ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 5, May 2016

Performance Enhancement of Big Data in Mobile Networks

First Tejaswini B, Second Pavithra KP, Third Namitha Kaveramma CR

Computer Science and Engineering department, Information Science and Engineering department, Information Science and engineering department.

Visveswaraya Technological University, Visveswaraya Technological University, Visveswaraya Technological University.

***Abstract*—This is an era of smart phones. Smart phones have made our work much more easier and have taken large data transfer. Hence the service provider cannot fulfill the demand for more bandwidth from the user and it is expensive.**
In this paper we propose a bandwidth enhancement algorithm based on cache coherency where the user data transfer is optimized without compromising the user expectation or the need for service providers to expand their capacity.
The proposed algorithm is compared with existing data transfer techniques and we show through representative analysis the efficiency of the algorithm to keep the same level of communication with less transfer.
Keywords* – *Big Data, Mobile Networks, Performance, Enhancement, Algorithms, , Cache Coherency.

I. INTRODUCTION

With the growing use of smart phones throughout the world, the Internet is becoming increasingly widespread on smart devices. Mobile Internet as it is popularly called these day's enables users to access the internet on the go. While there are many advancements being made in communication speeds through technology and new standards, service providers are unable to invest at the same speed. This has led the service providers to accommodate to new wave of mobile Internet users with existing infrastructure. Mobile devices are getting more powerful and more different every waking moment.

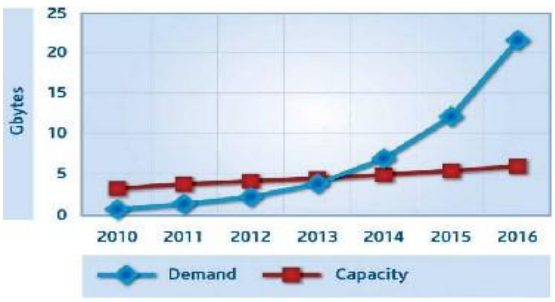

Figure 1 - Demand, Time Plot and Capacity

We have more devices capable of browsing web every day. Figure 2 shows the sales of smartphones all over the world. The graph shows the remarkable growth of smart phones over the last decade and will probably continue to grow a lot faster in the coming years. Smartphones have increased in disk size starting from just kilobytes of flash storage to 256GB of disk space. But it is more than just smart phones; we have smart watches, tablets and even cars and washing machines capable of connecting to the Internet. With this rapid explosion of devices it becomes more important to send data that matters, when needed and on-time keeping bandwidth optimization in mind.

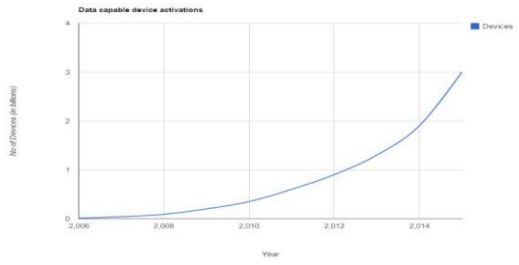

Figure 2 Device Activations over Time (From Google I/O)

There is a continuous development of new standards and features in web browsing. With the introduction of new standards such as HTML5, CSS3 and WebGL are able to perform and make applications we were never able to make before. With such open standards we have the ability to make an application that can run truly cross-platform. But all this comes at a cost. Previously plugins like Adobe's flash and Java were used to execute these tasks. These source files were compiled to byte code which though a security receptiveness, still a highly compressed format. But these open standards require code to be in a plain text format which has now created a new problem. A website now is typically a few hundred

kilobytes, which is within our reach .But with the introduction of WebGL we could be playing games and performing animations on the web browser which range from a few hundred kilobytes to many mega bytes. This might be in the acceptable range for a desktop computer but for a mobile phone this is a huge amount of data to be downloaded from the data network. One of the main reasons for this is the current caching algorithm which downloads the entire file again even when there is a tiny change. We used GIT [1] in [9] and [10] to create entire repositories, which though commonly used leads to higher overheads. GIT subversion control system [2] operates with a high speed, small in size to implement on a machine, its high levels of compression [3] and its distributed nature is very suitable for optimization. We brought subversion concept [9] and used in general non networking versioning system, to mobile communication world for the first time and show that it can be used for enhancing network performance.

In this paper, we propose an efficient and simpler system that calculates the difference of cache files and merges efficiently with the existing with the present file. The rest of this paper is organized as follows. Section II relates our study to previously published results. Section I The steps involved in the proposed solution are elaborated in Section III. Section IV presents our testing results .Finally, Section V concludes this paper.

II. EXISTING SYSTEM

Mobile networks are becoming popular in accommodating to Internet through smart phones. The next generation phones provide universal communication of voice,video and data through the hand held devices. It is becoming very expensive for service providers to accommodate higher bandwidth without investing on new technology or expansion. In this paper we proposed a bandwidth optimization algorithm based on cache coherency where the user data transfer is enhanced without compromising the user expectation or the need for service providers to expand their capacity. The proposed method is compared with existing data transfer techniques and we showed through simulation and analysis the efficiency of the algorithm to keep the same level of communication with less transfer. We compared the efficacy of the proposed method to the two year measured result available in Github and showed that the measurement corroborates with the simulation result validating the proposed solution.

The typical architecture of the system is shown in Figure3. Simple additions such as the Diff server and the Diff client to the software side can help achieve huge bandwidth gains. Figure 4 describes the time sequence operation.

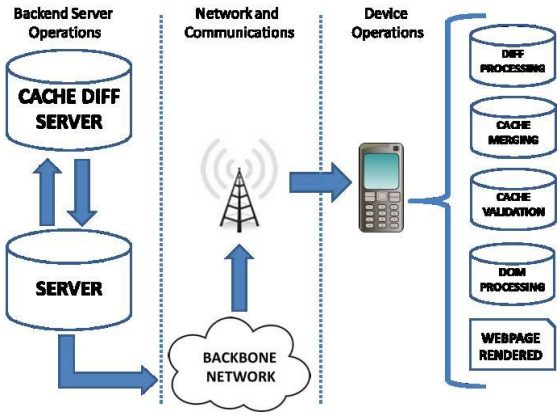

Figure 3 - High Level Network Architecture

When an HTTP request is sent to the server, the serverprocesses the request and sends a response. This response is typically an HTML file that has links to various styling and script files i.e the CSS and JS files. These files are also called static files as these files are not dynamically rendered.
This means that these files do not change as often.
Traditional caching algorithms take advantage of this fact by storing the timestamp of the file that is requested allowing the browser to check if the file was modified after. If it has not been modified then the file is rendered from cache saving valuable bandwidth. If this file was indeed changed ie, has a later timestamp, then the browser downloads the entire file again and caches the new file, dumping the old one.
Usually these changes are really small like minor tweaks which fix bugs or for better performance gains. Our algorithm takes advantage of this by constructing a Diff file. A diff file is the result of a difference algorithm operating on versions of the files to find the difference or the changes that are made to the file. This makes sure that the data transmitted only has the additional information and not the entire file.
The static files served by the server are usually stored in a We check performance by software testing.

Testing defines the status of the working functionalities of any particular system. Through testing particular software one can't identify the defects in it but can analyses the performance of software and its working behavior. By testing the software we can find the limitations that become the conditions on which the performance is measured on that particular level. In order to start the testing process the primary thing is requirements of software development cycle. Using

this phase the testing phase will be easier for testers. The capacity of the software can be calculated by executing the code and inspecting the code in different conditions such as testing software by subjecting it to different sources as input and examining the results with respect to the CDN (Content Delivery Network) in distributed server architecture or on the main server in case of small websites. When these static files are requested by the device, these files are run through a "Cache Diff Server". This Diff server uses a 6-digit hex hash to identify the version of the file on the device and calculates the difference from the current version to the version of the device. This difference along with the new 6 digit hex hash is then sent over the network. This difference file received by the mobile device is first validated. Once validation is successful, the device starts patching the Cache that it maintains. This process is called Merging. Once patching and merging are complete, the device then updates the version number to the new version number and the requested file is now rendered from the cache.

III. PROPOSED SOLUTION

Presently, when a mobile phone accesses the Internet, the inherent protocol transfers the entire data back to the phone. The data transferred from the base station to the end device (i.e., phone) occupies the most important resource in the air, namely the bandwidth.
The current algorithm [9] is as follows:

```
READ requested File, cached Files
IF requested File NOT IN cached Files
THEN
READ fetched File FROM network Loc
ADD to cached Files
ELSE
IF cached Files NOT up to date THEN
READ fetched File FROM network Loc
REPLACE outdated File IN cached Files
ELSE
READ requested File FROM cached Files
ENDIF
```

fetched from the network location and the cache is updated with the requested file. The algorithm [9] looks simple enough to implement, but we argue the inherent inefficiency in using such an algorithm for a mobile device communication. It may make sense when a person uses a fiber link to upload the file to cache, however to download the requested file blindly bleeds the most expensive resource, namely the bandwidth in a mobile network.
We propose a solution that is simple to implement, practical and optimizes the network bandwidth in mobile networks. Our proposed bandwidth enhancement algorithm is:

```
READ requested File, cached Files
IF Cached Files DO NOT exist THEN
CREATE Repository Cached Files
IF requested File NOT IN cached Files
THEN
DOWNLOAD requested File
UPDATE cached Files
ELSE
IF cached Files NOT up to date THEN
DOWNLOAD diff File
MERGE diff File with cached Files
UPDATE cached Files version
ENDIF
ENDIF
READ requested File FROM cached Files
```

Earlier, we proposed a solution [9] based on GIT subversion control system [1]. GIT subversion control system [2] operates with a high speed, small in size to implement on a machine, its high levels of compression [3] and its distributed nature is very suitable for optimization. We bring in the subversion concept used in general non networking versioning system, to mobile communication world for the first time and showed that it can be used for enhancing network performance. We better that proposed algorithm with a new efficient method that transfers data on need- basis thus enhancing the bandwidth.

Better use of Bandwidth

The proposed algorithm achieves high speed of transfer that is better than any other known systems. The cache coherence in the algorithm being proposed in this paper minimizes the data transfer without compromising. the customer expectation. Website developers and vendors will maintain cache repositories of the most frequently downloaded files of that website

IV. IMPLEMENTATION

Implementation is shown by software testing.Testing is the process of trying to discover every feasible fault or weakness in a work product.

Test case ID	Test case name	Test case description	Test steps				Test results P/F
			Step	I/p given	Expected o/p	Actual o/p	
TC01	User Registration	To check that the user has entered all the required details	Registering with user details	Required details for registration	Registration is Success full	Registration is Success full	Pass
	User Registration	To check that the user has entered all the required details	Registering with user details	Any Required details for registration Is missing	Registration is Success full	Registration is Un Success full	Fail
TC02	User Login	To verify Admin/User Registered	User login	Username & Pswd	Login Successfully	Login Successfully	Pass

It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the goal of ensuring that the Software system meets its requirements and user expectations and does not fail in an undesirable manner. There are various types of test. Each test type addresses a specific testing requirement.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page

Test Results: All the test cases mentioned above clearly explains about how the testing is done and results of the test have been mentioned in the above table.

V. CONCLUSION

Mobile networks are becoming popular in accommodate to Internet through smart phones. The next generation phones provide everywhere communication of voice, video and data through the hand held devices. It is becoming very expensive for service providers to accommodate

higher bandwidth without investing on new technology or expansion. In this paper we proposed a bandwidth optimization algorithm based on cache coherency where the user data transfer is optimized without compromising the user expectation or the need for service providers to expand their capacity.

The proposed method is compared with existing method and we showed through the software testing the efficiency of the algorithm to keep the same level of communication with less transfer.
We compared the usefulness of the proposed method to the two year measured result available in Github and showed that the measurement corroborate with the testing result validating the proposed solution.

ACKNOWLEDGMENT

We are extremely happy to submit the journal on the topic" **Performance Enhancement of Big Data in Mobile Networks".**

We express thanks to our guide Ms. Tejaswini B, Computer Science and Engineering Department, for constant motivation and support. Finally, we would like to thank each and every one helped us in the successful completion of this journal.

REFERENCES

[1] JC Hamano, "GIT – a stupid content tracker", Proceedings of the
Linux Symposium 2006, Ottawa, Canada
[2] Scott Chacon, "Pro Git", Apress publications, Section 4.1 to 4.11 and Section 9.2 to 9.6
[3] Jean-loup Gailly, Mark Adler, "Compression Algorithm(deflate)",http://www.gzip.org/algorithm .txt
[4] [LZ77] Ziv J., Lempel A., "A Universal Algorithm for Sequential
Data Compression," IEEE Transactions on Information Theory,
Vol. 23, No. 3, pp. 337-343.
[5] Wikipedia, "General Packet Radio Service", http://en.wikipedia.org/wiki/GPRS
[6] Wikipedia, "Enhanced Data Rates for GSM Evolution",
http://en.wikipedia.org/wiki/EDGE
[7] 4G Americas, "4G Mobile Broadband Evolution – Series of White
Papers", http://www.4gamericas.org
[8] Rysavy Research, "Mobile Broadband Capacity Constraints And
the Need for Optimization",
http://www.rysavy.com/Articles/2010_02_Rysavy_Mobile_Broad
band_Capacity_Constraints.pdf
[9] A. Ramaprasath, K. Hariharan, A. Srinivasan, "Cache Coherency
Algorithm to Optimize Bandwidth in Mobile Networks", Springer Verlag, Lecture Notes in Electrical Engineering, Networks and
Communications, Chapter 24, Volume 284, 2014, pp 297-305
[10] A. Ramaprasath, K. Hariharan, A. Srinivasan, "Cache Coherency
Algorithm to Optimize Bandwidth in Mobile Networks", Fifth
International Conference on networks and Communication,
Chennai, (NetCom 2013), Dec 28, 2013