EasyChair Preprint

№ 8422

Network Traffic Analysis in Map Reduce for Bigdata Applications

K Lakshmikanth, K N Prajwalsidhu, H N Rakshitha and
A V Krishnamohan

July 10, 2022

Network traffic analysis in Map Reduce for bigdata applications

Lakshmikanth K [1], Prajwalsidhu K N [2], Rakshitha H N [3] and A V Krishnamohan [4]

[4] Assistant Professor

[1 – 4] Department of computer science and engineering, Siddaganga Institute of Technology, Tumkur – 572102

1si18cs050@sit.ac.in [1], 1si18cs073@sit.ac.in [2], 1si18cs143@sit.ac.in [3] and krishnamohan_av@sit.ac.in [4]

Abstract. Through the use of parallel map and reduce activities, the map-reduce programming methodology makes it easier to handle massive amounts of data in groups of items. While significant work has been done to boost the efficiency of map reduce tasks, this work ignores the network traffic created during the shuffle phase, which is vital to boosting efficiency in general. Historically, a hash function is used to partition intermediate data between reduction activities, which, however, are not traffic efficient due to the fact that the network topology and the size of the data associated with each is not considered key code. In this paper, we will examine how to reduce network traffic costs for a map reduction process by designing a new intermediate data partitioning scheme. Plus, together let's not forget the hassle of aggregator location, where each aggregator can reduce the combined traffic of multiple map activities. A set of assigned algorithms based primarily on decomposition is proposed to address the problem of large-scale optimization for large data programs, and a web set of rules is also designed to dynamically modify the partitioning and aggregation of data. In the end, the simulation results demonstrate that our suggestions can still significantly lower network traffic, both online and offline.

Keywords: Bigdata, network traffic, Map Reduce, web application

1. Introduction

Map-reduce programming model has become the best unstructured bigdata processing. due to its easy-to-use programming paradigm and automated handling

of parallel execute. Leading companies like Yahoo!, Google, and Facebook have adopted the open Hadoop Map Reduce framework for a number of big data applications, including machine learning, bioinformatics, and cyber security.

Map reduce divides an activity into two phases: map and reduce, each of which is carried out by a different map and reduce method. To convert the initial input splits into intermediate data in the form of key/value pairs, map activities are carried out concurrently throughout the map phase. These key/value pairs are kept locally and broken up into multiple data divisions, one for each reduce task.

Each reduce job receives a portion of the data partitions from each map task during the reduce phase in order to produce the final output. There is a shuffle step in between the map and reduce stages. This stage involves processing, partitioning, and sending the map phase's generated data to the proper computers for the reduce phase. The effectiveness of data analytics systems may be severely hindered by the pattern of network traffic that results from all map tasks to all reduce tasks.

In this paper, we discussed network traffic analysis on the Map-Reduce programming paradigm using distributed algorithms and hash-based partitioning, as well as investigating and modelling the same in the cloud.

2. Literature survey

2.1. System

A system is a well-organized collection of interdependent components that are linked together in line with a plan to achieve a certain objective. Its primary characteristics include organizing, interaction, reliance, integration, and a main objective.

2.2. Analysis

A detailed investigation of a system's various processes and their interrelationships both within and outside the system is referred to as analysis. The definition of the system's boundaries and establishing whether or not a future system should consider other connected systems are two components of analysis. Data about the accessible files, decision-making processes, and transactions handled by the present system

are collected during the research. This comprises collecting data by analysing it using organized methodologies.

2.3. System Analysis

The use of the system method to problem solving with computers is known as system analysis and design. A system's outputs and inputs, processors, controls, feedback, and surroundings must all be considered while reconstructing it.

2.4. Existing System

However, this is inefficient since network topology and the data amount associated with each key are not taken into account. Traditionally, a hash function has been employed to divide intermediate data between reduce tasks.

Problems with the current system

- No intermediate data retrieval.
- Less security.

2.5. Proposed System

In this study, we investigate data partitioning and aggregation for a Map Reduce operation, with the aim of minimizing total network traffic. For big data applications in particular, we describe a distributed technique that divides the original large-scale problem into several parallelisable subproblems. A dynamic mechanism for handling data split and aggregation has also been created online. Lastly, thorough simulation results demonstrate that our methods may significantly reduce network traffic costs in both offline and online settings.

Benefits of the suggested system

- Lowers the cost of network traffic in both offline and online scenarios.
- Faults tolerance is increased.

3. Architecture

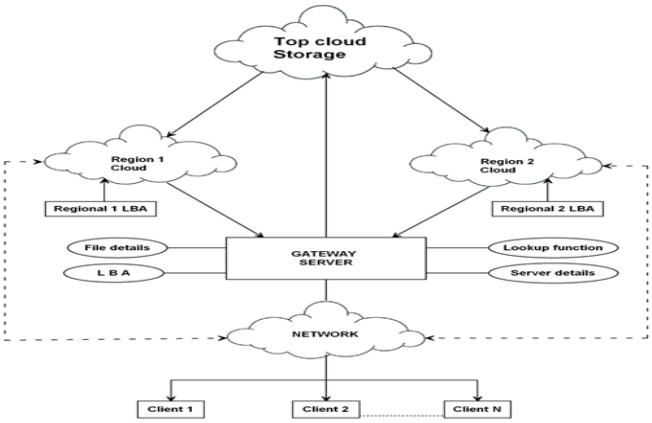

Fig. 1. Architecture diagram

Map Reduce programming model works on clusters of commodity hardware; different clients (N clients) can work on different servers. For our use case we considered 2 servers and a main server, few clients work on server 1 and file details and the logical base address of server 1 are stored in SQLYog. Similarly, for server2, file details (list of files uploaded and downloaded on server 2) and LBA details are stored in SQLYog. When some other client requests to download a file then it will directly interact with the main server, all the server 1 and server 2 file details are fetched from the main server. It will act as metadata for all file details.

3.1. Use case diagram

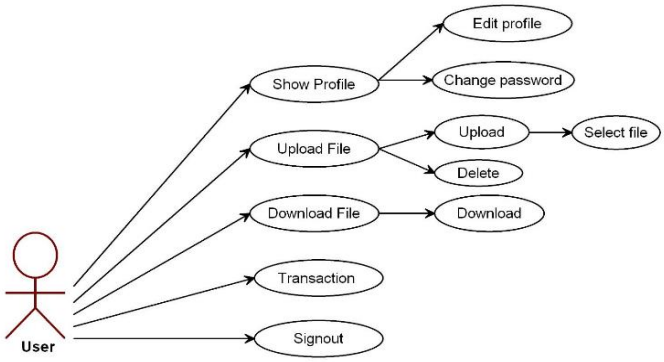

Fig. 2. User's use case

User can perform the following activities such as show profile, upload file, download file, transaction details and sign out. Show profile menu will display the edit profile and change password options from this user can change his profile details. The upload file menu will display the options such as upload and delete to

upload a new file or delete an existing file detail which correspondingly add the file contents to SQLYog and remove the existing file contents from SQLYog. Similarly, download menu will display download option to choose to request file to download contents from cloud/local storage.

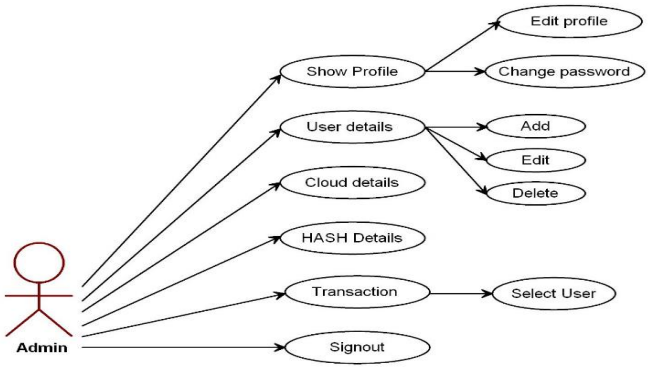

Fig. 3. Admin's use case

Admin can perform the following activities such as show profile, user details, cloud details, hash details, transaction details and sign out.

Show profile menu will display the edit profile and change password options from this admin can change his profile details. The user details menu will display all the user details that are stored in m_user table. The cloud details menu will display the all-server details such as region and IP details. Similarly, the hash details menu will display the hashtag for each file block stored m_HASH table.

3.2. Data flow diagram

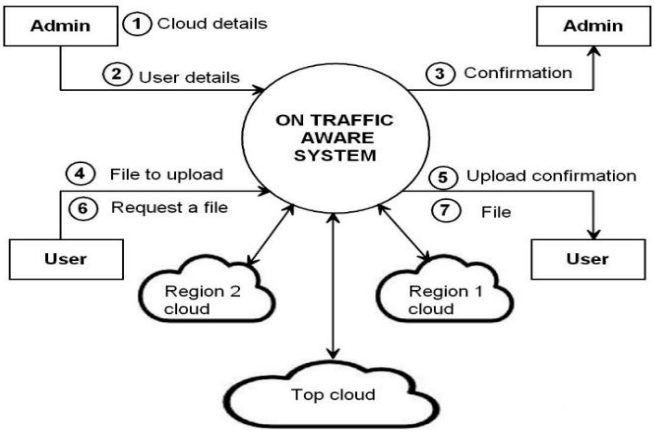

Fig. 4. Context analysis

Admin can check the user details and cloud details based on his/her activity system will confirm the results to an admin. User can upload file or download file from Server 1 or Server 2 all the file details are uploaded or downloaded from top cloud (Main Server) system will confirm the results to the user.

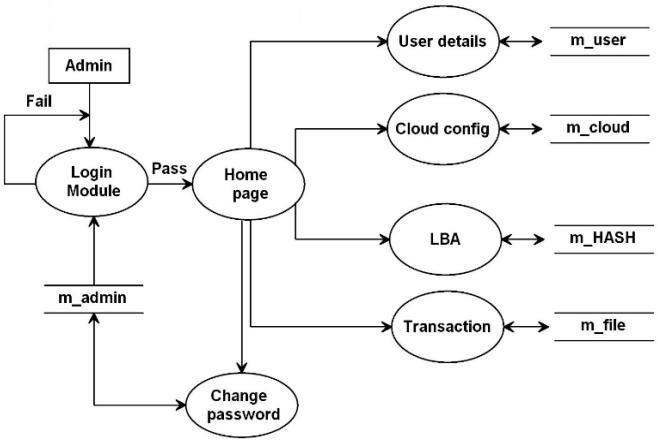

Fig. 5. Admin session workflow diagram

Admin can login through his/her credentials, Admin entered credential details are verified with stored credentials in table m_admin if that matches admin will login to the home page. Admin can view the user details these are stored in a table called m_user and he/her can view the hash value generated for each block in a file through m_HASH table. He/Her can also view the transaction details of each user through m_file table in SQLYog.

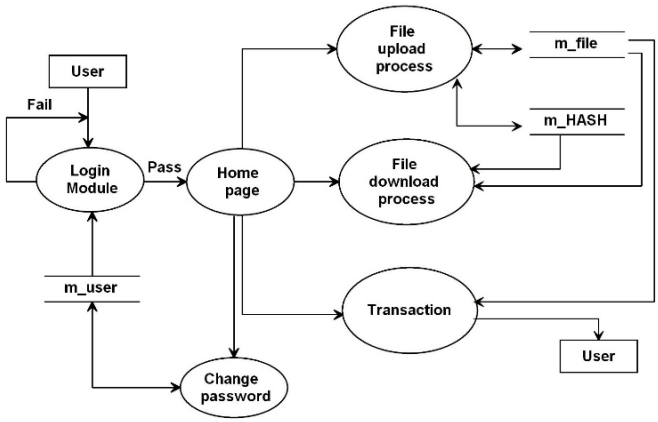

Fig. 6. User session workflow diagram

User can login through his/her credentials, User entered credentials are verified with stored credentials in table m_user if that matches user will login to home page. Screen will appear with list of activities that user can perform file upload process, file download process and transaction details. When the user uploaded the file, file

details are stored in m_file and corresponding hash details of file stored in m_HASH. When user downloaded the file corresponding file details and hash details are matched in local and cloud finally file downloaded. Also, user can check his/her transaction details stored in table user.

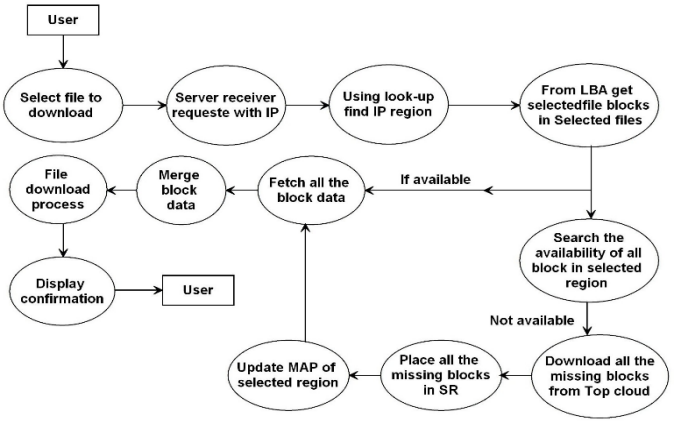

Fig. 7. File Download Process

File download process splits in to several sub tasks, User has to select a file which needs to be download then server receives request with IP based on look-up table it will find IP region whether it will belong to Server1 or Server 2. Based on LBA it will select the blocks for a requested file then it will search each block in selected region if it is available then fetch the block data, if not available then it will download all the missing blocks from top cloud that is main server and merge the block data and gives the final file content.

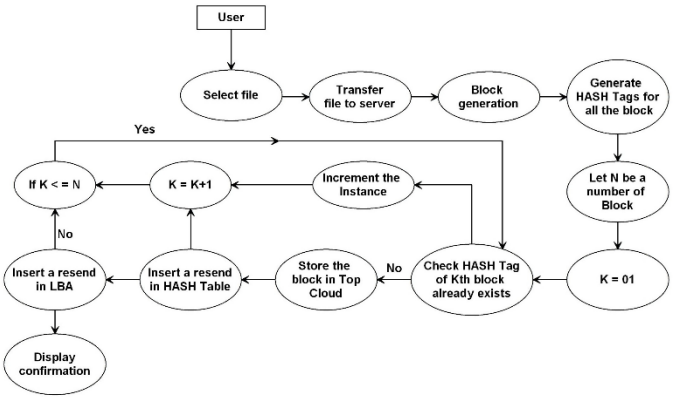

Fig. 8. File Upload Process

File upload process includes several sub activities in upload a file activity, User has to select a file that needs to be upload then it will split file into several blocks based on a packet size (500 bytes) and it will generate hash tag for each block. It will

check if the generated hash tag for a block already exists if it is then it will increment the instance of a block. If not, it will create a new block and store this content in SQLYog.

4. Results

4.1. Figures and Tables

File storage table

Table 1. m_file_one stores the all-uploaded file as block wise in detail.

file_no	file_name	file_size1	file_size2	padding	no_blocks	hash_blk_nos	user_id	date	time	day
1	first.txt	1058	0	0	3	31-33-34-	2	20-06-2022	22:42:16	Monday
2	second.txt	1056	0	0	3	31-42-43-	2	20-06-2022	22:42:29	Monday
3	main.txt	1058	0	0	3	31-33-34-	2	20-06-2022	22:42:46	Monday

Table file_no which gives number of files in table, file_name column gives file name for each file, file_size this will gives size of each file in bytes, no_blocks this will gives total number of blocks for each file based on packet size of 500 bytes, hash_blk_nos this will gives block numbers based on hash value generated on each block. The remaining columns such as time, date and day will gives the file uploaded time, day and date by user.

Hash code table

Table 2. m_hash stores the generated hash code of each uploaded file block.

hash_unique_code	hash_code	blk_name	no_instnce	uploaded_status
1	14de3af8024bdbf66052e53b61f2dd8e	1_blk_0	3	no
2	ec72af92ad465ff29de1149faff21528	2_blk_1	2	no
3	2781c7b5e3cd12e5068c627cd299f126	3_blk_2	2	no
4	371aafb2765cd84c3998be3df71b8062	4_blk_1	1	no
5	18568d748dcf23a8d6ecb7012035b03e	5_blk_2	1	no

Table hash_unique_code which gives the number of hash codes in the table, hash_code which gives unique hash tag generated from MD5 algorithm for each block of max size 500 bytes, unique block name for each block and no_instances which tells the no of times block content repeated by different files.

Result Analysis

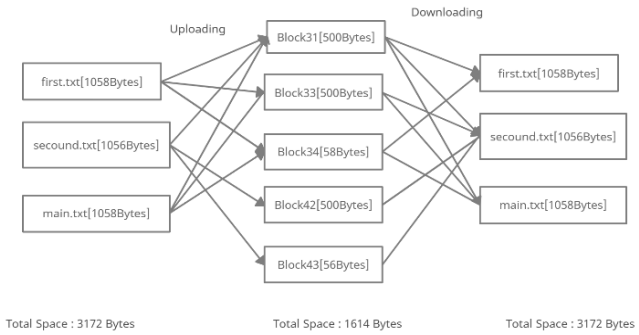

Fig. 9. File Upload-Download Process

Initially we are uploading 3 files of sizes 1058,1056,1058 bytes with total size of 3172 bytes. when we are uploading each file, it will divide file content into blocks based on packet size of 500bytes. As per this logic first file first.txt will divided into 3 blocks block 31,33 of 500 bytes size and 34 block of 58 bytes. Similarly file 2 second.txt divided into 3 blocks where one block content already exists it's named as block 31 with other two different content blocks 42 and 43. The main.txt file divided into 3 blocks where all the blocks contents are already exists hence blocks are named as 31,42 and 43. All the blocks are stored in local space and cloud, what we need to analyse is when we are uploading file total file size is 3172 bytes where as we are storing only 1614 bytes that means we are saving ~1550 bytes of space in memory. Whereas while downloading corresponding block instances are fetched from m_HASH table and downloaded files total size is 1372 bytes.

Simulation Analysis

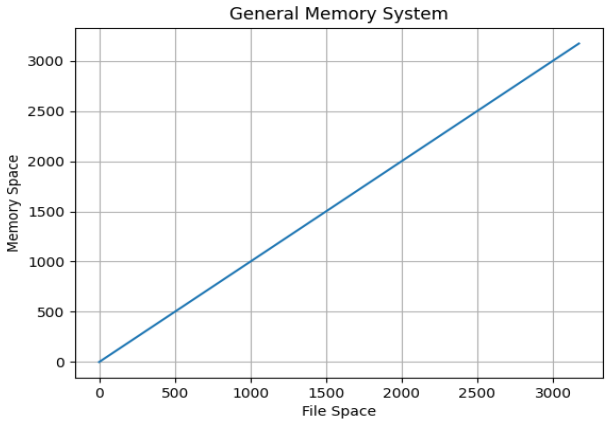

Fig. 10. General Memory System

File size parameter in the X axis and memory space on the Y axis, as per the general method of storing files and downloading files it will store all file contents of each file which means the file files list increases i.e., file size increases memory space also increases since we are storing all the file contents so memory space also increases. So, the graph results in Linearly.

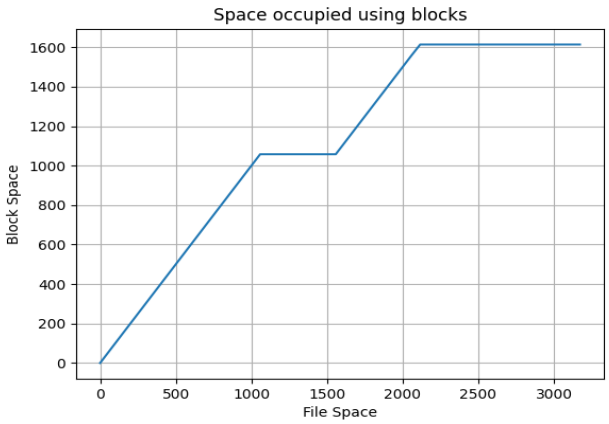

Fig. 11. Memory Occupied System

File size parameter in X axis and memory space in Y axis, consider 3 files for our use case first file of size 1058bytes when we uploaded it all the blocks are new blocks so the graph results in linearly till 1058bytes, after when we upload 2nd file one of the block out of 3 blocks of the second file is duplicate so only file size increases to 1558 and memory size remains constant to 1058 only for remaining 2 blocks of the second file it becomes linearly increase since both blocks are of different contents. X axis and Y axis is 2114 and 1614 respectively. When we upload third file since all 3 blocks of 3rd file are same so only X axis increases and Y axis remains constant. X axis results in 3172 bytes and Y axis results in 1614 bytes.

The graph increases linearly when the block contents are new and not exists and it becomes constant when block contents are already exists.

5. Conclusion and future scope

In order to lower the costs of network traffic for big data applications, we investigate in this study the combined optimization of intermediate data partition and aggregation. We provide a three-layer approach to solving this issue and express it as a mixed-integer nonlinear problem that may be converted into a linear

form and addressed using mathematical tools. We design a distributed approach to address the issue on several machines in order to handle the large-scale system caused by the vast amount of data. Additionally, we strengthen our method to handle the map-reduce task in an online manner when some system parameters aren't provided.

6. References

1. H. Ke, P. Li, S. Guo and M. Guo, "On Traffic-Aware Partition and Aggregation in MapReduce for Big Data Applications," in IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 3, pp. 818-828, 1 March 2016, doi: 10.1109/TPDS.2015.2419671. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
2. Y. Dong, B. Tang, B. Ye, Z. Qu and S. Lu, "Intermediate Value Size Aware Coded MapReduce," 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), 2020, pp. 348-355, doi: 10.1109/ICPADS51040.2020.00054.
3. M. Supriya, 2017, Traffic-Aware Partition and Aggregation in Map Reduce for Big Data Applications, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) RTICCT – 2017 (Volume 5 – Issue 17).