**DEVELOPMENT ASSIGNMENT**

**RESEARCH**

DEVCLUB RECRUITMENT ASSIGNMENT

KEVIN SHAH

2019CS10365

# **BRAIN WARM UP**

1. from zipfile import ZipFile, ZIP\_DEFLATED
2. import os
3. import time
5. name = 'test.zip'
6. directory = './ '
7. paths = []
9. for root, directories, files in os.walk(directory):
10. for filename in files:
11. filepath = os.path.join(root, filename)
12. paths.append(filepath)
14. start = time.time()
15. with ZipFile(name, 'w', compression=ZIP\_DEFLATED) as zip:
16. for file in paths:
17. zip.write(file)
18. print(time.time() - start)

I used the above python script to compress a folder containing various files. The first folder was of all relevant files of IIT, roughly 1.91GB. With no compression (just adding files to .zip), it took about 48s and with compression=ZIP\_DEFLATED, it took about 93s with final file size = 1.76GB. The second folder was of e-books in mostly pdf and epub format, roughly 3.73GB. With no compression, it took about 49s and with compression=ZIP\_DELFATED, it took about 160s and final size = 3.14GB. Most pdfs and jpgs had their size reduced by less than 20%.

Considering that heavy traffic would be encountered near exams (starting around 2-3 days before start of exams), generating a zip file for each request would be impractical since it would take a lot of computational resources to compress potentially hundreds of courses per hour during peak time.

Courses that enrol many students such as those compulsory for all freshers should be kept compressed for at least 10 days before and during exams with high compression ratio so as to reduce storage space and download time. These will also have larger size due to more potential contributors and more download requests. Compressed versions of advance courses such as ZZZ700 can be generated on demand since fewer students will download them and they also be relatively smaller. Considering storage space and computation power, remaining courses can be kept as compressed or can be generated on demand due to fewer download requests and smaller sizes.

During other times, it would seem feasible to generate a compressed version on demand due to a much smaller demand.

# **HOW DOES A WEBSITE WORK?**

A website is a collection of web pages, images and other content that are linked together. A website can be made up of a single page or many pages. Each page has its own text, images and other content. All web pages and content are then placed in a folder and stored on a web server. Each web page is written in codes and these codes describe the layout, format and content on the page.

1. To access a website, a client enters a domain name in a web browser.

Clients are internet-connected devices (for example, a computer connected to your Wi-Fi) and web-accessing software available on those devices (usually a web browser like Firefox or Chrome). A domain name is a unique address that one types into a web browser address bar to go to a website.

1. The browser then goes to the DNS server to find the IP address of the domain name.

DNS (Domain Name Servers) map a domain name to its IP address. When one enters a web address in a browser, the browser looks at the DNS to find the website's real address so that it can send HTTP messages to the right place.

1. An HTTP request is sent to the server across the internet using TCP/IP.

HTTP (Hypertext Transfer Protocol) is an application protocol that defines a language for clients and servers to speak to each other. TCP/IP (Transmission Control Protocol/Internet Protocol) are communication protocols that define how data should travel across the web. A web server is the computer that receives the request for a web page sent by a browser. Servers store webpages, images and other content. Unless a website is uploaded to a web server, it is not accessible or viewable on the Internet.

1. The server then may approve the client’s request and start sending the website’s files to the client’s browser in small chunks called data packets. These packets are assembled into a complete website by the browser.

One can accomplish showing an html page over Wi-Fi using python http.server module.

1. Open command prompt and change directory to folder containing html page.
2. Run “python -m http.server [port] –bind[IP]”

(replace [port] with port of your choice, ex:8000, replace [IP] with IP of your choice, ex:127.0.0.1)

1. In your web browser, navigate to “IP:port/index.html” (ex: 127.0.0.1:8000/index.html, index.html is name of html file).

The above method only works if both the devices are on the same network (Wi-Fi or hotspot) since the html page is stored only locally, it can only be accessed by the server computer or through a request from a client computer on the same network.

Apache: Apache is a web server application that establishes a connection between a server and a browser and delivers files back and forth. The communication takes place through HTTP. It is highly customizable and has a module-based structure. It uses a thread-based structure and cannot manage more than 10000 connections at a time.

Nginx: Nginx is a web server application. It is also used as a reverse proxy, HTTP cache and load balancer. It handles all requests in a single thread and so offers low memory usage and high concurrency. It can handle more than 10000 connections at a time.

# **DATABASE SYSTEMS**

A database is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc.

A database is used for:

1. efficient storage of data (ex: deleting duplicate copies)
2. avoiding inconsistency in data (ex: errors in manual copying, delay in updation)
3. efficient and quick access (all the data is stored in one place and can be easily accessed from a server)
4. secure storage (ex: passwords for accessing restricted data)
5. concurrent access
6. backup and recovery easily possible.

A NoSQL (non-relational) type database will be better to model this situation.

Considering the information to store i.e. personal information, projects, research data, a SQL type database is not ideal since it has a predefined schema. Although personal information (name, qualification etc.) and project information will be easy to handle on a SQL type database, research data will be difficult to handle. Research data will have vastly different number and types of data fields which can be easily handled on a dynamic NoSQL type database.

# **JAVASCRIPT TIMER**

setTimeout() and setInterval() are not a good choice for all applications because of some obvious restrictions.

One restriction for setInterval() is that time goes on while an alert is shown. If you had a setInterval() function that creates an alert, the timer will run even when the alert is not dismissed. This could have an effect an unintended effect of the time interval seeming shorter since some time is spent on reading the alert.

Ex:

1. let timer = setInterval(() => alert(‘message’), 2000);
2. setTimeout(() => clearInterval(timer), 5000);

The time between successive alerts will appear to be less than 2 second since some time is required to close the alert. Also, if the alert is not closed after 2 seconds, the second alert will not be shown until the first one is closed. A nested setTimeout() can instead be used to achieve the desired effect.

Some other restrictions are:

1. they are resource intensive
2. they have a delay of varying milliseconds
3. they ignore errors.

An alternative is window.requestAnimationFrame() which is also less resource intensive.

Computer security vulnerabilities such as Meltdown and Spectre work on the principle that it takes less time to access data that has been cached vs data to be retrieved from memory. Due to the above difference, it can infer protected information. Thus, to prevent an attack over the internet and JavaScript, JS timer’s performance is intentionally degraded.