints. Upgrading the network infrastructure to increase the bandwidth to the origin server is often expensive. In these cases, it is important to establish caches of data outside of the origin server, to reduce the load of any bottlenecks to the server. When a user requests a piece of data, they will first be directed to a local forward cache rather than to the origin server. If the cache does not have the requested data, it will recursively handle getting the data for the client from the origin server and then store that piece of data locally. In the future, if that piece of data is requested again, the local cache will return that data to the requesting users, without contacting the origin server.

Establishing a forward cache can speed up the performance of end user networks by cachin requested data for future users, but raises interesting design challenges. In this project we seek to make the investigate the following properties of caches:

- How big should the cache be (Is bigger always better)?
- How do we design an eviction policy for data objects in the cache?
- How does an empty (cold) cache compare to a full (hot) cache?

Experimental Methodology

Determining the efficacy of web caching requires testing different permutations of environments (i.e Caching software) and situations (e.g. Empty cache or Full Cache). In our project we will be comparing the efficiency of Apache Traffic Server (ATS). Figure 1 below shows our Architecture Diagram.

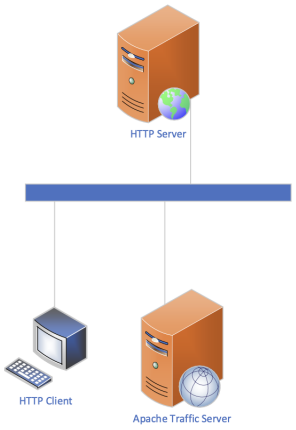

Figure 1: Geni Architecture

Clients and caches will be represented in the subnet below the internet gateway. This subnet will have a high speed infrastructure between the client server nodes and the gateway. This will allow for high speed communication between the client and caches. The link between the internet gateway and the origin server represents the “internet” as a slow speed connection from the subnet to the server.

Both caching servers will be compared in performance and the origin server in terms of latency and throughput to the client. The client will be measuring the following metrics:

1. Origin Server RTT (Response time)
2. Cache Server RTT (Response time)
3. Cache hit percentage
4. Throughput of Each Link

The experiments will be implemented in Java using curl and wget commands and Java classes to track metrics and run at the client node. The origin server will be a simple Apache HTTP server. The links speeds will also be altered to reflect varying distances from the client.

In this environment variables of cache-size, object-size, max-age etc. to test the caching performance in a myriad of situations. If time allows, advanced features such as “Read While Writer” and Hierarchical Caching will be included to see if the benefits of web caching situations are compounded or if there lies a further complexity in enabling such features. The control data will be collected from fetching from the same batch of websites without a cache present. Ideally, Multiple websites with varying distances (from client), complexity, and size will be used to conduct our test to simulate different scenarios that a normal user may encounter.

Division of Labor

1. Richard Andreas: REST API with CURL and Wget, Run Experiments
2. Patrick Carpenedo: Writing Base Code, Run Experiments
3. Igor Solomatin: Environment setup, Run Experiments
4. Sean Brady: Writing Base Code, Run Experiments