Indexed Map-Reduce Join Algorithm

Mohamed Helmy Khafagy
Computer Science Department
Fayoum University
Egypt
Mhk00@fayoum.edu.eg

Abstract— Map Reduce is used to handle and support massive data sets .rabidly increasing in data size, and big data are imperative today to make an analysis of this data. Map-Reduce gets more helpful information by using two simple functions map and reduce with load balancing, fault tolerance, and high scalability .the most important operation in the analysis process is join. This paper explains new two-way join algorithm called Indexed Map Reduce Join Algorithm that used Index in the large table to Decrease I/O and Shuffling that cause Best performance in Map Reduce Join. Our experimental result shows that using Index-join algorithm has high performance than other algorithms while increasing the data size from 100 million records to 500 million without memory overflow.

Index Terms— Algorithm, Big Data, Hadoop, Index, Join, Map Reduce, Performance

1. INTRODUCTION

The most important issue in researches nowadays is analyzing and processing massive data sets [1]. The map-reduce based system is designed to process and analyze these data sets to obtain more knowledge and useful information in order to support industry and academia researches [1].

Map-reduce is considered as a programming model appears since 2004 by Google [1]; it is used to analyze and help heterogeneous datasets. It is becoming more common in order to simplify the interface, deal with fault tolerance, load balancing [2, 3] and high scalability.

Map-reduce [4] is considered as the simplest programming in order to use two certain functions map function and reduce function these two defined functions are written by the programmer to obtain this task and everything such as dealing with fault tolerance and load balancing that are done by the framework by default. Each record of data that is assigned to map task can be produced as a key value pair then this output are sent to the reduce task to do the major operation assigned by the programmer by gathering the values with the same key to producing final output [4].

Apache Hadoop [5] is considered as an open source framework that was developed by Google. It is used to interact with heterogeneous data, the graduate large number of nodes, automatically dealing with node failures; it is used to distribute data processing, and it is written in java.

Hadoop distributed file system (HDFS) is a file system that is used by Hadoop to save file system Metadata and application data independently that can append and search data in distributed file system [6]. It has two independent servers to store Metadata at name node and application data at data node. It uses replication of data to reserve data rather than using RAID. All file data are stored in more than one node [7].

According to the speedy increase in data size we need to accomplish join operation to find hidden pattern and valuable information [8], also there is essential need to optimize the join operation [9, 10, 11] but on multiple data set map-reduce has some restrictions to perform join operations because map-reduce used Network connection to send whole datasets among nodes in the cluster That may cause performance bottleneck [12].

Many researchers through 30 years in the database area, they are using semi-join and hash tables to join operations on massive data sets [12]. Though, map-reduce is designed to deal with single large dataset as input in order to haven't any data structure and database design like indexes, filters or query execution plan as in the database. In this paper, for the better join performance in Hadoop we are using the index in the large table to minimizing I/O in shuffling and map function. We apply this index join techniques only for two data sets. We list some types of join algorithms in map-reduce and then compare our new algorithm with them.

The rest of this paper is ordered as follow: Section 2

IJSER

characterizes some previously map-reduce join. Sections 3 discusses index join algorithm that is concerned with our work. Section 4 we state our experimental results. In the end, we round off and consider the conclusion and future work.

2. RELATED WORK

The two-ways join algorithms using map function at least to do join operation like broadcast join and reduce side join the two-ways join may using more than one phase and using map function and reduce function as see in Equation 1 . Two-way join algorithms that are used to join two dataset R and L as in equation 1.

$$R(A, B) \bowtie L(B, C) \qquad (1)$$

2.1 Broadcast join

Broadcast join algorithm[13,14] is similar to memory join but small data set must fit in the memory but not completely it is called hash join because it's used hash table .loaded one dataset into memory, streamed over other dataset. We used BJ abbreviation for this algorithm. Figure 1 shows pseudo-code for broadcast join [15].

```
Init ()
    if R not exist in local storage then
        remotely retrieve R
        partition R into p chunks R_1..R_p
        save R_1..R_p to local storage

    if R < a split of L then
        H_R ← build a hash table from R_1..R_p
    else
        H_L1..H_Lp ← initialize p hash tables for L

Map (K: null, V: a record from an L split)
    if H_R exist then
        probe H_R with the join column extracted from V
        for each match r from H_R do
            emit (null, new_record(r, V))
    else
        add V to an H_Li hashing its join column

Close ()
    if H_R not exist then
        for each non-empty H_Li do
            load R_i in memory
            for each record r in R_i do
                probe H_Li with r's join column
                for each match l from H_Li do
                    emit (null, new_record(r, l))
```

Figure. 1 Show pseudo-code for broadcast join [15].

Broadcast join implementation:
If R fits into memory and R << S
Distribute R to all nodes
Map over S loads R in memory for each mapper and hashed by join key
Look up join key in R For every tuple in S,
There are no reducers, unless for regrouping or resorting tuples.
Broadcast join Decrease I/O time by avoiding shuffling data and by avoiding using reduce phase. However, it is Sensitive to data skew. The main problem in the broadcast join is dealing with a small table and if the size of the table more than memory size can cause memory overflow. We show in Figure 2 the architecture of broadcast join

.

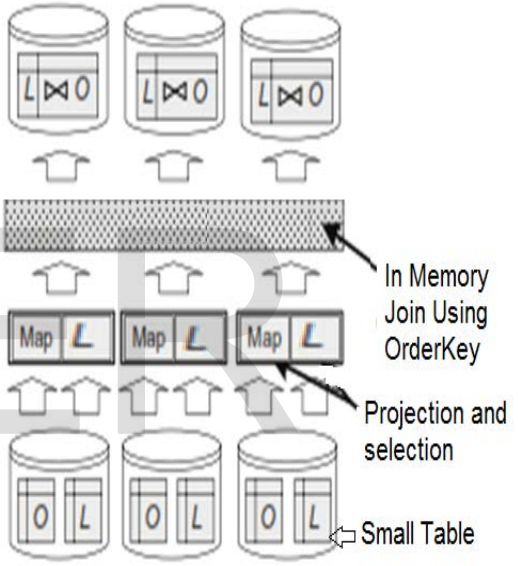

Figure 2 the architecture of broadcast join.

2.2 B. Reduce side join

Reduce side join has no restrictions on the size of your datasets, it can join abundant data sets jointly at once as you want and used to join massive large datasets that are being joined by a foreign key[13]. Map function is ready to join operation to eject the join key as intermediate key/value, and map with both data sets to tag each record with a table name and then outputs a list of tagged keys/ values pairs to see where the record comes from. Hash partitioned function is used to distribute, arrange and combine the output (intermediate key/value) of the map function and distributes them to reduce all records with the same join key and various tags are fit to the same reducer and does the cross product to this records to join results. Figure 3 presents a full phase for reduce side join using Query 1 and Figure 4 Pseudocode for

reduce side join [13] we used the abbreviation RSJ for this algorithm.

```
SELECT l_orderkey, o_shippriority,
FROM orders, lineitem
WHERE l_orderkey = o_orderkey
AND o_custkey IN [X]
AND o_orderdate > [Y]
```

Query 1

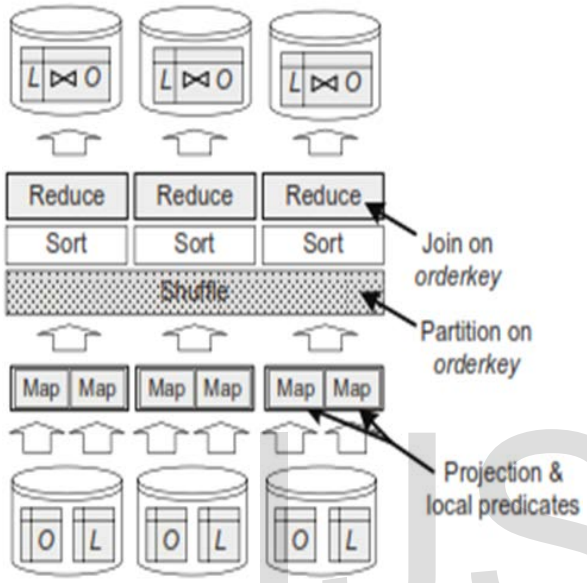

Figure. 3. Full task for reduce side join

```
Map (K: null, V from R or L)
    Tag = bit from name of R or L;
    emit (Key, pair(V,Tag));

Reduce (K': join key, LV: list of V with key K')
create buffers Br and Bl for R and L;
    for t in LV do
        add t.v to  Br or Bl by t.Tag;
    for r in Br do
        for l in Bl do
         emit (null, tuple(r.V,l.V));
```

Figure .4. Pseudo code for reduce side join [13]

Reduce side join is Simplest to implement. And Reduce side join can use any datasets size with no restrictions. Also, it has an extreme time-consuming in order to contain an additional phase to transfer the data from one phase to another phase over the network. Reduce-Side join technique may cause a network bottleneck because of shuffling both datasets over the network. And also it is Sensitive to the data skew

3. INDEX -JOIN MAP-REDUCE

Index Join build and use index in the large table to enhance the performance of join operation as following:

First Map function prepares join operation to emit the join key and index from the large table and at the same time another map make a selection and projection for the second table. Hash partitioned function is used to partition, merge the output (intermediate key/value) of the map function and distributes all to reduces. All records with the same join key and different tag are fit to the same reducer and perform cross product to this records to join results. In the final phase, there is a map function complete missing data from a large table using the index. Figure 5 shows a full phase for Index Join using Query 1

For Example when we want to join Order Table (OID, CID, and TYPE) with Customer Table (CID, NAME) using Join Key CID

1. First, when loading data to HDFS index generated and stored in the Order Table.

2. Second, a map function runs on Order Table to make a selection and projection for Join Key (CID) and Index. At the same Time, another map function runs on Customer Table.

3. Third, Hash partitioned function is used to partition, merge the output of the map function and distributes all to reduces. All records with the same join key CID fit to the same reducer and perform cross product to this records to join results include the index column.

4. Finally, Using Index in Order Table a map function run on Order Table to complete the missing information TYPE, OID

Figure 6 shows how to join between Order Table and Customer Table.
Index Join implementation:
Create Index in R
Map over R---→R (index, Join key) and map over S
Distribute R and s to all nodes using hash partition
Reduce-→ RxS
MAP OVER R→(TYPE, OID)

IJSER

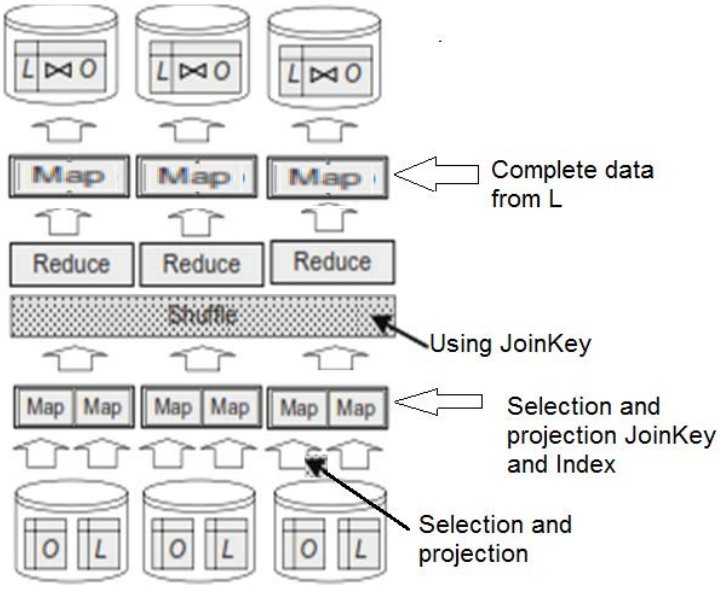

Figure 5 full phases for Index Join

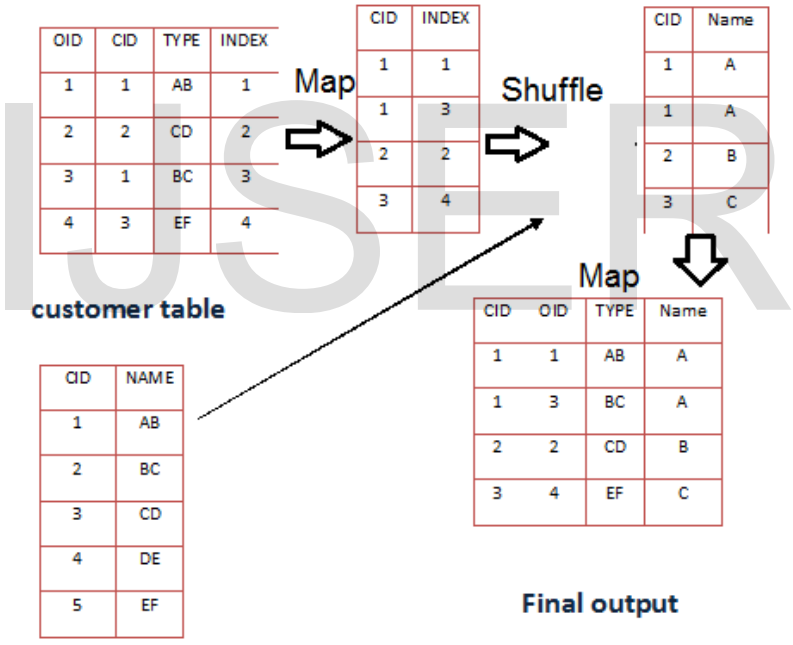

Figure 6 join between Order Table and Customer Table.

Index join has no limitation on the size of your datasets, it can join any size of data sets together at once by a foreign key.it reduce I/O and Decreased by adding index on Large table, so by using this index it can reducing I/O in shuffling and cause enhance in total join time.

4. Experimental Results

4.1 Hardware

We present experimental results of our implementation. We have 3 cluster machines one of them master node (name-node), and two other are slave node (data node). Cluster configuration consists of Intel Core I5 2.4 GHz processor, 4 GB memory for every node, 500 GB SATA disk

and operating system Ubuntu 13.14 Linux with Apache hadoop release 1.2.1.

4.2 Dataset

We use TPC-H benchmark [16] dataset to evaluate our implementation with original Hadoop. We use two table customers and orders to join according to join key where O_CUSTKEY= C_CUSTKEY

We apply three experimental results varying in data size:

1. Experiment 1: we use 100 million records in order table joined by 200 million records in the customer table.
2. Experiment 2: we use 150 million records in order table joined by 300 million records in the customer table.
3. Experiment 3: we use 200 million records in order table joined by 400 million records in the customer table.
4. Experiment 4: we use 250 million records in order table joined by 500 million records in the customer table.

4.3 Results

Table 1 show the Comparison result between broadcast join - reduce side join and Index Join in different data size

Table 1 Comparison between broadcast join - reduce side join and Index Join

Comparison	100 & 200 m	150 &300 million	200& 400 million	250& 500 million
broadcast join	673	Memory OverFlow	Memory OverFlow	Memory OverFlow
reduce side join	801	1244	1540	2320
Index Join	711	1066	1340	2018

Figure 7 show the Comparison result between broadcast join - reduce side join and Index Join with 100 Million Records in order table joined by 200 million records in customer table.and the result show that broadcast join has the best performance because that their join done in the memory also the result of Index join is better than reduce side join with the small difference between it and broadcast join

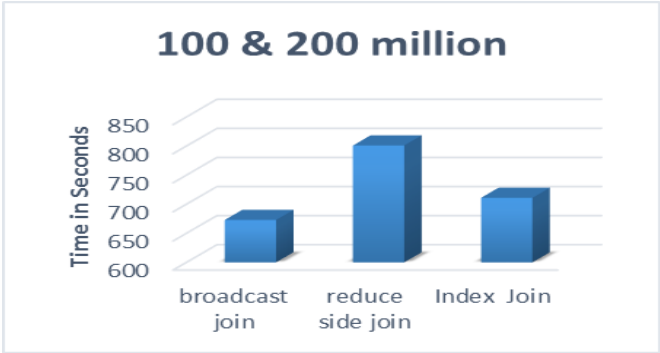

Figure 7 100x200 million records

Figure 8 show the Comparison result between broadcast join - reduce side join and Index Join with 150 Million Records in order table joined by 300 million records in customer table.and the result show that broadcast join cause memory overflow then it cannot complete the join, but Index join still has better performance rather than reduce side join

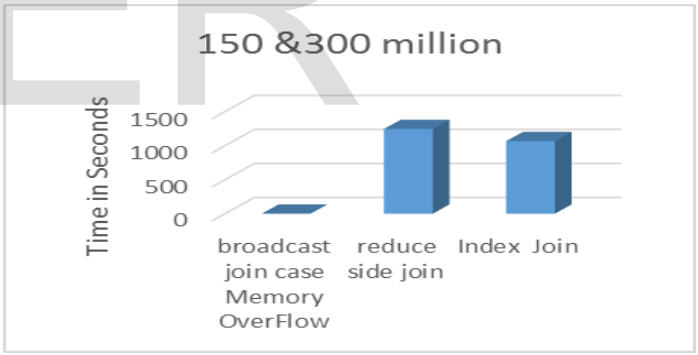

Figure 8 150x300 million records

Figure 9 show the Comparison result between broadcast join - reduce side join and Index Join with 200 Million Records in order table joined by 400 million records in customer table.and the result show that broadcast join cause memory overflow then it cannot complete the join, but Index join still has better performance rather than reduce side join

IJSER

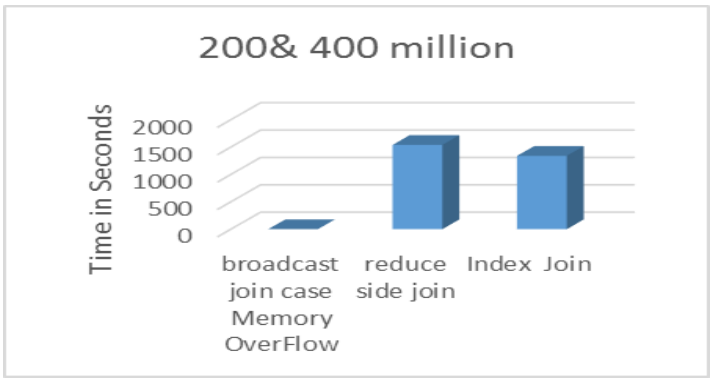

Figure 9 200x400 million records

Figure 10 show the Comparison result between broadcast join - reduce side join and Index Join with 250 Million Records in order table joined by 500 million records in customer table.and the result show that broadcast join cause memory overflow then it cannot complete the join, but Index join still has better performance rather than reduce side join

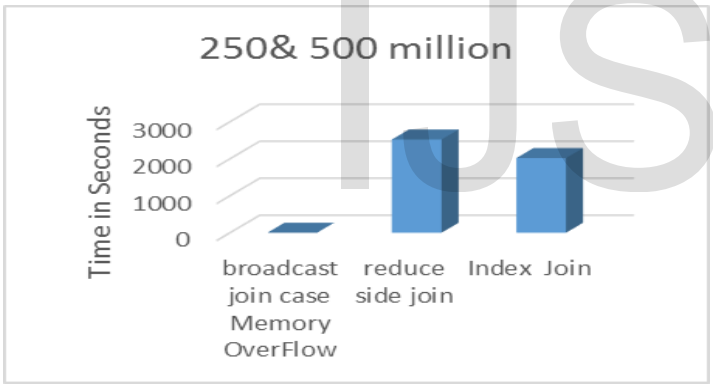

Figure 10 250x500 million records

Figure 11 show that Index join can run with best performance even with the increase of the size of dataset.

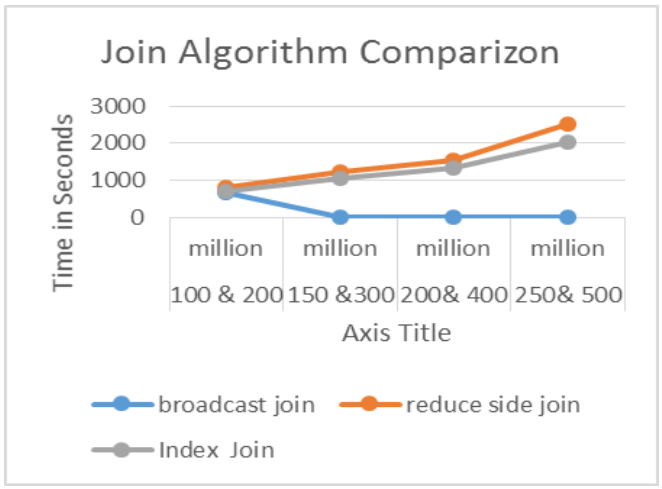

Figure 11 Comparison between join Algorithms in different dataset size

5. Conclusion and Future Work

This work show new proposed join algorithm Index join that it has no limitation on the size of your datasets, it can join any size of data sets together at once by a foreign key.it reduce I/O and Decrease by adding index on Large table, so by using this index it can reducing I/O in shuffling and cause enhance in total join time. . We run new join algorithms with various data size to see the performance while increasing in data size. Our experimental result shows that our algorithm used the index to get high performance, and it is being the best one in running time than two others and memory size not affect in cost.

In the future work, we want to implement the join algorithms using several datasets benchmarks and increasing number of nodes to show performance and used hash semi-join algorithm idea and enhanced in this algorithm to run multi-way join.

References

[1] Dean, J., & Ghemawat Mapreduce: Simplified data Processing on large clusters. In OSDI: Proceedings of the 6th symposium on operating systems design, San Francisco, USA (pp. 137– 150) 2004.

[2] Ebada Sarhan, Atif Ghalwash,Mohamed Khafagy ,Queue Weighting Load-Balancing Technique

for Database Replication in Dynamic Content Web Sites ",APPLIED COMPUTER SCIENCE (ACS'09) University of Genova, Genova, Italy, 2009, Pages 50-55

[3] Ahmed M Wahdan Hesham, A. Hefny, Mohamed Helmy Khafagy," Comparative Study Load Balance Algorithms for Map Reduce Environment" International Journal of Applied Information Systems,2014, Issues 7(11),pp 41-50.

[4] Kumar, A., et al. (2013). "Verification and Validation of MapReduce Program Model for Parallel K-Means algorithm on Hadoop Cluster." International Journal of Computer Applications 72(8): 48-55.

[5] Stefan Richter, Jorge-Arnulfo Quian e-Ruiz, Stefan Schuh,Jens Dittrich: Towards Zero-Overhead Adaptive Indexing in Hadoop.http://infosys.cs.uni-saarland.de, rXiv:1212.3480v1 [cs.DB] 14 Dec 2012.

[6] Haytham Al Feel, Mohamed Khafagy, Search content via Cloud Storage System. International Journal of ComputerScience Issues (IJCSI)bVolume 8 Issue 6, 2011.

[7] E Sarhan, A Ghalwash, M Khafagy,"Agent-based replication for scaling back-end databases of dynamic content web sites", Proceedings of the 12th WSEAS international conference on Computers,2008 pp 857-862

[8] Lee, T., et al. (2012). Join processing using Bloom filter in MapReduce. Proceedings of the 2012 ACM Research in Applied Computation Symposium, ACM.

[9] Mina Samir Shenouda, Mohamed Helmy Khafagy, Samah Ahmed Senbel," JOMR: Multi-join Optimizer Technique to Enhance Map-Reduce Job", The 9th International Conference on INFOrmatics and Systems (INFOS2014),2014 pp 80-86

[10] FR Sayed, M. H. Khafagy," SQL TO Flink Translator",IJCSI International Journal of Computer Science Issues 12 (1), 2015,169:174.

[11] Marwah N Abdullah, Mohamed H. Khafagy," HOME: HiveQL Optimization in Multi-Session Environment ", 5th European Conference of Computer Science (ECCS '14),2014, 80:89

[12] Zhang, C., et al. (2013). "Efficient processing distributed joins with Bloom filter using MapReduce." Int J Grid Distrib Comput 6(3): 43-58.

[13] Pigul, A.: 'Comparative Study Parallel Join Algorithms for MapReduce environment'2013

[14] VIKAS JADHAV1, J.A., SUNIL DORWANI2: 'JOIN ALGORITHMS USING MAPREDUCE: A SURVEY', International Conference on Electrical Engineering and Computer Science, 21-April-2013

[15] Blanas, S., Patel, J.M., Ercegovac, V., Rao, J., Shekita, E.J., and Tian, Y.: 'A comparison of join algorithms for log processing in MapReduce'. Proc. Proceedings of the 2010 ACM SIGMOD International Conference on Management of data 2010 pp. Pages

[16] www.tpc.org

IJSER