High Speed Streaming Data Analysis of Web Generated Log Streams

Sonali Agarwal
Indian Institute of Information Technology Allahabad
Uttar Pradesh, India
sonali@iiita.ac.in

Bakshi Rohit Prasad
Indian Institute of Information Technology Allahabad
Uttar Pradesh, India
rs151@iiita.ac.in

***Abstract*— Web logs provide useful insight of large scale web based applications and helpful in deriving web usage patterns. Since, web usage patterns are available at a high rate and a high volume and also continuously updating in a real time environment, must be handled through modern big data architectures supported by powerful real time big data processing tools. Web generated log streams have most significant impact when it is feasible to analyze them at a time when they are emitted. In proposed research work, an advanced stream analytics framework especially for web generated log streams has been proposed by using the dataset of web access logs representing HTTP requests received by NASA Kennedy Space Center Server. The proposed framework can resourcefully handle the challenging issues associated to manage multiple web based log streams that are distributed across a fleet of web based applications and present a summarized view of statistical profile of web based applications which may be useful for web usage mining.**

Keywords—Web Logs, Access Logs, Web generated log streams, Apache Spark, Stream Analytics

I. INTRODUCTION

A web generated log stream is a real-time, continuous, ordered sequence of status information of web usage patterns. The ordering of log streams may depend upon implicitly by arrival time or explicitly by timestamp. The analysis of web log streams is an essential module of recent web based applications because it helps to explore the way to faster diagnostics for proactive alerts and comprehensive actionable reporting of any system. It can also facilitate quality and performance monitoring of critical streaming assets [1] [2].

The objective of present research work is to develop a real time in memory advanced stream analytics framework especially for web generated log streams. Web logs are mainly divided as error logs and access logs [3]. In this paper the analysis of access log has been focused to understand the web server records of all the visitors. The web log analyzers are either serving as a static web log analyzer or dynamic web log analyzer [4].

TABLE I. STATISTICAL INFORMATION RELATED TO STATIC WEB LOG ANALYSIS

Parameters	Details
User name	User identification using IP address provided by Internet Service provider.
Visiting Path	The path taken by the user while visiting the site.
Path Traversed	The details of various link visited by the user.
Time stamp	The time spent by the user in each web page which is also known as session.
Page last visited	Last visited page before the session closes.
Success rate	Number of downloads/access/ copying from a web page/site.
User Agent	Browser that sends the user request to the web server.
URL	An HTML page, a CGI program, or a script accessed by a user.
Request type	Method used for information transfer such as GET and POST.

Dynamic or Streaming web log analysis is different from static web log analysis because in this case it is not possible to control the arrival order of data items and since they are continuously arriving cannot be completely stored locally. Table 2 shows additional characteristics of web generated log streams which are crucial to control and manage [6].

TABLE II. ADDITIONAL CHARACTERISTICS OF WEB GENERATED LOG STREAMS FOR DYNAMIC WEB LOG ANALYSIS

Characteristics	Description
Guaranteed Availability	It is the ability to successfully view a stream which is a challenging task due to connection failures to the server, dropped connection and time delay involved in buffering.
Startup Time	Time taken to record a log. It is affected by DNS lookup time, connect time and initial buffer time.
Effective Bitrate	It is a stream quality parameter affected by packet loss and jitter.
Re-buffer Ratio	It is ratio of time spent in re-buffering as compared to the total time required to create a stream.
Window Size	It is a specific time duration of observing a particular stream.
Sliding interval	Time interval required to slide/move the window to observe next sequence of stream.

It is clear from table 1 and table 2 that web log files hold various unformatted and unstructured information. Scanning large scale unstructured log files on a regular basis for large-scale systems cannot be handled by conventional processing tools. Therefore using Apache Spark with its capability to handle in memory computations is helpful to improve quick readability and easy integration within the system having sheer

volume of unstructured log messages. Apache Spark is an open source, big data processing tool supporting Scala, Java, Python and R languages. Since web log streams are continuously generated and holding the details of relevant events including all set of instances, it is quite challenging to collect, store and analyze massive web logs as they come in real time. Another challenge of high speed streaming data analysis of web generated log streams are mainly due to the unstructured format of log files which do not store user details rather it stores records of IP Addresses therefore user oriented queries cannot be answered easily. Due to similar reason, user profiling became tedious process and requires additional predictive analysis [7] [8]. Through proposed framework, it is aimed to develop an efficient streaming web log analyzer which could be powered by Spark, an in memory computation based big data processing tool, and capable to derive a web usage profile in real time manner.

The paper is organized into 5 sections. In section 2, the related research work based on web based log analytics has been discussed. An advanced stream analytics framework especially for web generated log stream has been proposed in section 3. Section 4 provides the details of web access logs representing HTTP requests received by NASA Kennedy Space Center Server located in Florida. The derived statistical information from the proposed model could be efficiently visualized through a Dashboard, is also presented in section 4. Lastly, concluding remarks and possible future perspective of proposed research work is significantly indicated in section 5.

II. RELATED WORK

Log analyzers are widely used in many web based applications and there are several research work available on traditional static log analyzers. Few recently developed log analyzers are also became popular and adopted to overcome the drawbacks of static log analyzer[9][10]. A research work performed by Joshila et.al. provides a detailed discussion about static web log analysis, log types, log formats, access mechanism and their usefulness for web mining. The authors of the paper also discussed about creating an extended log file for user profiling [11]. Lalit Agarwal proposed a hybrid web log analyzer using both IIS and tomcat server log. They proposed an ontology mapping approach to get required information. Data mining technique was also used to match user profiles and their interest [12].

Park et. al, in their paper utilized a Korean web search engine named as NAVER for transaction logs. They proposed innovative methods of log cleaning, log sessions and classification of logs. The output of their research work is to analyze user queries in different ways for the better performance of search engines [13]. A web log analysis system for E Governance website was proposed by Wang et. al. The authors proposed a synthetic index which is a combination of subjective indexes and objective indexes. These indexes are developed by a citizen centric survey. It is helpful to evaluate the performance of E-Governance Websites [14].

Recent research work has been carried out by Madhury Mohandas on Hadoop Log Analysis Tools. In the paper the need of activity monitoring in large scale computing has been highlighted. Several Hadoop based log analyzers have been discussed to perform assessment of failure detection and monitoring [15]. Since log analysis supports optimization of overall system performance and resource utilization, following are the recent developments have been reported in the area related to log analysis and activity monitoring[1][2].

TABLE III. LOG ANALYZER TOOLS AND MODELS

Tools	Description
Vaidya	It works on Hadoop cluster and identifies the poor performance jobs present within Hadoop cluster [16][17].
Chukwa	It serves as a network management and data collection tool and supports big data analytics [18].
Salsa and Mochi	Salsa monitors Data Node logs and Task Tracker logs to visualize the execution of data and control flow for the purpose of fault detection and diagnosis [19]. Mochi establishes the impact of time, space and volume on any executing task and also integrates their interdependencies in the distributed environment [20].
Weblog Parse	Developed by ACME Labs and used to extract specific fields from a web log file [21].
Weblog	It is developed by Darryl C and serves as a log files analysis tool to keep track of activities of website on daily, weekly and monthly basis [22]
Analog	It is developed by University of Cambridge and keeps track of user visits of any website [23].

All the existing solutions limited to static log analysis and designed for conventional websites having relatively low traffic. Recent developments in the field of big data needs live log analysis tools which can handle big data processing in real time manner.

III. PROPOSED FRAMEWORK

The proposed framework for high speed analysis of web access log streams is shown in figure 1. It is based on integration of Kafka and Spark streaming module. Apache Spark processing utilizes RDDs (Resilient Distributed Data) which are distributed collection of data and considered as a key abstraction provided by the Apache Spark [24]. As shown in figure 1, the proposed framework has various components assembled in way to facilitate the stream analytics task on sequence of high speed streams of web access logs. The streams of web logs are simulated from the web access log file to work as live input log streams from server. Other components of proposed framework are described as below:

A. Kafka Server

Apache Kafka provides a distributed server for message queuing. It works on the concept of publish-subscribe mechanism. It is design to facilitate a single cluster that works as a data backbone for any organization. Streams of data are distributed over various machines in this cluster to provide large scalability for storing and consuming data. Due to replicated and persistent storage, it curbs up the risk of data loss [25].

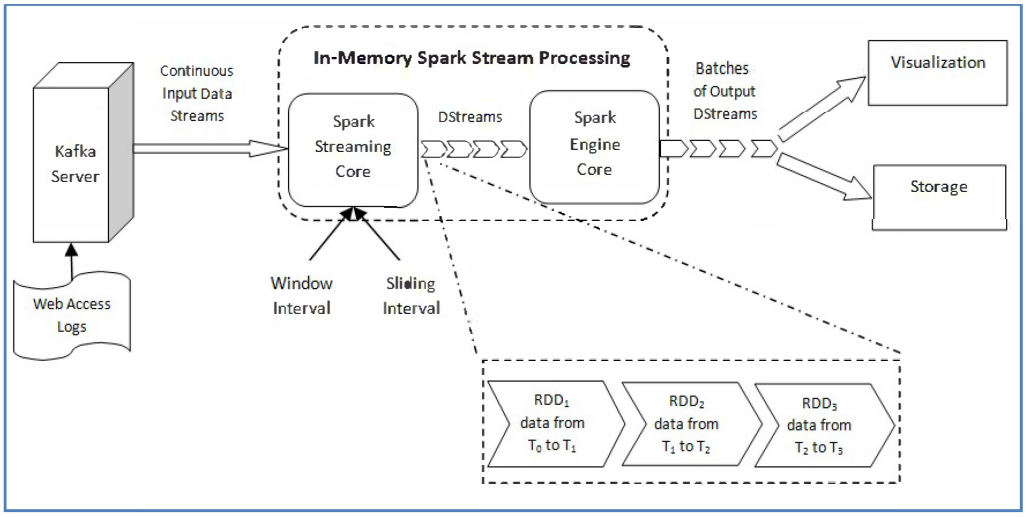

Fig. 1. Advanced stream analytics framework for web generated log stream

B. *In-Memory Spark Stream Processing*

Spark achieves high speed processing power due to its in-memory computation technique unlike Hadoop and similar kind of systems. Therefore, spark is well suited for real time stream processing for web access logs that are being generated continuously at high speed. This component of the proposed framework has two important sub-components [26]:

1) Spark Streaming Core: It provides core APIs required to setup the streaming context to effectively process the stream of logs (or any data streams). Also, it supports various high level functionalities like 'map' and 'reduce' functions to fasten the development of applications. The spark streaming basically converts the continuous input log streams into Discrete Streams (DStreams). Internally, the DStreams are also processed as RDDs. While initializing the Streaming Context, spark streaming takes usually two inputs; Window Interval and Sliding Interval. The Window interval specifies over what duration, the streams are to be focused currently. On the other hand, Sliding Interval specifies, at what interval the observation widow must be moved further. In Scala language, we can set these two intervals as below:

>scala val WINDOW_LENGTH = new Duration(100)
>scala val SLIDE_INTERVAL = new Duration(20)

It implies that at a particular moment the streams over the recent past 100 ms will be observed for processing (WINDOW_INTERVAL) and after each 20 ms the observation window will move forward.

2) Spark Engine Core: It provides core functionalities of Apache spark that are responsible for setting up the spark computing environment and takes care of generating, transforming and distribution of RDDs [27]. Also, Spark Driver in Spark Engine manages internally, the distribution of task on various machines in the spark cluster, collecting and aggregating the response, etc. as depicted in figure 2.

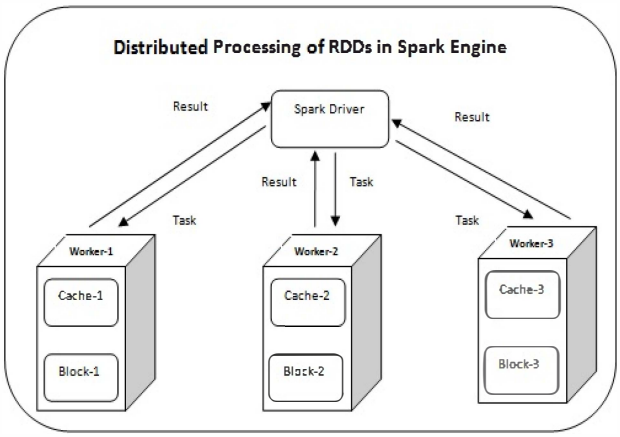

Fig. 2. Distributed Processing of Spark Engine Core

C. *Result Visualization Dashboard*

After processing the data, the results are produced as the batches of output streams which may be directed towards permanents storage such as file systems like HDFS (Hadoop Distributed File Systems) or may be stored in some Database or simply may be continuously visualized and monitor on a simple HTML page as an online dashboard.

IV. Data Set Description and Experimental Analysis

The dataset used in our experimental analysis consists of sequence of web access logs representing HTTP requests received by NASA Kennedy Space Center Server located in Florida [28]. Each log entry represents one access attempt and contains following fields separated by a space:

- *Host:* It specifies the name or IP address of the host which has made the request.
- *Timestamp:* This field contains Date and Time information in "DD/MON/YYYY: HH:MM:SS - ZZZZ" format where DD signifies the day of a month (ranging from '01' up to a maximum of '31'), MON represents the month in a year (JAN, FEB, etc.) and YYYY specifies the respective year. HH, MM and SS represent hours, minutes and seconds respectively in 24 hour clock format whereas ZZZZ signify time zone.
- *Request String:* It is the access request string mentioned in quotes.
- *HTTP Response Code:* This field shows the HTTP response code.
- *Content Size:* This column records the size of reply in bytes.

A sample of five log entries in the log file is shown in figure3.

```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 2004179
```

Fig. 3. Snapshot of NASA Web Access Log file

An experimental analysis has been performed on NASA log file dataset and following parameters have been calculated to establish the web access profile:

A. Content Size

It includes the measurement of minimum, maximum and average content size of the reply corresponding to various request made to the server over a specified interval. The output snapshot of the result is shown in figure 4.

```
15/07/10 10:53:20 INFO SparkContext: Starting job: foreachRDD at
15/07/10 10:53:20 INFO DAGScheduler: Job 31 finished: foreachRDD
Content Size Avg: 3483, Min: 0, Max: 14897
15/07/10 10:53:20 INFO CheckpointWriter: Saving checkpoint for t
15/07/10 10:53:20 INFO JobScheduler: Finished job streaming job
```

Fig. 4. Snapshot of content size

B. Response Code Count

It computes the number of times a particular response is generated in a stipulated window of time as shown in figure 5. The response count of response status codes 200, 304 and 302 is 41, 10 and 1 respectively.

```
15/07/10 10:51:10 INFO DAGScheduler: ResultStage 23 (foreachRDD
15/07/10 10:51:10 INFO DAGScheduler: Job 16 finished: foreachRDD
Response code counts: [(200,41),(304,10),(302,1)]
15/07/10 10:51:10 INFO JobScheduler: Finished job streaming job
15/07/10 10:51:10 INFO JobScheduler: Starting job streaming job
```

Fig. 5. Status of Response Code Count

C. IP Addresses Hit Count

Counting the number of times a particular IP or host requesting the server is one of the crucial tasks in web log analysis and could be significantly useful in host centric recommendation systems and in analyzing and identifying potential security attackers. Figure 6 contains the hosts who have hit the server more than 3 times in a set time window.

```
15/07/10 10:57:31 INFO DAGScheduler: Job 10 finished: foreachRDD
treamingTotal.scala:94, took 0.035600 s
IPAddresses > 3 times: [uplherc.upl.com]
15/07/10 10:57:31 INFO JobScheduler: Finished job streaming job
.2 from job set of time 1436506050000 ms
```

Fig. 6. IP Addresses Hit Count

D. Broken Links Count

Counting broken links are extremely useful in identifying the URLs/URIs of server which are no more linked over the internet and thus are non-responding. This measurement has been easily done by counting 404 status codes generated by the server in response to its request. The command to count the total number of broken links in Scala language and its output is shown in figure 7 and figure 8 respectively. On the other hand, the command for listing those broken links and respective output is shown in figure 9 and figure 10.

```
srl@ubuntu: ~/spark-1.4.0-bin-hadoop2.4
scala>

scala> log.filter(line => getStatusCode(p.parseRecord(line)) == "404").count
```

Fig. 7. Scala Script for counting total number of broken links

```
15/07/09 16:11:28 INFO TaskSchedulerImpl: Removed TaskSet 0.0,
all completed, from pool
15/07/09 16:11:28 INFO DAGScheduler: Job 0 finished: count at
 3.023934 s
res0: Long = 10056
```

Fig. 8. Snapshot of counting total number of broken links

```
scala> log.filter(line => getStatusCode(p.parseRecord(line)) == "404").map(getRe
quest(_)).count
val recs = log.filter(line => getStatusCode(p.parseRecord(line)) == "404").map(g
etRequest(_))
val distinctRecs = log.filter(line => getStatusCod(p.parseRecord(line)) == "404
").map(getRequest(_)).distinct
distinctRecs.foreach(println)
```

Fig. 9. Scala Script for listing the broken links

```
srl@ubuntu: ~/spark-1.4.0-bin-hadoop2.4
Some(GET /shuttle/missions/sts-69.htm HTTP/1.0)
Some(GET /nba/ HTTP/1.0)
Some(GET /history/apollo-13/apoll-13.html HTTP/1.0)
Some(GET /shuttle/missionssts-70/woodpecker.html HTTP/1.0)
Some(GET /software/software.html HTTP/1.0)
Some(GET /shuttle/missions/sts-82/mission-sts-82.html HTTP/1.0)
Some(GET /procurement.htm HTTP/1.0)
Some(GET /software/winvn/release.html HTTP/1.0)
Some(GET /shuttle/images HTTP/1.0)
Some(GET /elv/elvpage.htm#VIDPICS HTTP/1.0)
Some(GET /classes/net/www/protocol/wais/Handler.class HTTP/1.0)
Some(GET /winvn.html HTTP/1.0)
Some(GET /it-www.hq.nasa.gov/EmailWhitePaper.html HTTP/1.0)
Some(GET /X500/x500.html HTTP/1.0)
Some(GET /ksc.shtml HTTP/1.0)
Some(GET /shuttle/missions/missions.htm HTTP/1.0)
Some(GET /procurement/expo95.htm HTTP/1.0)
Some(GET /ads/images/wsji_ad.gif HTTP/1.0)
Some(GET /shuttle/missions/sts-69/images/index HTTP/1.0)
Some(GET /shuttle/resources/orbiters/endeavour.html>)
```

Fig. 10. Snapshot of listing of broken links

E. List of Broken URIs

Following command as shown in figure 11 extracts the actual URIs out of broken request strings and its final result is shown in figure 12.

```
scala> val distinctRecs = log.filter(line => getStatusCode(p.parseRecord(line))
== "404")
                    .map(getRequest(_))
                    .collect { case Some(requestField) => requestField }
                    .map(extractUriFromRequest(_))
                    .distinct
distinctRecs.count
distinctRecs.foreach(println)
```

Fig. 11. Scala Script for extracting the URIs

```
sri@ubuntu: ~/spark-1.4.0-bin-hadoop2.4
/apollo/apollo.html
/shuttle/countdown/liftoff.htm
/shuttle/countdown/count.html#T-06H00M
/shuttle/missions/sts-51/sts-51-patch-.*
/pub/winwn/nt
/shuttle/status
/shuttle/technology/sts-newsref/sts-gncc.html
/suttle/missions/sts-69
/shuttle/missions/sts-71/movies.html
/images/new.gif
/hubble/
/history/apollo/-apollo-13/apollo-3.html
/software/techdoc/td-1.4.3.1.html
/stats
/history/html
/history/apollo/a-003/sounds/
/ASM_WebDir.html
/shuttle/missions/sts-69/images/list
/shuttle/missions/51-b/mission-sts-51-b.html
/shuttle/missions/sts-10/sts-10-patch.jpg
/history/apollo-13/images
```

Fig. 12. Snapshot of extracting the URIs

F. URI Hit Count

Lastly, a Scala script as shown in figure 13 is used to compute the URI Hit Count for each of the URI which specifies the number of times a URI is requested. A sample of some URIs and their hit count is shown in figure 14 and further this result is sorted in decreasing order of hit count, by using Scala script mentioned in figure 15. The sorted result, as depicted in figure16, shows the topmost hit URIs of the server. Thus, this analysis may be very effective in recommendation systems, web usage mining, actionable alert system and many other applications.

```
scala> val uriCounts = log.map(p.parseRecord(_).getOrElse(nullObject).request)
                    .map(_.split(" ")(1))
                    .map(uri => (uri, 1))
                    .reduceByKey((a, b) => a + b)

val uriToCount = uriCounts.collect
```

Fig. 13. Scala Script to compute the URIs

```
uriCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[35] at reduceBy
Key at <console>:38
uriToCount: Array[(String, Int)] = Array((/images/shuttle-patch-logo.gif,1), (/h
istory/apollo/images/apollo-logo.gif,1), (/shuttle/missions/sts-71/images/KSC-95
EC-0423.gif,1), (/shuttle/missions/sts-70/movies/movies.html,2), (/persons/nasa-
cm/mike-sm.gif,1), (/icons/text.xbm,1), (/htbin/cdt_main.pl,1), (/persons/nasa-c
m/jmd.gif,1), (/history/apollo/images/footprint-logo.gif,1), (/history/apollo/im
ages/apollo-small.gif,2), (/persons/nasa-cm/tnn-sm.gif,1), (/images/KSC-logosmal
l.gif,11), (/facilities/lc39a.html,2), (/history/apollo/apollo-16/apollo-16-patc
h-small.gif,1), (/images/rss.gif,2), (/images/lc39a-logo.gif,2), (/images/USA-lo
gosmall.gif,5), (/history/skylab/skylab.html,1), (/shuttl...
```

Fig. 14. A sample of some URIs and their hit count

```
scala> import scala.collection.immutable.ListMap
val uriHitCount = ListMap(ur Tocount.toSeq.sortWith(_._2 > _._2):_*)
```

Fig. 15. Scala script for Sorted URI Hit count

```
scala> uriHitCount.take(10).foreach(println)
(/images/KSC-logosmall.gif,11)
(/images/NASA-logosmall.gif,11)
(/images/ksclogo-medium.gif,7)
(/,7)
(/history/apollo/images/apollo-logo1.gif,7)
(/images/USA-logosmall.gif,5)
(/images/WORLD-logosmall.gif,5)
(/shuttle/countdown/,5)
(/images/MOSAIC-logosmall.gif,5)
(/images/launch-logo.gif,4)
```

Fig. 16. Snapshot showing the topmost hit URIs of the server

The output of all Scala script can be further displayed through a html based user interface which can serve as a Dashboard for high speed streaming data analysis of web generated log streams. Figures 17 and 18 gives an insight view of the Dashboard generated as an output of the proposed framework.

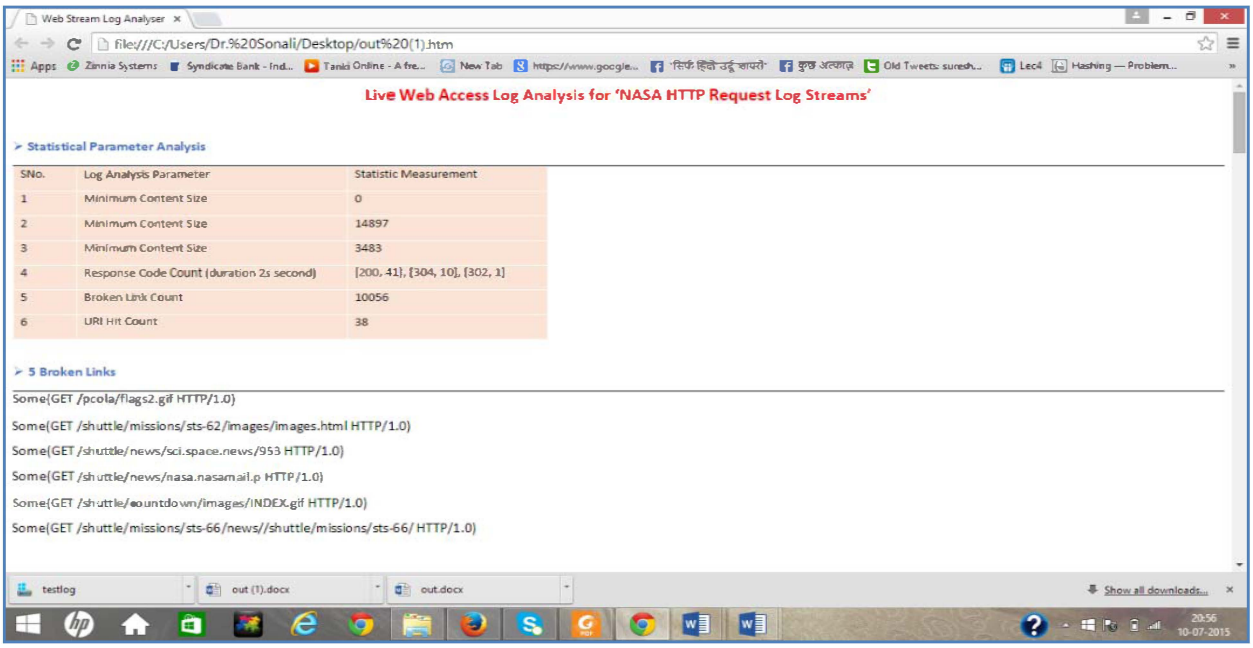

Fig. 17. Snapshot of the Dashboard

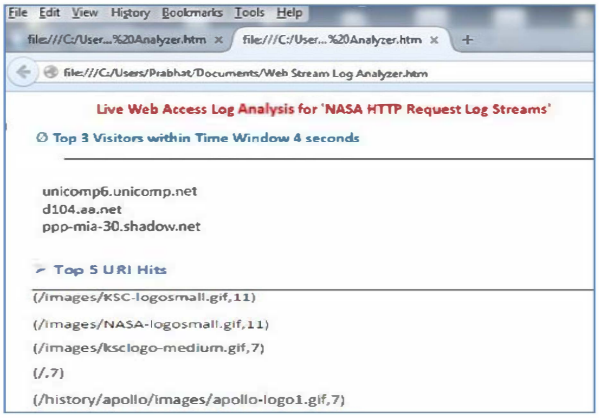

Fig. 18. Snapshot of Dashboard showing Top 3 Visitors & Top 5 URI Hits

V. CONCLUSION

The high speed streaming data analysis of web log streams has been fairly useful for implementation of real time tags and alerts, anomaly detection, inactivity alerting, troubleshooting, diagnostics, server monitoring and application usage analytics. Apache Spark is a powerful tool of stream processing performs in-memory storage and computation of web generated logs in distributed as well as real time manner. The proposed framework enables a prompt analysis of web logs and develops a statistical profile of various web usage patterns by using NASA Kennedy Space Center Server dataset. In future, the statistics generated with the help of web log streams can be further utilized to develop applications such as web usage mining, opinion mining and recommendation systems. In the similar way, error log streams can also be monitored to developed real time secure fault tolerant systems.

Acknowledgment

The authors are thankful to the people of Kennedy Space Center for their submission of this log data. We gratefully acknowledge Jim Dumoulin, Martin Arlitt and Carey Williamson for their significant contribution regarding making the dataset available for research purpose.

References

1. K. R. Suneetha, and R. Krishnamoorthi, "Identifying User Behavior by Analyzing Web Server Access Log File," IJCSNS International Journal of Computer Science and Network Security, vol. 9, pp. 327-332 2009.
2. Tasawar Hussain, Dr. Sohail Asghar, Dr. Nayyer Masood, "Web Usage Mining: A Survey on Preprocessing of Web Log File" 978-14244-8003- 6/10/2010
3. Theint Theint Shwe, Thida Myint , "Framework for Multi – purpose Web Log Access Analyzer", 2010, IEEE, Vol. 3, 289
4. Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Hua Zhu. 2000. "Mining Access Patterns Efficiently from Web Logs". In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, Springer-Verlag, London, UK, 396-407.
5. Mamshad Mobasher, Robert Cooley, and Jaideep Srivastava "Prsonalization Based On Web Usage Mining", August 2000, Vol.43, No. 8, ACM.
6. Akamai Stream Analyzer, http://www.atoll.gr/media/brosures/Stream_Analyzer_Service_Description.pdf
7. Stermsek, Gerald, Mark Strembeck, and Gustaf Neumann. "A User Profile Derivation Approach based on Log-File Analysis." IKE. Vol. 2007. 2007.
8. Maria Carla Calzarossa, Luisa Massari, "Analysis of Web Logs: Challenges and Findings" Performance Evaluation of Computer and Communication Systems. Milestones and Future ChallengesLecture Notes in Computer Science Volume 6821, 2011, pp 227-239
9. Kosala, R., Blockeel, H., (2000). "Web Mining Research: A Survey", ACM 2(1):1 -15.
10. Beyond Web Application Log Analysis using Apache Hadoop http://www.orzota.com/wpcontent/uploads/2014/04/loganalysis-paper.pdf
11. Joshila Grace, L. K., V. Maheswari, and Dhinaharan Nagamalai. "Analysis of Web Logs and Web User in Web Mining." arXiv preprint arXiv:11015668 (2011).
12. Lalit Agrawal, Anil Patidar, "Multifunction Web Log Analyzer: A Survey", International Journal of Computer Science and Information Technologies, Vol. 6 (2), 1184-1187.
13. Park, Soyeon, Joon Ho Lee, and Hee Jin Bae. "End user searching: A Web log analysis of NAVER, a Korean Web search engine." Library & Information Science Research 27.2 (2005): 203-221.
14. Wang, Suozhu, et al. "A Method of E-government Website Services Quality Evaluation Based on Web Log Analysis." LISS 2013. Springer, 2015. 1157-1162.
15. Mohandas, Madhury, and P. M. Dhanya. "An Exploratory Survey of Hadoop Log Analysis Tools." International Journal of Computer Applications 75.18 (2013): 33-36.
16. Vaidya, http://hadoop.apache.org/docs/stable/vaidya.html
17. Revisiting the physician: Hadoop Vaidya, http://www.hadoopsphere.com/2013/01/revisitingphysician-hadoop-vaidya.html
18. J. Boulon, A. Konwinski, R. Qi, A. Rabkin, E. Yang, andM. Yang, Chukwa, a large-scale monitoring system,In First Workshop on Cloud Computing and itsApplications (CCA '08), Chicago, IL, 2008J.
19. Tan, X. Pan, S. Kavulya, R. Gandhi, and P. Narasimhan, Salsa: Analyzing logs as state machines, In Workshop on Analysis of System Logs, San Diego, CA, Dec 2008.
20. J. Tan, X. Pan, S. Kavulya, R. Gandhi, and P. Narasimhan, Mochi: visual log-analysis based tools for debugging hadoop, In Proceedings of the 2009 conference on Hottopics in cloud computing, HotCloud'09, Berkeley, CA, USA, 2009.
21. Weblog_parse source:http://acme.com/software/weblog_parse accessed on July 2015
22. Darryl C. Burgdorf, Web Log source: http://awsd.com/scripts/weblog/, accessed on July 2015.
23. Gasson, Gaelyne R. (April 2000). "Web Analysis Using Analog". Linux Journal 2000 (72es). ISSN 1075-3583.
24. Spark Streaming Programming Guide - Apache Spark, https://spark.apache.org/ accessed on July 2015
25. Apache Kafka, A high-throughput distributed messaging system source: http://kafka.apache.org/ accessed July 2015.
26. Intro to Apache Spark : http://cdn.liber118.com/workshop/itas_workshop.pdf, accessed on July 2015
27. Zaharia, Matei, et al. "Spark: cluster computing with working sets." Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. (2010).
28. Dataset of NASA Kennedy Space Center Server source: ita.ee.lbl.gov/html/contrib/NASA-HTTP.html accessed on July 2015.