Data Optimization Techniques using Bloom Filter in Big Data

Ritu Jain
M. Tech. Scholar
CSE Dept. MIET
Meerut, India

Mukesh Rawat
Assistant Professor
CSE Dept. MIET
Meerut, India

Swati Jain
Security Officer
GID, IBM
Bangalore, India

ABSTRACT
Due to the advent of new technologies, devices, and communication means like social networking sites, the amount of data produced by mankind is growing rapidly every year. Traditional computing techniques are not enough to process that much large amount of data. Hadoop is a bunch of technology & have capacity to store large amount of data on Data nodes. Hadoop uses MapReduce algorithm to process and analyze large scale datasets over large clusters. MapReduce is essential for Big Data processing. This algorithm divides the task into small parts and assigns those parts to many computers connected over the network, and collects the results to form the final result dataset. Bloom filter technique is probabilistic data model which is used to make processing of data more efficient. Implementation of this filter with mapper can reduce the amount of data travel. In this paper we implemented Bloom filter in Hadoop architecture. This help to reduce network traffic over network which save bandwidth as well as data storage.

Keywords
Big Data, Hadoop, MapReduce, Bloom filter.

1. INTRODUCTION
With explosion of data in recent scenario only traditional database is not enough to handle it. With high rate of changing data on web applications there is need of database which can perform to provide consistency as well as partition tolerance.

Present database system work with vertical enhancement that gives scale-in facility for system. That is not enough for huge database like Linkedin, facebook, Amazon etc. That huge amount of data needs to have horizontal enhancement that give scale-out property. By this enhancement any number of nodes can be added with system.

For Big data there is use of MapReduce [6] programming model that perform operation on single large file so that there is no need to split data.

1.1 Basics of MapReduce
Big Companies start using Hadoop. Hadoop can run MapReduce programs written in various languages. MapReduce works by breaking the processing into two phases: the map phase and the reduce phase. Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer.

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves TaskTracker execute the tasks as directed by the master and provide task-status information to the master periodically. The JobTracker is a single point of failure for the Hadoop MapReduce service which means if JobTracker goes down, all running jobs are halted.

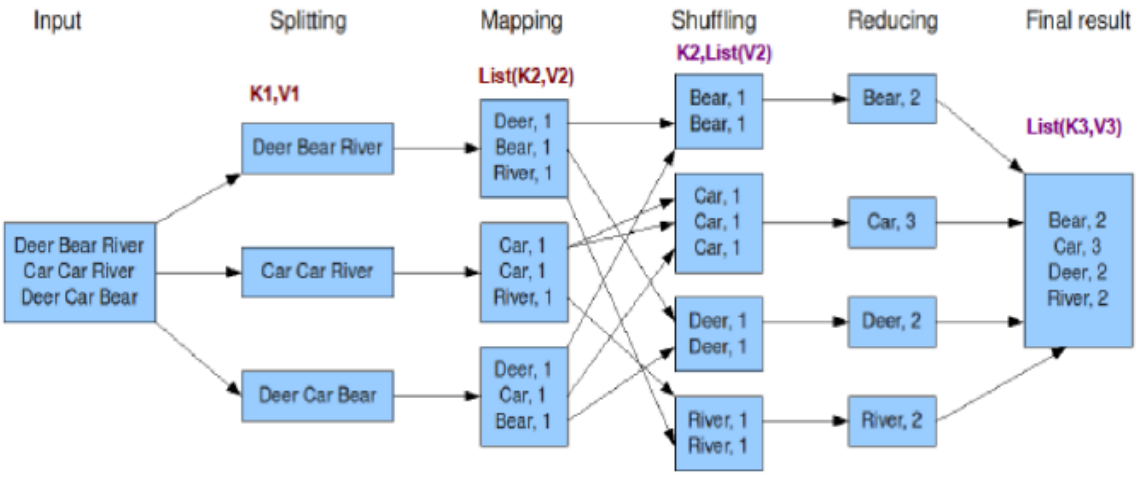

Fig 1: MapReduce

MapReduce consist several phases as shown in fig. 1 splitting, mapping, shuffling after that reducing. Map phase makes key value pair with default key value is page offset and passes these to reduce phase. Reduce phase accept these value pair implement logic on that [6].

1.2 Basics of Bloom Filter

Bloom filter is probabilistic data structure which is space efficient with little error allowable when there is test performed. Its data structure is developed by Burton H. Bloom in 1970 [4]. Bloom filter stores elements in an array using hash functions. Let say set S =x_1; x_2;….; x_n, construct data structure to answer queries about presence of element existence like is y in S? This query does not provide direct result, it just provide idea about data's presence. It also deals with some allowable errors with Bloom filter.

Let assume there are 'm' bit array consider as bloom filter that consist 'n' elements of 'k' bits. Probability to set any bit of array '1' by hash function is '1/m' then probability not set it to 1 will be '1-1/m', if it is not set 1 by any n member of array then probability '$(1-1/m)^n$' since there are k bits in message then probability will become '$(1-1/m)^{nk}$'.

If element found in array i.e. 1 then it should be '1- $(1-1/m)^{nk}$'. For complete message found in array probability becomes 'f' i.e.

$$f = (1 - (1 - 1/m)^{nk})^k \approx (1 - e^{\{-kn/m\}})^k \quad (1)$$

Since item can also indicate FP in bloom filter so FP mostly depend with size of bloom filter. So that should be optimal as FP probability can obtain the minimum $(1/2)^k$ when optimum value of k is,

$$k = (m/n)\ln 2 \quad (2)$$

Size of bit array m is can be chosen as [4]

$$m = -(n \ln p)/(\ln 2)^2 \quad (3)$$

2. RELATED WORK

J.Dean et al [6] proposed distributed computation using MapReduce function model that attract much attention for Big data processing. This model based on key-value pair that processed with data. Pavlo et al. [1] compare both system i.e. parallel DBMS and MapReduce model performance with scalability and fault tolerance in which MapReduce program perform well in term of performance. MapReduce run program once and it is performing task simultaneously without interfering of user. Chu et al. [5] explain MapReduce model for parallelizing machine learning algorithm on single machine with multiple processor. According to Abouzeid et al. [3] that try to fulfill gap between parallel database and MapReduce programming model and implemented a hybrid system that takes best features from both. Hive [2] provided by facebook is presenting feature of SQL so that data can be easily retrieved. By Hive join operation will also be easy by declaring schema of database. These join operation can be perform on Map side as well as Reduce side. There is comparable study of both join operations in scale of time.

Yang et al. [8] proposed a merge component after the reduce function for performing a join operation on two datasets. However, the Map-Reduce-Merge approach introduces an extra processing step that is not there in the standard MapReduce frame-work and therefore will not be found in a standard MapReduce deployment.

A white paper from Oracle [7] describe Big data processing in coming scenario with 'ac-quire', 'organize', 'analyze' and 'decide'. The main focus of the paper was to integrate structured and unstructured data, traditional data warehouse systems and big data solutions that is to use MapReduce as preprocessing tool for traditional, relational sources. The second focus that was mentioned is to plan for and facilitate experimentation with big data. Another white paper

[9] Link to NoSQL database with key value pair and data organize in NoSQL database with integration. It use distribute approach to store data in different nodes. According to paper released by Oracle in 2012 compare the techniques of knowledge discovery through data mining into tradition database with big data environment. It describes the big data capabilities with reference to storage, processing, data integration and statistical analysis.

2.1 Motivation and Aim

As MapReduce framework use for processing large amount of data that have several files that proving high input. So it need to process whole data every time when it required. When files need to joined then there is need to tag columns in a file afterwards compare it to another file. These operation can be performed with data flowing in map and reduce both. Map start to tag data with its join columns then send it to reducers. Reducers perform function on it. So lots of data will participate in flowing.

Bloom filter is technique which provide probabilistic model with defined array size. This array is optimal so that whole data of file will store in it. So that at the time of performing operation whole data will not flow. Till now Bloom filter contributed a lot in networking fields like to find routes and fast searching web contents that having malicious activities.

This proposed work is implementing Bloom filter on Big data so that is whole data will not flow, only user defined array size will participate in operation. This method decrease amount of I/O cost as well as reduces time for operation. Proposed work is implementing Bloom array to Datanodes on Hadoop so that when it instruct to perform any operation then it can perform by Bloom array and produce result. Resultant of proposed work will provide quick result as searching on items or perform other operations.

3. PROPOSED ARCHITECTURE

Fig 2 is showing the proposed implementation of filter for fast accessing data on Datanodes.

(a) Job Initialization: Job is initialized and submitted to Namenode where JobTracker runs. Namenode contain all information that needs to execute in hadoop. Job tracker read job files from distributed file system then create map reduce function.

(b) Map Phase 1: JobTracker assign task for TaskTrackers. These TaskTrackers keep sending signals to prove its aliveness.

(c) Bloom Filter Creation: bloom filter need to create in each mapper. Each mapper create <key, value> pair based on bloom filter that produce intermediate results. Each mapper has its own result after that combined result is send to JobTracker.

(d) Result combination: TaskTrackers need to send result to JobTracker. This time TaskTrackers will send only filtered records. Filter records having all information but it take less space to provide all.

(e) II Map Phase: This record also performs same work as I map phase. Now all that submit to JobTracker where first record already there. Now there is perform of join operation on both filters.

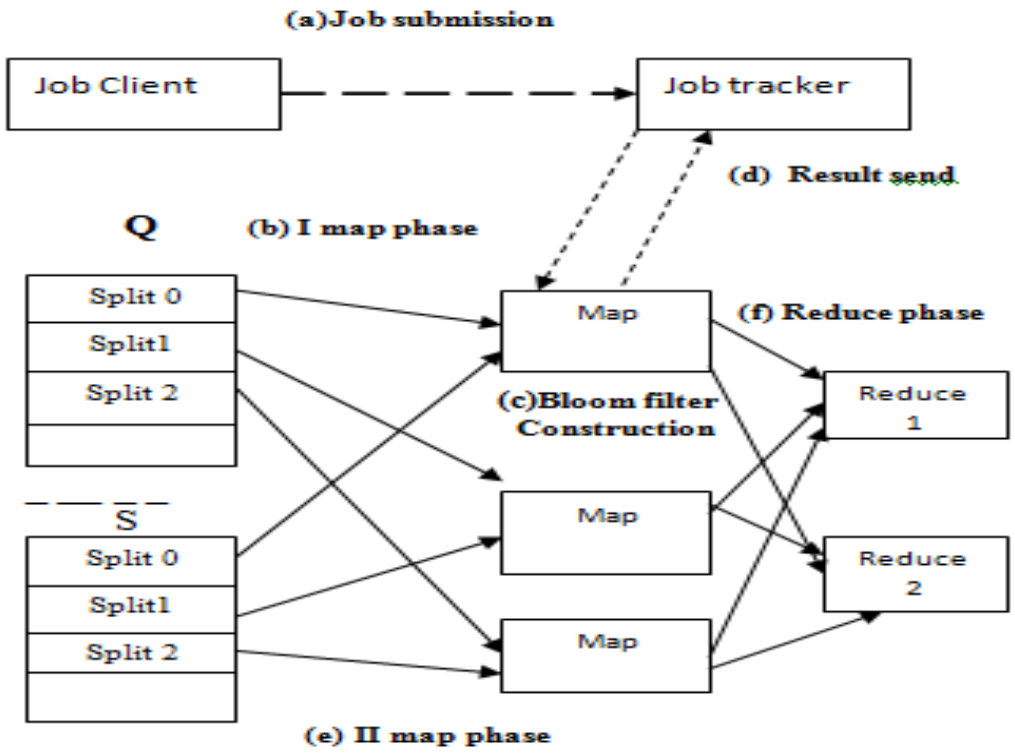

Fig 2: Proposed Implementation of Bloom filter

(f) Reduce Phase: This phase collect all intermediate records and run reduce function to provide results in output path.

Bloom filter construction

Each mapper create bloom filter array which store large amount of data in small size. This will transfer to JobTracker where operation may be implemented. JobTracker construct global filter to execute operation.

4. EXPERIMENTAL DATA AND ANALYSIS

This part contains experiment design and executes process, result and analysis with respect to error probability of false positive (FP). Experiment done with CDH3 machine it is installed on local machine.

4.1 Environment

Experiment is done on CDH3.

CPU: Intel(R) Core(TM) 2 Duo 2.40 GHz

RAM: 2 GB

HDD: 120 GB

System Type: 32 bit operating system

4.2 Creating bloom filter for data

Take sample of 500 sample data elements for Cloudera CDH3. For experiment take 6 different sizes of array and 3 hash functions. These arrays also contain all information as original file. That means no need to traverse all data that in original file, only few amount of data will give all required information. Following are the calculations of error probability of false positive (FP) with different array size by using the formula

$$f = (1 - (1 - 1/m)^{nk})^k \approx (1 - e^{\{-kn/m\}})^k$$

Table 1: Error probability result with CDH3

ARRAY SIZE	ERROR PROBABILITITES
500	0.86
1000	0.47
1500	0.25
2000	0.15
2500	0.091
3000	0.06

Fig 3 and Fig 4 are showing result of data used in CDH3 machine.

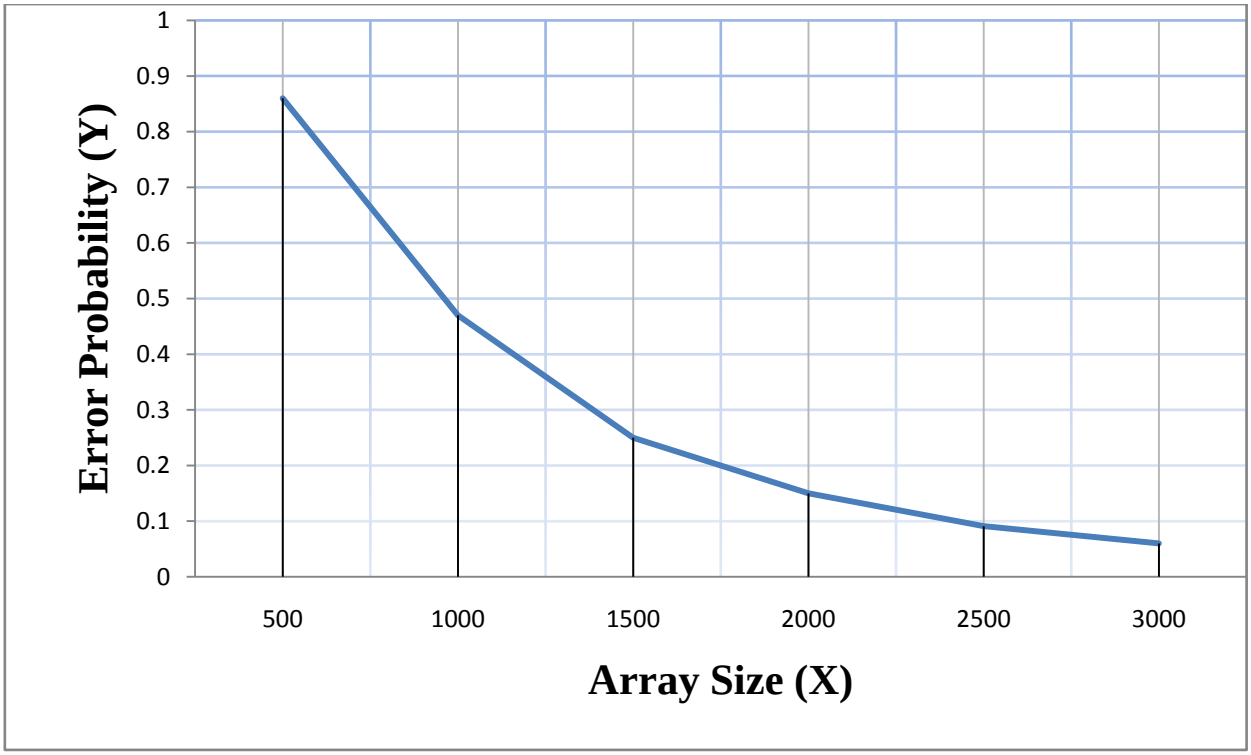

Fig 3: Result between array size and error probability in CDH3

00000000	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
00000000	40	7a	86	e0	23	06	08	08	28	12	80	04	81	e6	79	07	@zt à#...(.€. æy.
00000010	0b	36	00	83	20	e0	03	00	22	19	30	00	c3	40	16	10	.6.ƒ à.."0.Ã@..
00000020	53	50	8d	55	00	8d	4d	02	90	a0	03	20	80	a0	01	25	SP U. M. . € .%
00000030	30	04	c4	91	b9	25	29	c8	4a	35	0e	22	30	84	84	02	0.Ä‘¹%)ÈJ5."0„„.
00000040	80	40	e0	71	9a	7a	02	00	0f	2a	88	09	80	84	28	80	€@àqšz...*ˆ.€„(€
00000050	21	36	45	22	41	60	20	0a	04	63	08	6b	27	62	32	a0	!6E"A` ..c.k'b2
00000060	14	20	04	0c	22	59	01	9a	60	ce	49	70	0a	c0	08	71	. .."Y.š`ÎIp.À.q
00000070	10	58	44	e0	f2	48	59	60	01	02	29	a6	d2	64	01	89	.XDàòHY`..)¦Òd.‰
00000080	08	22	48	30	04	90	08	00	18	48	21	80	c2	f6	c0	00	."H0. ...H!€Âöà.
00000090	30	c0	cc	34	40	88	18	11	45	42	04	21	54	31	da	c0	0ÀÌ4@ˆ..EB.!T1ÚÀ
000000a0	20	04	04	01	cc	38	80	41	90	00	22	00	01	00	18	d4	...Ì8€A ."....Ô
000000b0	2c	01	2a	60	16	89	00	04	40	08	90	03	85	92	60	81	,.*`.‰..@. .…’`
000000c0	20	25	51	f3	95	24	78	54	14	32	0b	10	25	62	40	46	%Qó•$xT.2..%b@F
000000d0	a0	42	46	08	c0	10	84	32	01	87	20	0d	81	84	04	00	BF.À.„2.‡ . „..
000000e0	20	21	72	21	2e	20	43	8b	67	c4	07	42	fc	ad	46	76	!r!. C‹gÄ.Bü-Fv
000000f0	38	e0	04	02	85	38	24	22	c4	a8	c2	2a	41	0a	00	0c	8à...…8$"Ä¨Â*A...
00000100	ae	b9	0e	5a	02	38	0b	80	4c	02	26	02	8a	06	ed	83	®¹.Z.8.€L.&.Š.íƒ
00000110	e4	81	36	18	58	40	00	20	e4	08	87	10	a0	8e	11	6d	ä 6.X@. ä.‡. Ž.m
00000120	b0	84	00	78	20	82	b2	ce	84	84	04	80	2e	18	70	49	°„.x ‚²Î„„.€..pI
00000130	06	08	5c	c1	00	87	00	12	65	00	24	2a	c3	20	01	0a	..\Á.‡..e.$*Ã ..
00000140	04	18	0c	22	46	01	08	04	09	31	08	00	80	24	10	40	..."F....1..€$.@
00000150	b9	54	a1	d0	a4	88	08	a3	58	06	1a	22	09	0c	0a	08	¹T¡Ð¤ˆ.£X."....
00000160	00	20	0e	a0	92	46	10	00	04	91	02	42	00	80	0d	a7	. . ’F...‘.B.€.§
00000170	c1	10	08	10	01	03	00	..	..	..	..	..	..	..	..	..	Á.....

Fig 4: Resultant array output

5. CONCLUSION

Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, storage, search, sharing, transfer, analysis and visualization. This paper focused on one major challenge that is accessing of data. Here concept of Hadoop is implemented to process data which makes use of MapReduce algorithm. This algorithm divides the task into small parts and assigns those parts to many Datanodes. On the top of Hadoop, Bloom filter is implemented to optimize space and time complexities. Firstly Bloom filter is converting original data into array of bits which is helping in reducing space complexity. Array of bits is used to perform search operation instead of original data which is helping in reducing time complexity for search operation. Although bloom filter has its own limitations as it is a probabilistic data structure which means a search query will return either "possibly in set" or "definitely not in set". So there are chances of getting false positive results. The

experiment results in reducing the probability of getting false positive as shown in table 1.

6. ACKNOWLEDGEMENT

Our thanks to everyone who supported us to complete this experiment successfully.

7. REFERENCES

[1] A. Pavlo, A. Rasin, S. Madden, M. Stonebraker, D. DeWitt, E. Paulson, L. Shrinivas, and D. J. Abadi . A comparison of approaches to large scale data analysis. Proceed-ings of the 2009 ACM SIGMOD International Conference on Management of data, pages 165-178, year 2009.

[2] Apache Hive. Available at http://hive.apache.org

[3] Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel J. Abadi, Avi Silberschatz, and Alex Rasin. Hadoopdb: An architectural hybrid of mapreduce and dbms technolo-gies for analytical workloads . Proceedings of the VLDB Endowment, Pages 922-933, Vol 2, Issue 1 2009.

[4] B. H. Bloom "Space/time trade-offs in hash coding with allowable errors." Commun. ACMvol. 13, no. 7, pp. 422426, 1970.

[5] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun. . Map-reduce for machine learning on multicore. . NIPS, 2006 pages 281-288 2006.

[6] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clus-ters. In Proceedings of the 6th USENIX Symposium on Operating Systems Design & Implementation (OSDI), pages 137-150, 2004.

[7] Helen Sun and Peter Helleri. Oracle Information Architecture: An Architects Guid to Big Data. Oracle, 2012.

[8] Hung chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D. Stott Parker. Map-reduce-merge: simplified relational data processing on large clusters . Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 1029-1040, 2007.

[9] Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma..Reasoning about Record Matching Rulesi. Proceedings of the VLDB Endowment, volume 2 of PVLDB, pages 407418. 2009.