DEVELOPMENT AND IMPLEMENTATION OF A WEB BROWSER LOG ANALYSER

A thesis submitted in partial fulfillment of the requirement for the degree of

Master of Computer Application

Department of Computer Science and Engineering

Jadavpur University, Kolkata

By

RAJDIPTA BARMAN

Registration Number: 137319 of 2016-2017

Examination Roll Number: MCA196009

Under the guidance of

Prof. Chandan Mazumdar

Department of Computer Science and Engineering

Faculty of Engineering and Technology

Jadavpur University, Kolkata-700032

India

May, 2019

JADAVPUR UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

CERTIFICATE OF RECOMMANDATION

This is to certify that the thesis entitled "**DEVELOPMENT AND IMPLEMENTATION OF A WEB BROWSER LOG ANALYSER**" has been satisfactorily completed by Rajdipta Barman (University Registration No.: 137319 of 2016-17, Examination Roll No:MCA196009).It is a bonafide piece of work carried out under my guidance and supervision and be accepted in partial fulfillment of the requirement for the degree of Master of Computer Application, Department of Computer Science and Engineering, Faculty of Engineering and Technology, Jadavpur University, Kolkata.

_____

Prof. Chandan Mazumdar (Thesis Supervisor)

Department of Computer Science and Engineering

Jadavpur University, Kolkata-700032

Countersigned

_____

Prof. Mahantapas Kundu

Head,Department of Computer Science and Engineering,

Jadavpur University, Kolkata-700032

_____

Prof. Chiranjib Bhattacharjee

Dean,Faculty of Engineering and Technology,

Jadavpur University, Kolkata-700032

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL

This is to certify that the thesis entitled "**DEVELOPMENT AND IMPLEMENTATION OF A WEB BROWSER LOG ANALYSER**" is a bonafide record of work carried out by Rajdipta Barman in partial fulfillment of the requirements for the award of the degree of Master of Computer Application in the Department of Computer Science and Engineering, Jadavpur University during the period of January 2019 to May 2019. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

_____

Signature of Examiner

Date:

_____

Signature of Supervisor

Date:

ACKNOWLEDGEMENT

I am pleased to express my deepest gratitude to my thesis guide, Prof. Chandan Mazumdar, Department of Computer Science and Engineering, Jadavpur University, Kolkata for his invaluable guidance, constant encouragement and motivating words during the period of my dissertation.

I would also like to thank Dr. Anirban Sengupta, Principal Research Engineer CDCJU, Mr. Subhomoy Karmakar, Research Engineer CDCJU and my classmate Mr. Rishi Dey for sharing their knowledge and experience with me and also their immense support and co-operation.

I am thankful to all the teaching and non-teaching staff who helped me to have a smooth journey during the time of my research.

Last but not the least; I would like to thank my family members, classmates, seniors and friends for giving me constant encouragement and mental support throughout my work.

_____

Rajdipta Barman

University Registration No. : 137319 of 2016-17

Examination Roll No. : MCA196009

Master of Computer Application

Department of Computer Science and Engineering

Jadavpur University

Contents

1 Introduction
1.1 Advantages of Log file Analysis……...4
1.2 Limitations of Log File Analysis……...5
1.3 Objective……5
1.4 Organization of the thesis……..5

2 Web Browser Software
2.1 What is Web browser……6
2.2 The process of browsing……...6
2.3 Features of a Web Browser……....7
2.4 Example of Web Browsers……8
2.5 How a Web Browser works……...12

3 Structure of Web Browser logs
3.1 What is browser log……15
3.2 Collection of browser log……...15
3.3 Structure of Google Chrome browser log……..19

4 Web browser log analysis
4.1 Log Analysis……31
4.2 Purpose of wen log analysis……31
4.3 Collected log……33
4.4 Analysis……...34
4.5 Proposed Analysis for future work……37

5 Implementation
5.1 Introduction……39
5.2 Feature of analyser……....39
5.3 Architecture……41
5.4 Technologies Used……42
5.5 Future Work……42
5.6 Conclusion……..45

Chapter 1:

1. Introduction

Log files are generated by web servers or different applications which consist of various figures or records about the use of any website or web browser or any system as well. System generated log files include system log, server log etc. Web browser log files can be generated as well which depicts different aspects of web browsing such as URLs, timestamps, type of event and many more.

Huge records stored in the file can be used for understanding many statistics about pages or systems and implementing new features or making predictions. This is where the process of log file analysis comes into practice. Thorough analysis and decision making from the log files is an important aspect of ensuring system security and integrity as well.

For system generated log files, analyses can help in mitigating security problems, auditing, performance optimization etc. Alerts can be generated if any unwanted event type is detected in the log.

Web browser log files can, too, help in solving security problems and perform necessary operations in predicting and profiling of users. Unauthorized entry to a webpage or unethical activities in a network can be flagged within an organization. Securing log from several turns the problem of analysis into a problem of big data because of the sheer amount of that need to be processed. Different methods and tools of Big Data are used in log analysis tasks to make those systems more efficient and adaptable to different kinds of data in real time. Huge amount of data corresponds to bigger processing time. This problem is constantly being mitigated through various kinds of tools and algorithms which are being developed by data scientists all around the world.

Data visualization is another important aspect of data analytics and decision making. Graphical representations help in better understanding of the data. As number of web browser users are increasing day by day,

need for analysing the log files of web browser is increasing. This will not only be of help in some technical project, but also it can be a tool for carrying out social experiments as well.

1.1 Advantages of log file analysis

- Realignment of historical data: Web browsers can continuously record log files. If the files are saved again and again, they can be evaluated flexibly.
- When a website is accessed from a server, the firewall does not interfere. The log file can therefore log the access and other records exactly.
- Simple formatting: If the log file is not too large, the data can be read out and segmented with conventional data processing programs such as Excel. Therefore, no complex program solutions are required.
- Log file analysis does not only help in understanding activities of users. Results of different analyses can be helpful in working and content of different websites, based on user preference.
- Security problems in different organizations can be detected and mitigated efficiently by log file analysis.

The web server can store the following data in a log file:

- The operating system used by the user or client
- The browser used
- The time and date of access
- Commands requested by the server
- The protocol, e.g. https
- The number of bytes transferred
- The URL previously called up by the user
- The server's response
- The file name and the file path
- The IP address or DNS address

But in case of a web browser's log file, other important details are stored as well. These will be of immense help in this project where the objective will be to analyse different aspects of user activities.

1.2 Limitations of Log file Analysis

- If amount of data increases i.e. long hours of browsing is done in several machines log files can become of huge size. Analysing it in real time is a challenge.

- Not all the data stored in a log file is relevant or useful towards achieving an objective. Storing those data can consume space in the database. If it is to be left out, parsing should be done which can bring complexities in the development of the tool.

- Log files of different browser are generated in different formats. Bringing it under one format and analysing it can also be a challenge in the process.

1.3 Objective

Acquiring log files from different browsers and doing necessary analyses on the log files is the objective which will be achieved by making a tool called Web Browser Log Analyser. This tool will have other necessary features including visualisation option for the efficient processing and analysis of the files.

1.4 Organization of the Thesis

Chapter 2 describes the different web browsers used around the world, while chapter 3 discusses about the structure of the web browser log files collected form Google Chrome. Chapter 4 describes all the implemented and proposed analysis. The Last chapter discusses the intricate details of the implementation process of the log analyser.

Chapter 2:

2. Web Browser Softwares

2.1. What is a Web browser?

A web browser, or in short, a browser is a software application used to locate, retrieve and display content on the World Wide Web, including webpages, images, video and other files. They translate requested web pages and websites to human-readable content using Hypertext Transfer Protocol (HTTP). Browsers can also display other protocols and prefixes, such as secure HTTP (HTTPS), File Transfer Protocol (FTP), email handling (mailto:), and files (file:). In addition to these, most browsers also support external plug-ins required to display active content, such as in-page video, audio, game content etc.

2.2. The process of "browsing"

Every single web page is identified by a unique Uniform Resource Locator (URL), enabling browsers to retrieve these resources from a web server and display them on a user's device. All the URLs start with either "http:" or "https". It signifies that the browser will retrieve the pages with Hypertext Transfer Protocol. "https" signifies that the communication between the browser and the web server is 'secure' or encrypted. When an user types an URL and presses enter, the browser retrieves the corresponding webpage from the World Wide Web and the browser's rendering engine displays in on the user's device. This process is described elaborately later in this chapter.

2.3. Features of a web browser

Most of today's web browsers have many common features regarding UI, privacy and security. Although each type of web browsers offer unique features that make them distinct.

The UI features of browsers are:

- An address bar for entering URL.
- Navigation buttons which allow users to go back to the previous page or go forward to the next one.
- A reload or refresh button to reload the current web page.
- Provision to open multiple pages or tabs at the same time(It can done in separate browser windows as well).
- Home page button to go back directly to user's home page.
- Default search engine can also be set from 'Settings' option.
- Multiple features can be added to the home page from 'Settings' option such as 'Bookmark' list.

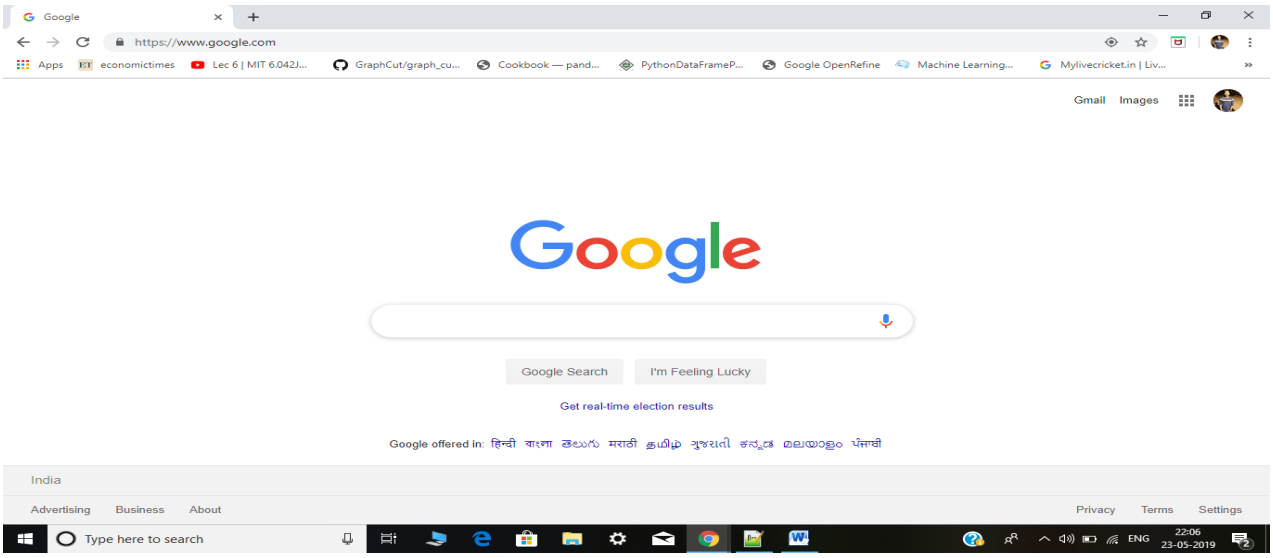


Fig 1: Google Chrome browser home page with google.com set as default search engine.

Privacy and security features of a typical web browser

- During browsing, cookies are stored by the browsers from different websites which may contain login credentials and other important information. The web browser removes cookies when the browser window gets closed.
- "Incognito" mode can be used to stop tracking of user activities.
- Various security protocol provide authentication, privacy, data integrity between two communicating computer applications such as TLS(Transport Layer Security), SSL(Secure Socket Layer) etc.

2.4. Examples of web browsers

There are a number of browsers in use today. The most popular ones are,

- Google Chrome
- Mozilla Firefox
- Google Chromium
- Microsoft Edge
- Internet Explorer

Mozilla Firefox 

Mozilla Firefox was developed in 2002 by Mozilla Corporation. It's an open source web browser and it is available in multiple operating systems. It is the second-most widely used web browser in the world today with 11.78% of usage share. In the recent updated version, Firefox has improved download manager, stronger phishing and malware protection.

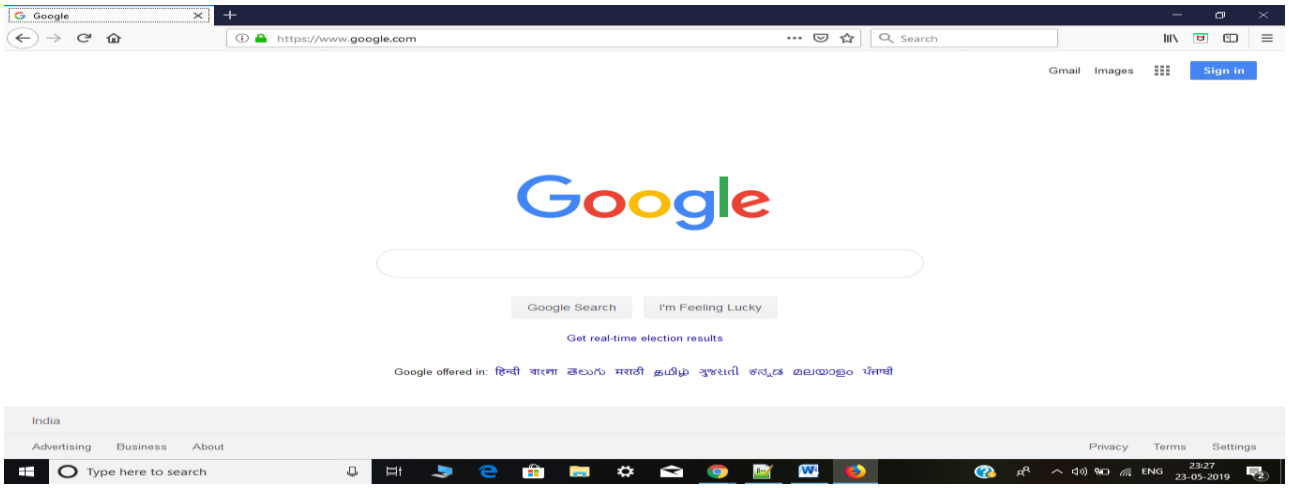


Fig 2: UI of Mozilla Firefox

Microsoft Edge

Microsoft Edge was released in July 2015 as a replacement for Internet Explorer. It is the default browser for Windows 10 operating systems. Edge is exclusive to Windows 10 and cannot be used on previous Windows versions. It has many improved features such as,

- A feed of personalized content.
- Built-in Cortana assistance.
- Reading support.
- Snoozing tabs for later.
- Option to scribble over web pages.
- Better speed.

Google Chromium

It is an open-source fully functional web browser and software project run by Google. It was released in September 2008.It is similar to Google Chrome but has less features than Chrome. It is essentially an

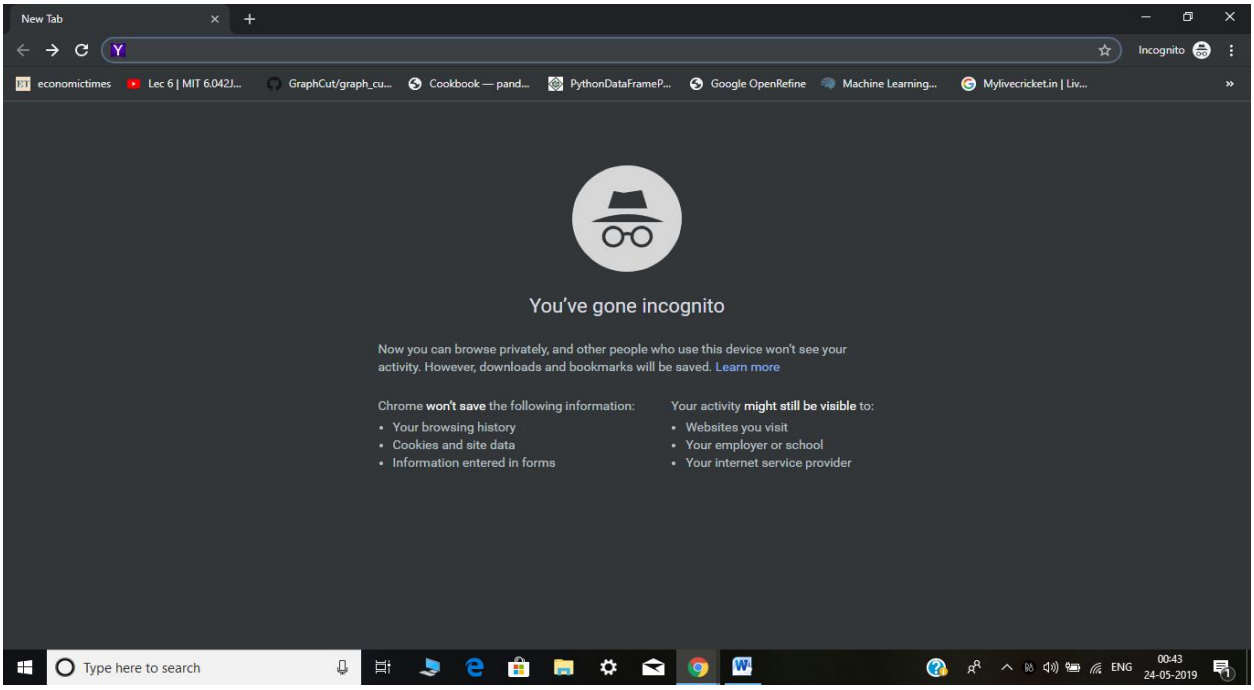
open-source browser project that aims to build a safer, faster, and more stable way for all Internet users to experience the web. Chromium runs on Windows, MAC OS and Linux.

Google Chrome

Google Chrome is the most widely used web browser in the world in almost 60% usage share. It was developed by Google and released in December 2008. It provides synchronization with other Google products and services as well. It can be used in Windows, Linux, macOS, iOS, and Android.

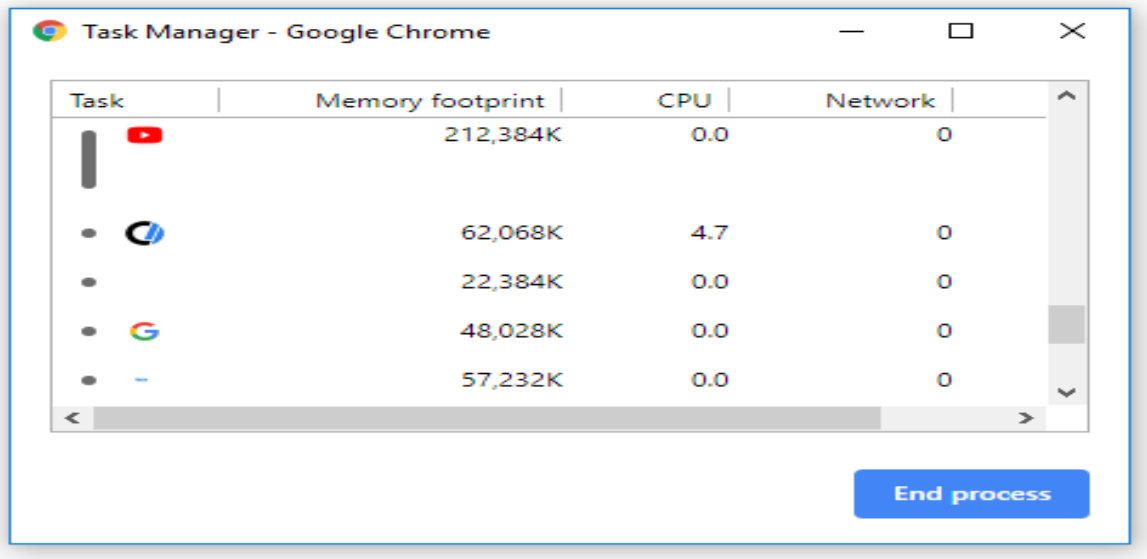
Plethora of useful features have made this the most popular browser and its superiority lies from user security to UI and compatibility.

- **Incognito Mode**: This feature allows users to browse websites privately, without saving anything in history. It can be done from “Customize and control Google Chrome” option or we can use a short cut (ctrl+shift+n) to activate incognito mode.



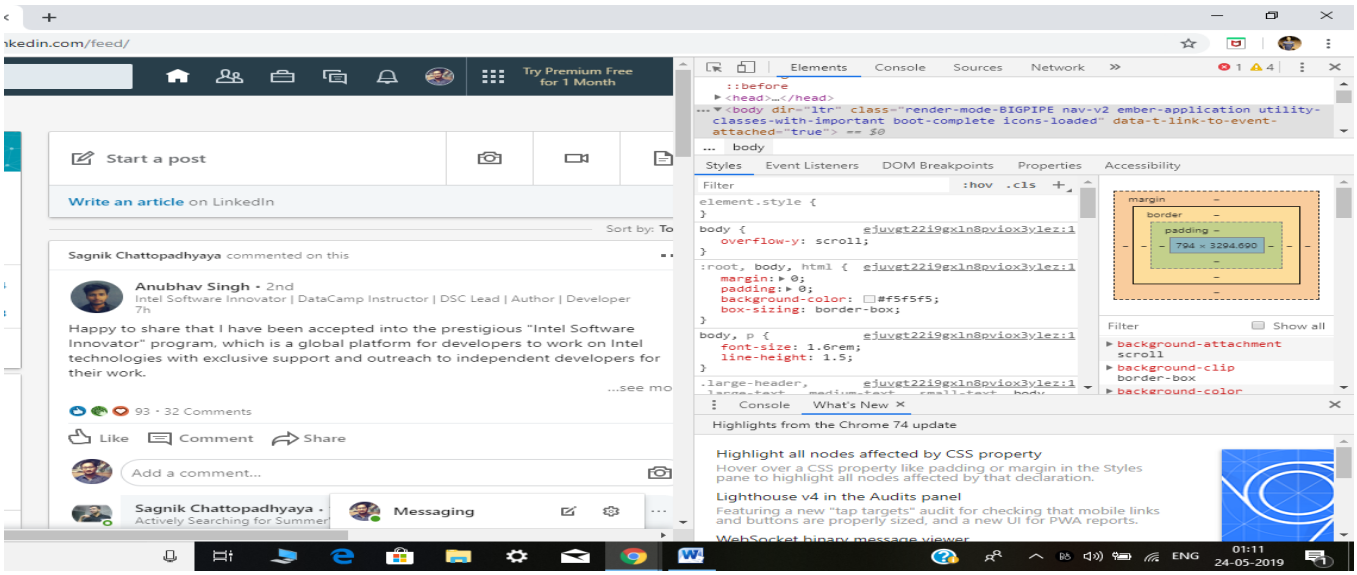
- **Printing an web page:** Users cannot always print any document straight from the website. Chrome enables the users to save the document in different forms, as per requirement. It increases user-comfort.

- **Task manager:** Chrome has its very own task manager.Users can track each and every task in the task manager. Typing Shift+ESC brings forward the task manager.



- **Developer tools**: Chrome's Developer Tools enable the user to closely analyse all the visual, interactive, and technical components of website locations. The developer view allows users to navigate the web on one side of the window, and inspect the resource's components and attributes on the other.

Inspecting network activity, securing debugging logs, understanding the web page structure and analysing the HTML code of the web page can be done from Developer tools. Users can run Audits and get a report of the website's performance.



2.5 How a web browser works:

A web browser is made up of a group of components or structured codes which perfoms a series of tasks to retrive a requested web page and display it on the screen.Nowadays, browsers can interpret and display HTML web pages, applications , Javascript, AJAX and other contents hosted on web server. So, the groups of code are invoked as per requirement and follows a particular order for a task.

To understand what happens when a user requests for a webpage , knowledge of different components of a browser is needed. The main components are,

- User Interface(UI): This is the component which interacts with the user. As mentioned earlier in the chapter, this consists of address bar, navigation buttons, home page, reload button by defualt. Other features can also be shown by adjusting the settings.

- Browser Engine: Upon receiving inputs from various UIs, it makes queries and manipulates the rendering engine.Basically, it works as an connector between user interface and rendering engine.

- Rendering Engine: It carries out the task of rendering the requested web pages on the screen after translating HTML, XML documents and images that are formatted using CSS. Different we browsers use different rendering engines; Firefox has Gecko, Google Chrome has Blink etc.

- Networking: This component handles all aspects of internet communication and security and retrieves the URLs using internet protocols suchs as http, https ot ftp.

- Javascript Interpreter: It interprets and executes the JavaScript code embedded in a website and send the results to the rendering engine so that it can be displayed .

- UI backend: This one is used for forming basic widgets like combo boxes and widows.

- Data Storage: Web Browsers support different storage mechanisms such as localStorage, IndexedDB, WebSQL and FileSystem. A small database gets created on the local drive of the machine where the browser is installed. User data such as cache, cookies, bookmarks are managed by it.

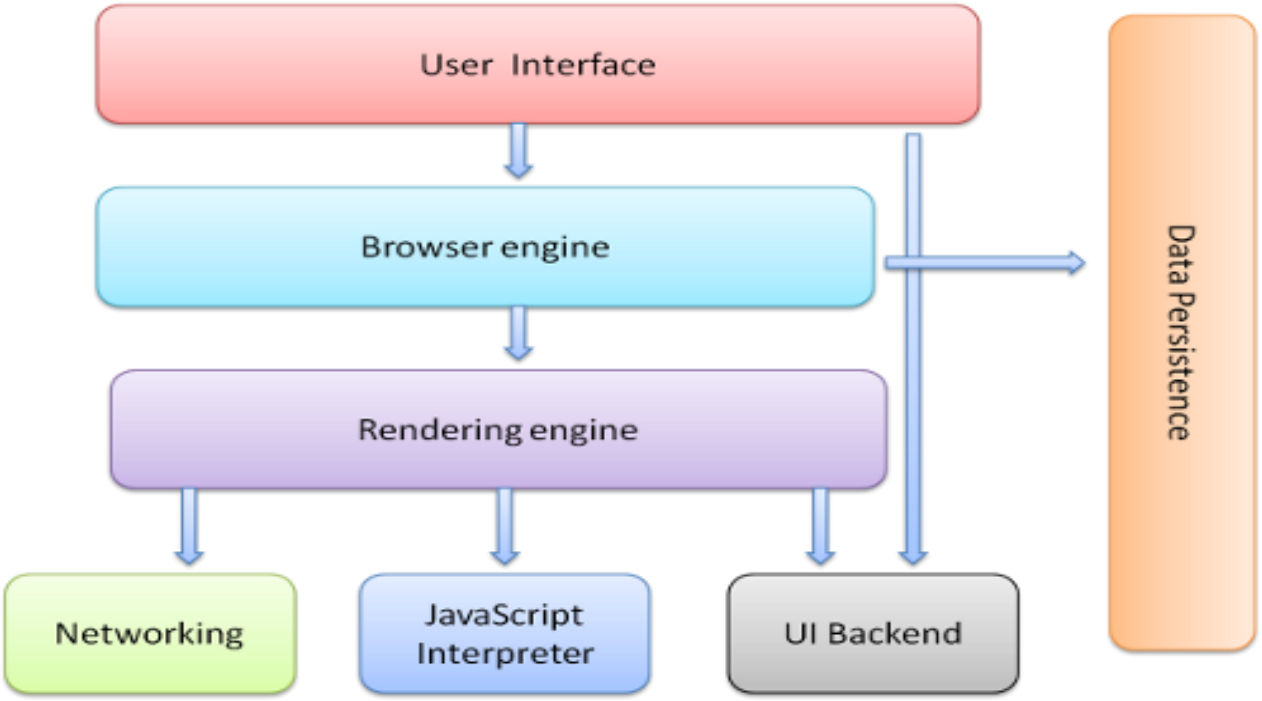


Fig: Components of a typical web browser

The process starts with URLs being entered on the address bar. As a response, web browser communicates and sends the contents. The network layer starts sending contents of the requested web pages to the rendering engine.HTML documents gets parsed and elements are converted to 'DOM nodes' in a tree called 'Content tree' or 'DOM tree'.

At the same time, another tree called the 'render tree' gets created which stores the visual element of the page in the order in which they will be displayed. After that the 'layout' process takes place which is basically giving each node of the render tree the exact position and size in the screen.

Subsequently, the render tree is traversed and rendering engine's 'paint()' method gets called which displays the contents on the screen. It is done by the UI backed layer.

Chapter 3:

3. Structure of Web Browser Logs

3.1. What is browser log?

During the course of browsing the internet, the web browser can store various intricate records of the browsing process such as request type, event type, event id, URL, timestamp, content type, filename in case of downloads, source id etc. in different formats. This file is called the browser log file. Analysing the data stored in a log file, detection of malicious and unauthorized activities, user profiling and monitoring and many other important tasks can be done.

All the browsers have their own way of storing browser log. But it does not get generated until we enable the browser to do so. It may be retrieved from the developer tools and can be stored in separate files in the local memory.

Log files are often very large and can have complex structure. Although the process of generating log files is pretty simple and straightforward.

3.2. Collection of Browser Log

All the browsers mentioned in the previous chapter have their own way of enabling a retrieving log files. Processes for the same are mentioned below.

3.2.1 MOZILLA FIREFOX

For Windows operating systems, Command Prompt window needs to be open first. Depending on the configuration of the OS different commands can entered.

For 64bit Windows:

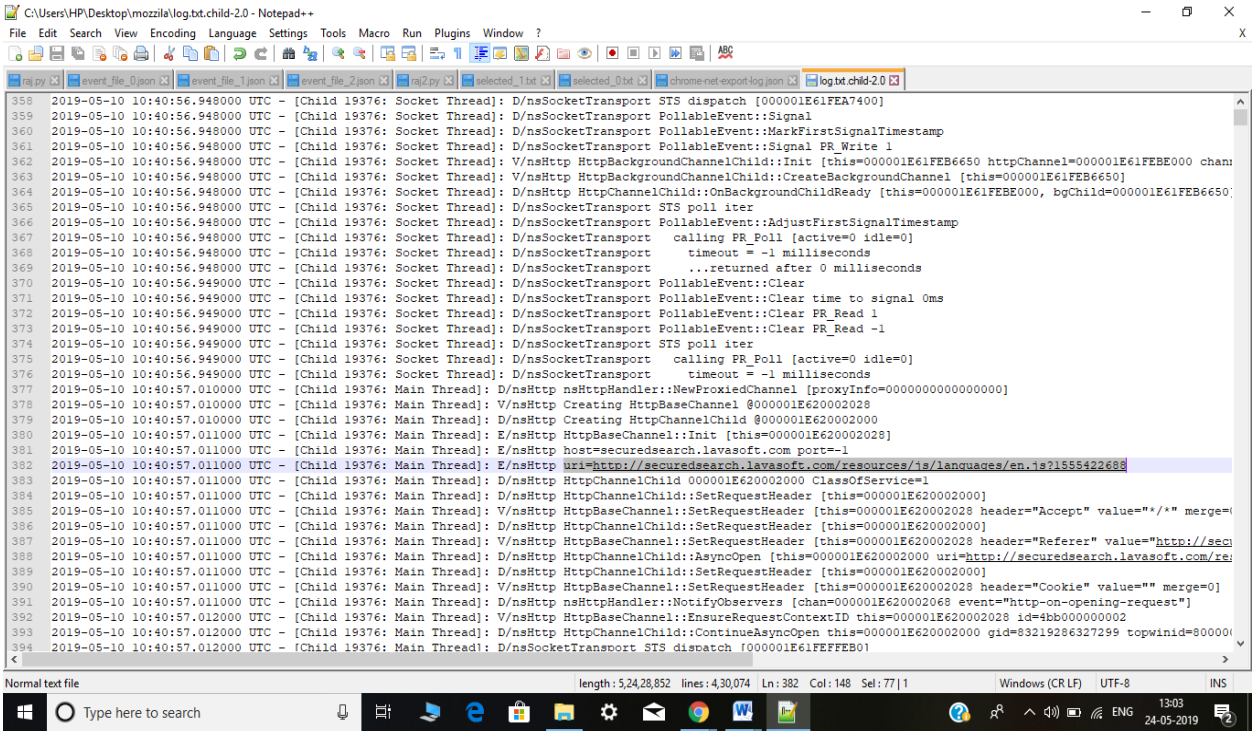
- set MOZ_LOG=timestamp, rotate:200,nsHttp:5,cache2:5,nsSocketTransport:5, nsHostResolver:5
- set MOZ_LOG_FILE=%TEMP%log.txt
- “c:\Program Files\Mozilla Firefox\firefox.exe”

For 32bit Windows:

- set MOZ_LOG=timestamp, rotate:200,nsHttp:5,cache2:5,nsSocketTransport:5, nsHostResolver:5
- set MOZ_LOG_FILE=%TEMP%log.txt
- “c:\Program Files (x86) \Mozilla Firefox\firefox.exe”

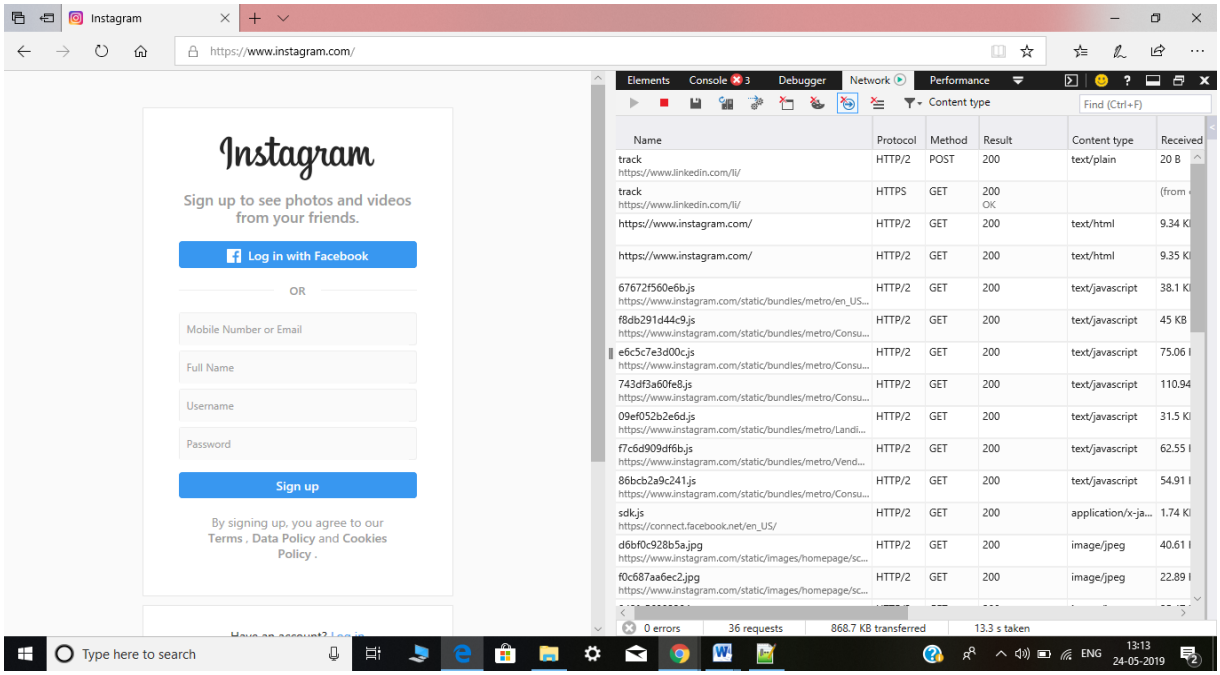
These commands will only work if Firefox has been installed in the default location and C is the start-up drive of the machine. Although necessary adjustments can be made in the commands if the log files are to be stored in a different location or Firefox has been installed in a different location.

The log records of firefox gets stored in .O format. The O file type is a standard executable and linking format for executables, object code, shared libraries, and core dumps. The format of the log file is shown below:



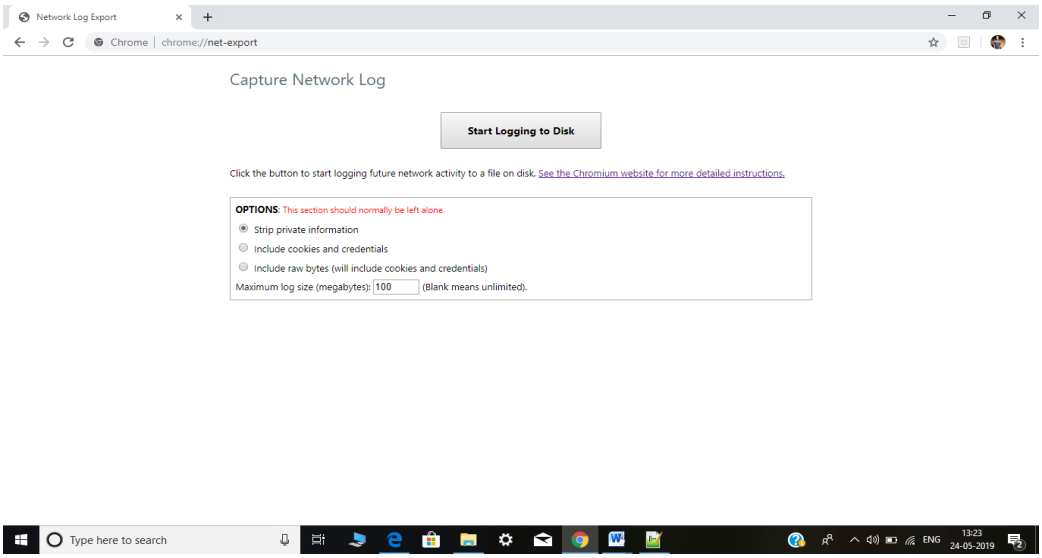
3.2.2 MICROSOFT EDGE

The browser log of Microsoft Edge is collected from developer tools. After opening the Developer tools, “Preserve log” box must be ticked. By clicking on the “Network” tab log records can be seen in this manner.



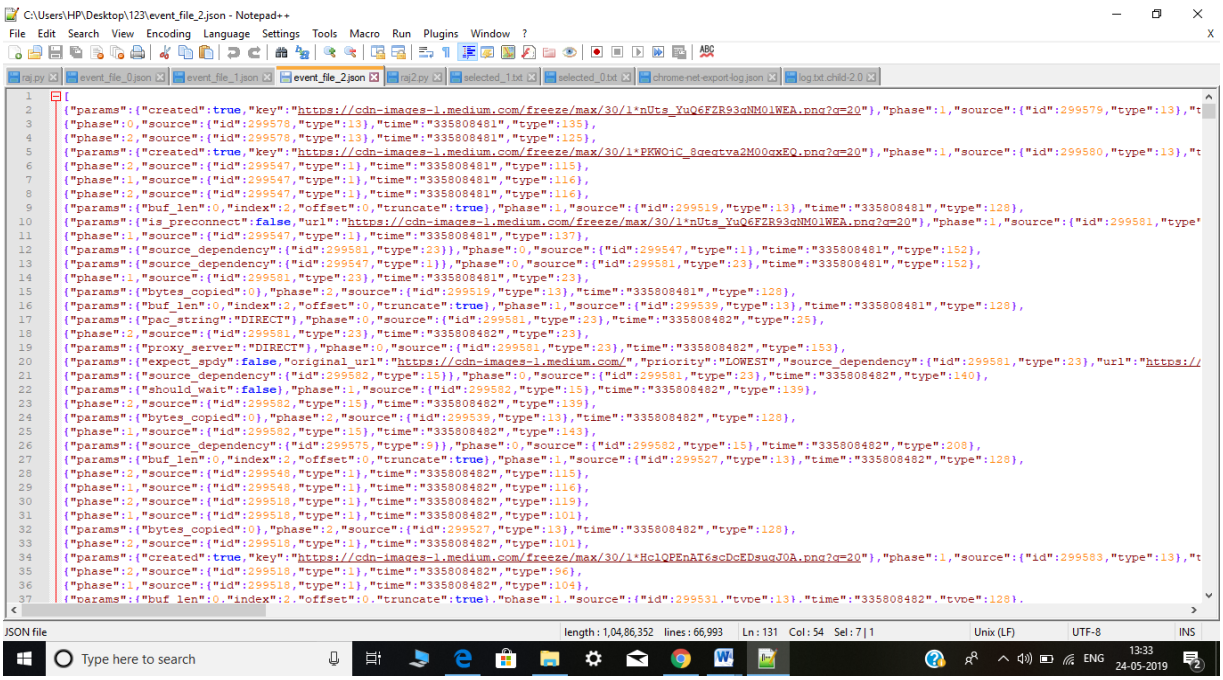
3.2.3 GOOGLE CHROME AND CHROMIUM

The process of storing log files for these two browsers is same. After opening the browser An URL must be entered i.e "chrome://net-export". This redirects to a page that looks like this:



This is the page from where the browsers can be enabled to store log files by clicking on the "Start Logging to Disk" button. Also, the maximum log file size can be set. Default value for the same is 100 MB. Upon reaching the limit, logging automatically stops. If not, we can click on the "Stop Logging" option. The process of logging can be restarted from the very same page.

The event files get stored in specified location in JSON(Java Script Object Notation) format. As new tabs are opened in the browser separate event files get generated. Logging stops when individual limit of the event files are reached.



3.3 Structure of Google Chrome Browser Log

This particular log file is stored in JSON format. Different tags are used to describe the events such as “priority”, “source”, “type”, “time”, “url”, “key” etc. Depending on the type of event, different tags get generated. The types of log records are described below.

TYPE 1:

{"params":
{
"created":true,
"key":https://cdn-images-1.medium.com/freeze/max/30/1*nUts_YuQ6FZR93gNM01WEA.png?q=20
},
"phase":1,
"source":
{
"id":299579,
"type":13
},
"time":"335808481",

"type":125
}

- “created”:

Types of values- “true” and “false”

Significance: Related to disk/memory cache. Value is true if entry is created.

- “phase”- Types of values- 0 , 1 and 2

This enumeration can be one of BEGIN, END, NONE. To log the duration of a URL_REQUEST, BEGIN and END is used specify the phases. Absence of an event is specified by NONE.

- **“source”-**

```
"source":
  {
   "id":299579,
  "type":13
 },
```

Type: Object

The source object itself is comprised of two sub-fields “id” and “type”. The “id” is unique across all types. The type field is included as a convenience so that processing the event stream can be done in a stateless manner. Type is the ID of the event type.

Events for the different values of “type”:

“type”:0 signifies SOURCE_TYPE(URL_REQUEST)

“type”:1 signifies SOURCE_TYPE(PAC_FILE_DECIDER)

“type”:2 signifies SOURCE_TYPE(HTTP_PROXY_CONNECT_JOB)

“type”:3 signifies SOURCE_TYPE(SOCKS_CONNECT_JOB)

“type”:4 signifies SOURCE_TYPE(SSL_CONNECT_JOB)

“type”:5 signifies SOURCE_TYPE(TRANSPORT_CONNECT_JOB)

“type”:6 signifies SOURCE_TYPE(WEB_SOCKET_TRANSPORT_CONNECT_JOB)

“type”:7 signifies SOURCE_TYPE(SOCKET)

“type”:8 signifies SOURCE_TYPE(HTTP2_SESSION)

“type”:9 signifies SOURCE_TYPE(QUIC_SESSION)

“type”:10 signifies SOURCE_TYPE(QUIC_CONNECTION_MIGRATION)

“type”:11 signifies SOURCE_TYPE(HOST_RESOLVER_IMPL_JOB)

“type”:12 signifies SOURCE_TYPE(DISK_CACHE_ENTRY)

“type”:13 signifies SOURCE_TYPE(MEMORY_CACHE_ENTRY)

“type”:14 signifies SOURCE_TYPE(HTTP_STREAM_JOB)

“type”:15 signifies SOURCE_TYPE(EXPONENTIAL_BACKOFF_THROTTLING)

“type”:16 signifies SOURCE_TYPE(UDP_PACKET)

“type”:17 signifies SOURCE_TYPE(CERT_VERIFIER_JOB)

“type”:18 signifies SOURCE_TYPE(PROXY_CLIENT_SOCKET)

“type”:19 signifies SOURCE_TYPE(BIDIRECTIONAL_STREAM)

“type”:20 signifies SOURCE_TYPE(NETWORK_QUALITY_ESTIMATOR)

“type”:21 signifies SOURCE_TYPE(HTTP_STREAM_JOB_CONTROLLER)

“type”:22 signifies SOURCE_TYPE(CT_TREE_STATE_TRACKER)

“type”:23 signifies SOURCE_TYPE(SERVER_PUSH_LOOKUP_TRANSACTION)

“type”:24 signifies SOURCE_TYPE(QUIC_STREAM_FACTORY_JOB)

“type”:25 signifies SOURCE_TYPE(HTTP_SERVER_PROPERTIES)

“type”:26 signifies SOURCE_TYPE(HOST_CACHE_PERSISTENCE_MANAGER)

“type”:27 signifies SOURCE_TYPE(TRIAL_CERT_VERIFIER_JOB)

“type”:28 signifies SOURCE_TYPE(COOKIE_STORE)

“type”:29 signifies SOURCE_TYPE(HTTP_AUTH_CONTROLLER)

- **“time”-** The time in milliseconds when the event occurred. This is a time tick count and not a Unix timestamp. However it is easily convertible given the time tick offset.

TYPE 2:


```
{
"params":
{
"priority":"HIGHEST",
"url":https://glyph.medium.com/font/78ce731/0-
3j_4g_6bu_6c4_6c8_6c9_6cc_6cd_6ci_6cm/fell-400-normal.woff },
"phase":1,
"source":
{
"id":299642,
"type":1
},
"time":"335808965",
"type":2
}
```


- **“priority”:**

Types of values- “HIGHEST”, “LOW”, “LOWEST”, “MEDIUM”, “IDLE”.

Specifies resource prioritization to better handle how websites are currently built and to optimize the loading for the user experience.

- **"url":**

The URL actually being used for the steam job, it is different from “original_url” if an alternate service is being used.

➢ "source_dependency":
{
"id":299581,
"type":23
}

Parameters of “source_dependency” specifies the source identifier for the request of the HTTP_STEAM_JOB.

- The last tag of a particular log record is “type”. This can have 402 values starting from 0 to 401. Each value signifies a different event such as EVENT_TYPE(CANCELLED), EVENT_TYPE(REQUEST_ALIVE), EVENT_TYPE(HOST_RESOLVER_IMPLE_REQUEST), EVENT_TYPE(HOST_RESOLVER_IMPL_IPV6_REACHABILITY_CHECK), EVENT_TYPE(HOST_RESOLVER_IMPL_CHACHE_HIT), EVENT_TYPE(TCP_CONNECT), EVENT_TYPE(SSL_CONNECT) and many more.

  Some of the events seen during the collection of log are described afterwards. Detailed description of the tags that get created during the course of logging process are attached also.
- **“type”:0** signifies EVENT_TYPE(CANCELLED)

An event got cancelled (What has been cancelled can be determined based on the log context around it.)

- **“type”:1** signifies EVENT_TYPE(FAILED)

Something failed (What has been cancelled can be determined based on the log context around it.)
The event has the following parameters:
{
"net_error": <The net error code integer for the failure>,
}
Acquired values from the log file for "net_error" are 2,118,109,804.

- **"type":2** signifies EVENT_TYPE(REQUEST_ALIVE)

Marks the creation/destruction of a request (net::URLRequest).

- **"type":3** signifies EVENT_TYPE(HOST_RESOLVER_IMPLE_REQUEST)

The start/end of a host resolve (DNS) request. These events are logged for all DNS requests, though not all requests result in the creation of a HostResolvedImpl::Request object.
The BEGIN phase contains the following parameters:
{
"host": <Hostname associated with the request>,
"address_family": <The address family to restrict results to>,
"allow_cached_response": <Whether it is ok to return a result from the host cache>,
Possible values are "true" and "false"
"is_speculative": <Whether this request was started by the DNS prefetcher>
Possible values are "true" and "false"
}
If an error occurred, the END phase will contain these parameters:
{
"net_error": <The net error code integer for the failure>,
}
Acquired values from the log file for "net_error" are 2,118,109,804.

- **"type":4** signifies EVENT_TYPE(HOST_RESOLVER_IMPL_IPV6_REACHABILITY_CHECK)

This event is created (in a source of the same name) when the host resolver creates a UDP socket to check for global IPv6 connectivity.

It contains the following parameter:
{
"ipv6_available": <True if the probe indicates ipv6 connectivity>,
}
Possible values are "true" and "false".

- **"type":5** signifies EVENT_TYPE(HOST_RESOLVER_IMPL_CHACHE_HIT)

This event is logged when a request is handled by a cache entry.
It contains the following parameter:
{
"address_list": <The resolved addresses>,
}

- **"type":9** signifies EVENT_TYPE(HOST_RESOLVER_IMPL_JOB_EVICTED)

This event is created when a HostResolverImpl::Job is evicted from PriorityDispatch before it can start, because the limit on number of queued jobs was reached .

- **"type":34** signifies EVENT_TYPE(SOCKET_ALIVE)

Marks the begin/end of a socket (TCP/SOCKS/SSL/UDP/"SpdyProxyClientSocket").

The BEGIN phase contains the following parameters:
{
"source_dependency": <Source identifier for the controlling entity>,
}
Example: "source_dependency":{"id":299685,"type":6}

- **"type":35** signifies EVENT_TYPE(TCP_CONNECT)

The start/end of a TCP connect(). This corresponds with a call to

TCPClientSocket::Connect().

The START event contains these parameters:
{
"address_list": <List of network address strings>,
}

And the END event will contain the following parameters:
{
"net_error": <Net integer error code, on error>,
"source_address": <Local source address of the connection, on success>,
}

- **“type”:48** signifies EVENT_TYPE(SSL_CONNECT)

The start/end of an SSL "connect" (aka client handshake).

The following parameters are attached to the END event:
{
"version": <String name of the TLS version negotiated>,
"cipher_suite": <Integer code for the cipher suite>,
"is_resumed": <Whether we resumed a session>,
Possible values are “true” and “false”
"next_proto": <The next protocol negotiated via ALPN>,
}

EXAMPLE:
{"cipher_suite":49199,"is_resumed":false,"next_proto":"unknown","version":"TLS 1.2"}

- **“type”:61** signifies EVENT_TYPE(SSL_HANDSHAKE_MESSAGE_SENT)

An SSL connection sent or received a handshake message.

The following parameters are attached:
{

"type": <The type of the handshake message, as an integer>
"bytes": <The exact bytes sent, Base64 encoded.May be elided in some cases>
}

EXAMPLE:
{"bytes":"CAAACwAJABAABQADAmgy","type":8}

- **“type”:70** signifies EVENT_TYPE(SSL_CERTIFICATES_RECEIVED)

Certificates were received from the SSL server (during a handshake or renegotiation).

The following parameters are attached to the event:
{
"certificates": <A list of PEM encoded certificates in the order that they were sent by the server>,
}

- **“type”:71** signifies EVENT_TYPE(SIGNED_CERTIFICATE_TIMESTAMPS_RECEIVED)

Signed Certificate Timestamps were received from the server.

The following parameters are attached to the event:

{
"embedded_scts": Base64-encoded SignedCertificateTimestampList,
"scts_from_ocsp_response":Base64-encoded SignedCertificateTimestampList,
"scts_from_tls_extension": Base64-encoded SignedCertificateTimestampList,
}

The SignedCertificateTimestampList is defined in RFC6962 and is exactly as received from the server.

EXAMPLE:

{"embedded_scts":"",
"scts_from_ocsp_response":"",
"scts_from_tls_extension":"APAAdgCkuQmQtBhYFIe7E6LMZ3A
KPDWYBPkb37jjd80OyA3cEAAAAWk4tw0iAAAEAwBHMEU
CIQDg4t7nf8YWsy+VGawnvrwlWFUK5eGbpO1G7jXc2gVHdAI
gEa44gF+eQT9IAWO1j0wA0HzgBomT3B52TujsLjk8aWIAdgBV
gdTCFpA2AUrqC5tXPFPwwOQ4eHAlCBcvo6odBxPTDAAAAW
k4tw1kAAAEAwBHMEUCIHVaCXhkQuDrVSaI53VBKlmk5AG
vzP47m5XAgntJcPY9AiEA0jFA8Tzhf69ZqDAJVpkgBOaFtYoyJJ
dQ4Gq2izwLdbc="}

- **“type”:138** signifies EVENT_TYPE(HTTP2_HEADERS_SEND_HEADERS)

This event is sent for sending an HTTP/2 HEADERS frame.

The following parameters are attached:
{
"headers": <The list of header:value pairs>,
"fin": <True if this is the final data set by the peer on this stream>,
"stream_id": <The stream id>,
"has_priority": <True if the PRIORITY flag is set>,
"parent_stream_id": <Optional; the stream id of the parent stream>,
"priority": <Optional; the priority value of the stream>,
"exclusive": <Optional; true if the exclusive bit is set>.
}

EXAMPLE:
{"params":
{"exclusive":true,
"fin":true,
"has_priority":true,
"headers":
[":method: GET",":authority: cdn-images-1.medium.com",":scheme:
https",":path: /freeze/max/30/1*rXBBvMaeRp0wM-mS-
8A9vw.png?q=20","origin: https://towardsdatascience.com","user-agent:
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36","accept:
image/webp,image/apng,image/*,*/*;q=0.8","referer:
https://towardsdatascience.com/data-exploration-and-visualization-with-

r-ggplot-7f33c10ec1c","accept-encoding: gzip, deflate, br","accept-language: en-US,en;q=0.9"],
"parent_stream_id":25,
"source_dependency":
{
"id":299595,
"type":15
},
"stream_id":27,
"weight":147},
"phase":0,
"source":
{"id":299575,"type":9},
"time":"335808510",
"type":183}

N.B:
The log record for downloading a file from the internet is as follows:

{"params":
{"headers":
["HTTP/1.1 200 OK","Date: Sun, 19 May 2019 18:26:09 GMT","Server: Apache/2.4.25 (Debian)","Set-Cookie: [44 bytes were stripped]","Expires: 0","Cache-Control: must-revalidate, post-check=0, pre-check=0","Pragma: public","Content-Description: File Transfer","Content-Disposition: attachment; filename=\"The Chainsmokers - Roses ft. ROZES (Official Music Video).mp4\"","Content-Transfer-Encoding: binary","Content-Length: 10367619","Connection: Close","Content-Type: application/octet-stream"]},
"phase":0,
"source":{"id":17338,"type":1},"time":"22392091",
"type":169
}

This event is specified as EVENT_TYPE(HTTP_TRANSACTION_READ_RESPONSE_HEADERS) for “type”=169.

This event is sent on receipt of the HTTP response headers.

The following parameters are attached:

```
{
    "headers": <The list of header:value pairs>,
}
```

The header tag describes the time of download, server used, content type(html, plain_text, javascript, video, application, filename, cache control tec.).

Chapter 4:

4. Web Browser Log Analysis

4.1 Log Analysis

Log analysis is the process of analysing computer generated records and making important inferences and conclusions for mitigating different security risks and stopping unauthorized and unethical activities in the network. Log analysis also helps in reducing problem diagnosis and resolution time and in effective management of applications and infrastructure.

Now, analysing Web Browser log is the process of understanding what kind of activities a user gets involved in during browsing time. As we have seen in the previous chapter, the structure and type of browser log records the user activity. It might involve downloading content in different formats or simple browsing of plain text files or logging into an account and filling up a form for example.

The whole process may seem less risky and harmless at a glance but there can be numerous risks and unethical activities in a network. Tracking these activities and mitigating risks is essential for different institutions and organizations, be it educational or corporate.

4.2 Purpose of Web Log Analyser

A log analyser can handle the logs stored in the database and run necessary queries which can retrieve and reveal different user activities. It

can also store log in real time is given proper features and can run analysis on it in real time also.

Continuous and prolonged analysis and necessary visualisations can be extremely important in creating an activity profile of any particular user or a machine. Also it can help in understanding preferred activities and preferred websites of user such as downloading image, video etc. or spending more time in social media. But the real problem lies in finding out whether any content of undefined type is being downloaded or any security problem is happening which can jeopardize the data stored in the machine or the whole system for that matter.

The log files used in log analysis are generally provided by operating systems, applications, network equipment or similar devices. Logs are usually stored in a storage unit such as a hard drive or to an application such as a log collector.

For data driven analysis and decision making, many log analysis tools are available. Those are properly equipped with visualisation tools and other necessary statistical features.

For example,

- Loggly.
- Logentries.
- GoAccess.
- logz.io.
- Graylog.
- Splunk.
- Logmatic.io.

And many more.

The objective of this project is to develop a log analyser for the log files of a web browser with necessary features than can be useful for the tasks needed to be done.

4.3 Collected Log

The log files collected for the purpose of this project is for the web browser Google Chrome. The process of procuring the log files of Chrome is mentioned in the previous chapter.

The log files are in JSON(Java Script Object Notation) format. JSON data is written as name/value pairs. JSON objects are written inside curly braces.

For Example:

{"Name":"Rajdipta", "Surname”: “Barman"}

JSON arrays are written inside square brackets.

For example:

"employees":

```
[
   {"firstName":"raj", "lastName":"barman"},
   {"firstName":"ria", "lastName":"paul"},
   {"firstName":"deb", "lastName":"barman"}
]
```

The collected logs are stored in similar manner. Different event files are created for different tabs that were opened. So several files should be stored and analysed. But first the acquired records are converted from JSON format to CSV (Comma Separated Value) using python language to better understand the structure and the information stored in the log file.

4.4 Analysis

Different kinds of analyses can be done on the data by taking each taking each tag and useful information can be procured.

4.4.1 Finding the number of secure and unsecure websites visited.

The first part of URL has ‘http’ or ‘https’ which signifies whether a website is secure on or not. The protocol ‘https’ states that the website is secure.

The ‘url’ tag stores the complete URL of the web page visited. This tag can be parsed to using regular expression to check whether the fifth character of the URL has ‘s’ or not. If it has ‘:’ then the protocol used for the website is ‘http’, if not, there can only be ‘s’ in the fifth position. This will signify that the protocol used for the website is ‘https’ i.e the website is secure.

The result of this analysis shows the number of secure and unsecure websites visited which can, in turn, mitigate some security problems for the machine.

For example:

"url": “http://jaduniv.edu.in/”

The above mentioned URL has been found in the log files of Google Chrome web browser. This clearly shows that the website “jaduniv.edu.in” is not using a secure protocol.

Algorithm:

Step 1: Taking a single entry from ‘url’ column from the database.

Step 2: Parsing the ‘url’ with regular expression.

Step 3: If the fifth character is ‘s’ increase the number of secured websites. If “:” is found increase the number the of unsecured websites.

Step 4: If next entry exists then go to step 2. If next entry doesn't exist go to step 6.

Step 5: Show the number of secure and unsecure websites visited.

Step 6: Stop.

4.4.2 Finding the frequency of 'GET' and 'POST' methods.

HTTP POST requests supply additional data from the client (browser) to the server in the message body. In contrast, GET requests include all required data in the URL.

In the 'params' dictionary of a particular log record, 'method' is a tag which can have values 'GET' and "POST'. If the value is 'POST', then the record signifies that large amount of data or confidential information has been sent during the course of browsing activity. Corresponding URL can also be found to which can show for websites GET or POST methods were used.

All the tags are separately stored in the database in a tabular manner. A simple SQL query would be sufficient to retrieve the data.

For example:

{"params":

{

"load_flags":832,

"method":"POST",

"privacy_mode":1,

"upload_id":"0",

"url":https://clientservices.googleapis.com/uma/v2

}

….

The aforementioned log record has five tags two of which are "method": "POST" and "url":https://clientservices.googleapis.com/uma/v2. This information will be shown in the analyser window when asked for.

Algorithm:

Step 1: Taking a single entry from ‘method’ column from the database.

Step 2: If the value of the tag is ‘POST’ increase the value of the variable which stores the number of ‘POST’ method. Go to step 4. Otherwise go to Step 3.

Step 3: If the value of the tag is ‘GET’ increase the value of the variable which stores the number of ‘GET’ method. Go to step 5.

Step 4: If flag indicates that the last entry was POST then add the value of the ‘url’ tag to a separate array. The URLs stored in this array will be shown with number of POST methods. Go to step 6.

Step 5: If flag indicates that the last entry was GET then add the value of the ‘url’ tag to a separate array. The URLs stored in this array will be shown with number of GET methods.

Step 6: Stop.

4.4.3 Names and content type of downloaded files

During the course of browsing different kinds of files can be downloaded. Be it audio, video, image etc. The log data distinctly shows the names of those files and the type of content stored in that.

Procuring this data can keep track of users’ activity, especially in educational institutions and corporate offices, where maintaining discipline and decorum is an essential aspect. Unethical activities can jeopardize the ethics of an institution.

To analyse the log and get this data, an example has been shown:

{"params":

{"headers":

["HTTP/1.1 200 OK",

"Cache-Control: private",

"Content-Type: image/jpeg; charset=utf-8",

"Server: Microsoft-IIS/8.0",

"Content-Disposition: attachment; filename=Mount_Elbrus_Mountain_Volcano_in_Russia_4K_Wallpaper.jpg"

…..

…..

The ‘params’ dictionary in this data has another dictionary in it called ‘headers’. This dictionary has different tags one of which is "Content-Type:” This tag shows the content type i.e ‘image’ and the extension used for the file.

The next tag is “Content-Desposition” which has the ‘filename’=<name of the file downloaded with extension>

As all the information is stored in the ‘header’ tag as a list, the whole data has been uploaded in the database in a single entry. Firstly the names of the tags are parsed i.e. ‘Content-Type’ and ‘filemame’. For ‘Content-Type’, value has been taken until a semicolon (;) has been encountered. For ‘filename’, data is taken until an inverted comma (“) is encountered.

4.5 Proposed Analyses For Future work

The collected log files consist of almost 60 types of records corresponding to different event types. Each tag and their values depict important aspects of activities done on a web browser. On detailed and efficient analysis, a number of user activity related predictions and security issues can be detected.

Proposed analyses are given below:

- Number of downloading events within specified time duration can be detected.
- A list of PEM encoded certificates in the order that they were sent by the server can be parsed from the log file.
- With ‘host’ tag, hostnames associated with a HTTP request can be found.
- By parsing the URLs visited by the users, user preference can be predicted.
- If any content of undefined type is accessed and downloaded, it can put system security in danger. Any occurrence of this sort can be detected.

Chapter 5:

5. Implementation

5.1 Introduction

Manual analysis of browser log can be a tedious and extremely time consuming task. The sheer size and amount of data can get too difficult to handle. A web browser log analyser may serve the purpose of making these tasks easier, unambiguous and far more efficient.

Any irregularity, unauthorized activity or users' preferred tasks can be found using this analyser. Name of the downloaded files, the websites from which those are being downloaded, whether any content of unknown type is being downloaded which raise security concerns can be found. Visualizations can also be implemented after prolonged logging and analysis of logs.

For the objective of this project, browser log files of Google Chrome have been used. Log files of other browsers can be used also with necessary adjustments.

These tabs are shown in the analyser which serves different purposes:

- Upload
- View
- Analyse

5.2 Features of Analyser

Following features have been added to the tool:

5.2.1 Collection and Uploading the Data

The process of acquiring the log files of Google Chrome has been described earlier. The log files are stored in JSON format. So, those are

converted into CSV files first. This allows the developer to understand the data in a structured manner. The objective of this tab would be to upload the separate log files in the database. For that purpose the log files need to be in tab delimited text files. A python program is executed to store the log file in the database on clicking the upload button of the analyser. Other log files can be subsequently uploaded by browsing from the analyser window. The files are uploaded in MySQL database. On successful upload, a message will show “File uploaded successfully.”

A drop-down menu can also be added to specify the browser whose log files would be uploaded.

5.2.2 View

“View” button is added to analyser to view the uploaded the log files from the analyser window. After successfully uploading the files, queries can be made to view any particular uploaded log file in a tabular manner. This allows viewing of the structured log files from the analyser window and not exiting from it to check the structure or data collected in the log file. Also, to analyse, the user needs to know the column name of the tables so that clear queries can be made.

Writing queries can be avoided if a drop-down menu is added to the window with the currently existing log file names. A particular file can be selected and it will be shown in the analyser window.

5.2.3 Analyse

This is similar to the “View” feature, only difference is; the desired records can be retrieved from the database different queries.

MySQL database is being used in this project for storing the data. Different SQL queries can be incorporated in the python program which is also being executed for fetching the data. For example, list of downloaded files with its content type can be searched and shown, or a particular type of event can be searched which is in correspondence to a

“TCP Connect” job; whether the method was ‘GET’ or ‘POST’ and frequency of occurrences of both the methods can retrieved. Fetched data can also be stored and used for the purpose of visualisation.

5.2.4 Visualisation

Visualisation is an extremely essential part of data analysis and subsequent decision making. Graphical representation of analysed data can better depict the information and inferences can be drawn rather easily; also it can save time in decision making for a particular kind of dataset whose analysis and result can be difficult to understand as well.

On this analyser, ‘Visualization’ button be used to draw bar charts of time versus and kind of record; be it ‘no of files downloaded files’, ‘no of visits’ for particular websites which can procured with parsed URLs and timestamps.

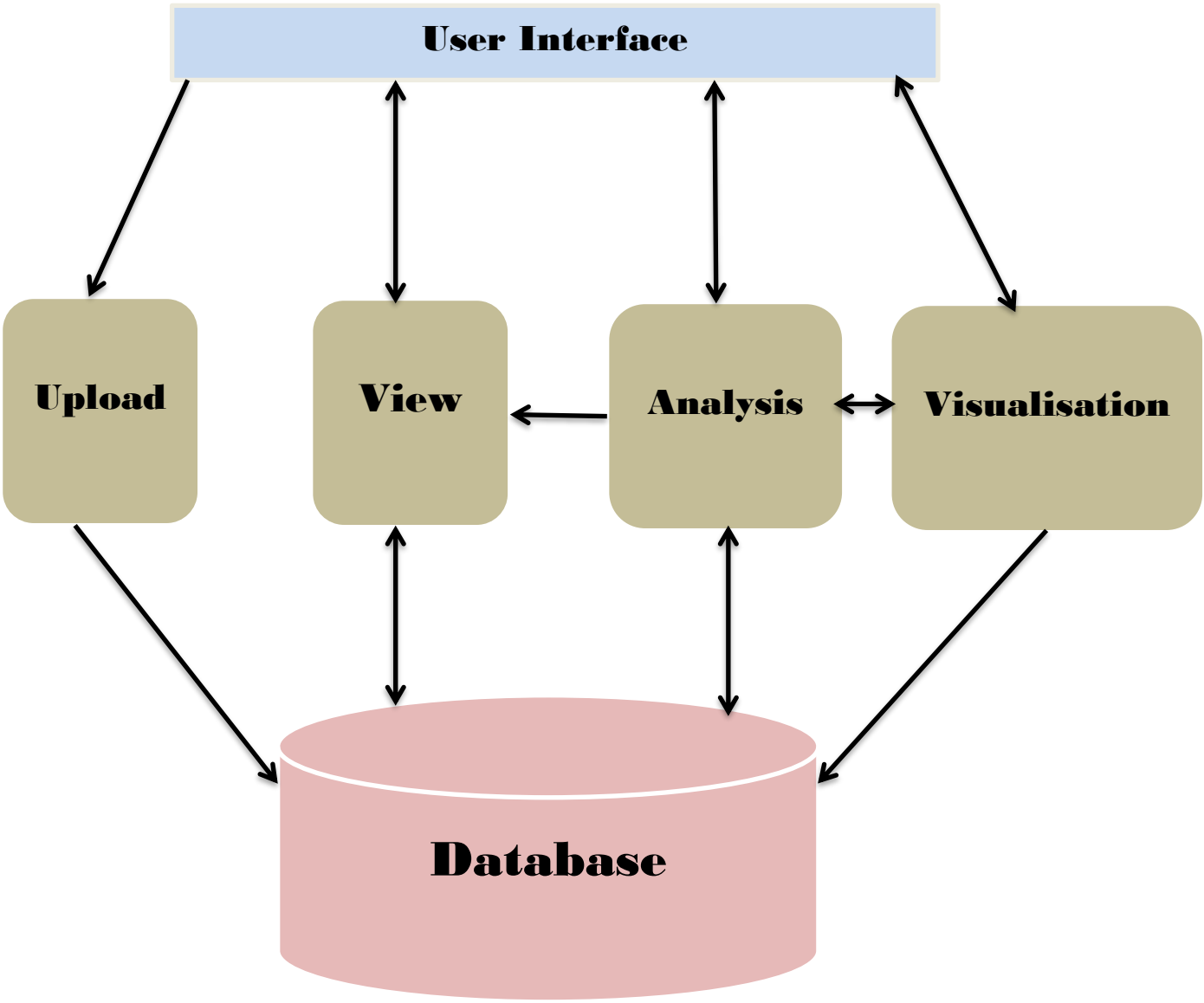
It can also show pie charts which would better visualize a certain comparison. Type of comparisons, although, would be predefined and shown as options in a drop-down menu present in the visualisation tab.

5.3 Architecture of the Analyser

This tool has a simple architecture which consists of a UI end which connects to 4 other modules namely ‘Data upload’, ‘Analysis’, ‘View’ and ‘ Visualisation’. These modules can be inter-connected as well for any task. For example any queries made in Analysis module will show its results to the user through the View module.

Below these modules lies the database which is probably the most essential part of the tool. Every query, result, report or visualisation would be done with the data stored in there.

A schematic diagram has been made to show the architecture of the tool in the next page.



5.4 Technologies Used

For the objective of this project following softwares were used:

- Python 3.7 was used for programming tasks. Python's UI development tools were helpful for this purpose. Other packages related to database were put into use as well.
- MySQL Workbench 8.0 has been used to for storing the data.

5.5 Future Work

In this project, web browser log files of only one web browser have been used. The process of procuring log files of other web browsers has been

mentioned in previous chapters. Modifications can made on the log analyser to put log files from different browser and analyse the same in a singular and well defined way.

Procuring log files from one machine and analysing them won't be of too much help. Also, analysing log files of each machine separately may be time consuming. Obtaining and assimilating log files from all the machines of an organization will amount to humongous size of data. This is where the concept of big data comes into practice.

Different aspects of Big Data analysis can enable the system to analyse the huge data acquired from all the machines of the organization and may even do so it real time. The data processing method of MapReduce will be extremely fruitful for the purpose of efficient parsing and analysis.

The MapReduce algorithm contains two important tasks, namely Map and Reduce. The Map serves the purpose of processing the input data. Generally the input data is in the form of file or directory(in JSON file format for Google Chrome) and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data in the form of key-value pairs. This can a very useful way of sorting out and storing the data with name of the tags as their "key". Processing time can be reduced by a significant factor using it.The job of the Reduce stage is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

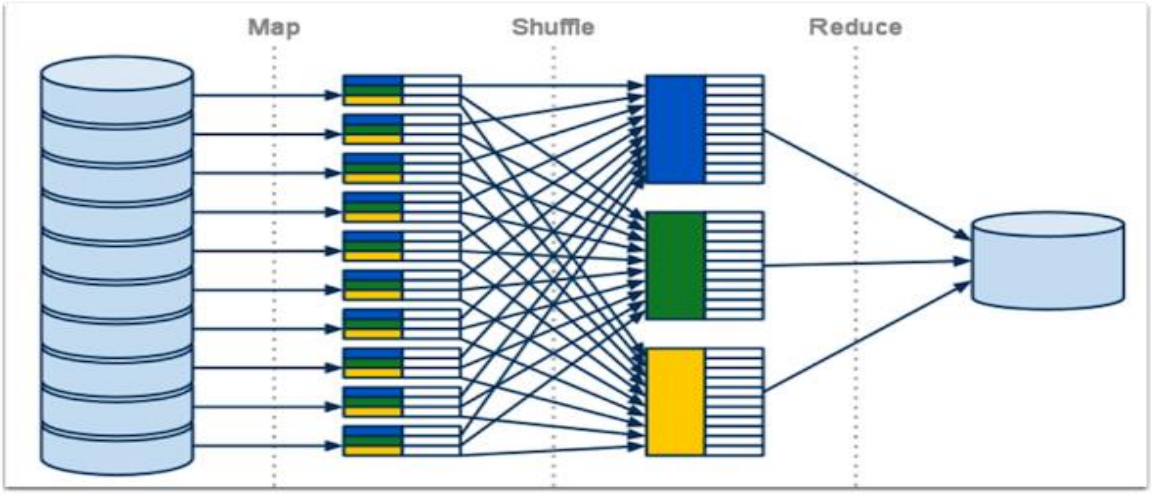


Fig 1: A Schematic diagram of the MapReduce process

Acquiring data from all possible sources and storing it centrally is a challenge. Apache Hadoop can be helpful towards solving this problem. It is an open-source software that provides usage of a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model.

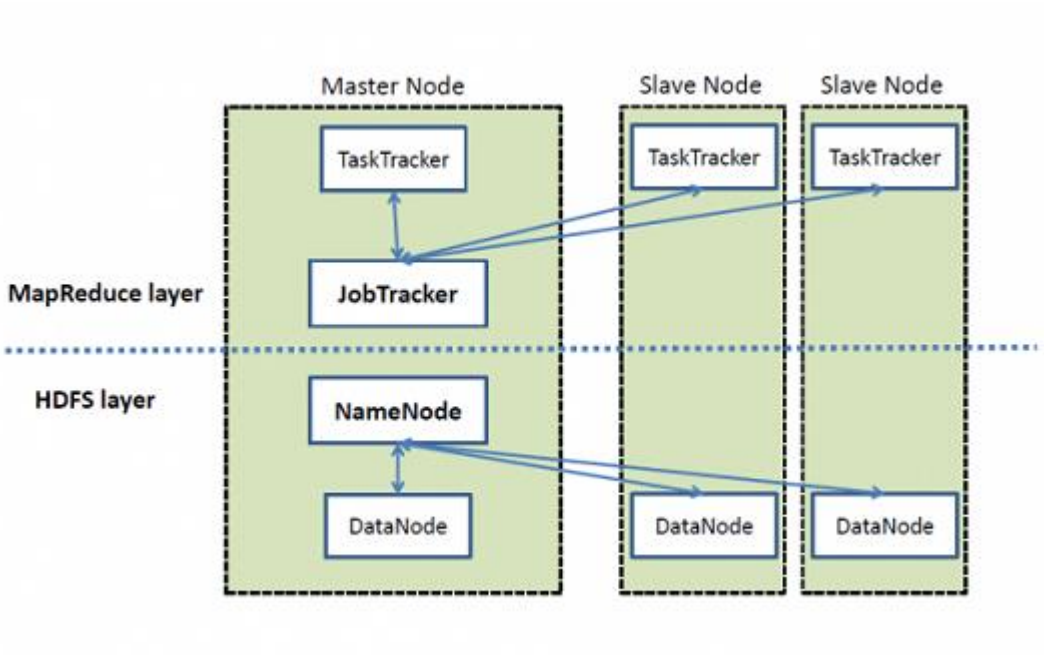


Fig 2: A high level architecture of Hadoop

5.6 Conclusion

On completion of this project several aspects of web browser log files have come into notice. Different kinds of data that get stored into log files and their implications can be used to implement many tasks. Types of events that take place and their significance, priorities of http requests and number and point of occurrence of GET and POST methods are easily available in the log files.

Downloading activities can also be properly monitored with necessary adjustments in the tool. Whenever any content is being downloaded alert can be generated centrally in an organization. Undefined content can be stopped from getting downloaded. A web log analyser can be useful towards finding users' inclination towards any content in the internet.

Other useful information such as security concerns and system's integrity can be checked using this tool. Proper modifications which can enhance this analyser's functionality need to be made and it would be appreciated.

References:

Chapter1:

1. https://en.ryte.com/wiki/Log_File_Analysis

Chapter 2:

1. https://www.techopedia.com/definition/31756/log-analysis
2. https://www.webopedia.com/TERM/L/log_file.html
3. https://en.wikipedia.org/wiki/Log_analysis
4. https://html5rocks.com

Chapter 3:

1. The Chromium Project –" https://www.chromium.org"
2. https://w3schools.com

Chapter 5:

1. Figure 1: http://www.dummies.com

2. Figure 2: https://www.dezyre.com

3. https://www.tutorialspoint.com