# FINAL PROJECT REPORT (CS 655 – MINI PROJECT)

## Personal Information:

Mohan Sai Krishna Thota (U72136551 & mohant@bu.edu)

Ajit Balla (U40978471 & ajit2001@bu.edu)

Srinivas Chellaboina (U93606015 & srinevas@bu.edu)

## Required Information:

- Github link:https://github.com/mohan57/CS655-MiniProject
- GENI slice name: cs
- GENIslicelink: https://portal.geni.net/secure/slice.php?project_id=0937e1e8-3dc2-40b1-8643-9d42ed0ab4ee&slice_id=80f482bf-2a88-468e-8fcb-90a41b879874&member_id=

(***please note that the resources that we have used are still in active)

## TITLE: ANALYSIS OF WEB CACHING

## INTRODUCTION:

This project is basically developed on GENI to understand the working and importance of web caching. We have developed a small server with two clients to deploy this.

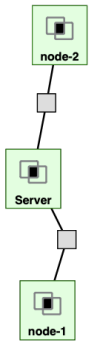
Caching is termed as a process of storing content temporarily from the previous requests, within the HTTP protocol. There are various types of caching and each of which has its own respected characteristics. Memory and application caches are known for their ability to speed up certain responses. Web caching is known for its core design feature of the HTTP protocol and its ability to minimize the traffic.

#### MOTIVATION:

We all know that there are various ways to improve the performance of any website (i.e. to clear the traffic). In which, caching is the most effective approach. Web caching is a common tool approach that is to be known to each and everyone who works online. By performing this project we'll come to know about the response time of a particular website with and without cache.

We implemented this technique on various websites and came to know about their respective time of execution.

# EXPERIMENTED METHODOLOGY:




We have created a Server(that handles cache) and two nodes to run separate files on each node. We ran Without cache file on node-1 and with Cache file on node-2. This model can also be scaled down to just a server and a single node structure and run both the files(with cache and without cache) on them. But for the sake of a friendly environment we have created two nodes to run separately. To deploy the codes we have to run with 15,30,50 requests withoutCache code and the detailed explanation is in the further sections.

- we used python to write the code files
- Geni is utilized to create the required structure
- Resources reserved:Cornell instaGeni

# RESULTS:

## 1. USAGE INSTRUCTIONS:

- create a structure using the RSpec file provided
- reserve your desired resources we have used *Cornell InstaGeni*
- on the node-1 and node-2 run the *Packages.sh* to install the required packages to run the code and build the graph
- on Server run the *Serverside.sh* to configure it
- after running the above file on server a notepad is open and you are supposed to add below lines to avoid the issues with regular port :
  -> type i and move the cursor near the files to add the below lines

*CONFIG proxy.config.url_remap.remap_required INT 0*
*CONFIG proxy.config.reverse_proxy.enabled INT 0*

->After adding the line , enter ESC

->Now type :wq and press enter to save the file

- now on the node-1 upload the withoutCache.py file using scp and run the py file using : *python withoutCache.py*

TO run withCache.py file :-

- on the Server run the following commands:

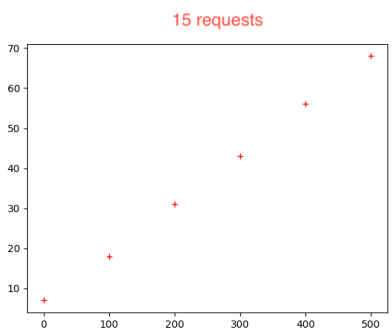
*cd /usr/bin*

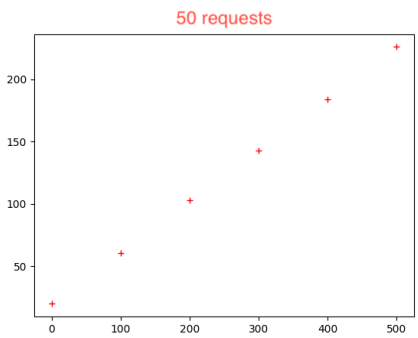
*sudo ./traffic_server start*

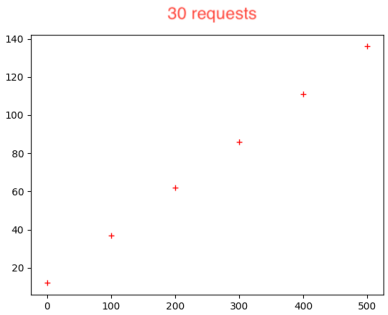
- on the node-2 upload the withCache.py file and run : *python withCache.py*

## 2. ANALYSIS:

To observe the difference on different requests we have tried with 15,30,50 using www.umass.bu . While performing the experiment we have run it on www.bu.edu / www.stevens.edu/ www.mit.edu to find the difference. During various times of the day we found that the traffic on the network is affecting the requests. but on the whole when we ran 15,30,50 request rate without cache we observed that the time taken has increased which is obvious. You can use your own website to run the code. In the below graph you can observe that the seconds increased when the requests increased.






Not to run the cache we initially ran it without any delay on 50 request rate and observed a spontaneous result. We are able to complete it in 2 seconds. To test further we have increased the delay to 300ms and still observed a spontaneous result . It completed in 10 seconds whereas it took nearly 200+ seconds on the node that has no cache .

```
mohant@node-2:~$ python withCache.py
================================================
THE INITIAL TIME BEFORE SENDING ALL THE REQUESTS
================================================
2022-12-08 18:07:06.994870
================================================
THE FINAL TIME AFTER SENDING ALL THE REQUESTS
================================================
2022-12-08 18:07:16.105995
============================================================
TIME TAKEN TO COMPLETE THE REQUESTS 10 seconds
============================================================
```

# CONCLUSION:

After executing this project we have understood that cache plays a vital role in networking. It reduces the delay and increases the efficiency. This project can later be extended by increasing the request rate and testing the cache capacity . There are different types of caches and we can use each of them to test the efficiency .

# DIVISION OF LABOR:

Mohan Thota : Geni setup and executing

Ajit Balla: documentation and research

Srinivas Ch : coding and dubbing