Science Arena Publications
Specialty Journal of Electronic and Computer Sciences
Available online at www.sciarena.com
2016, Vol, 2 (1): 9-14

Providing an improved method for the implementation of Hash Join algorithm within the framework of Map Reduce

Elham Ghaffari[1], Ebrahim Azhdari Poor[2]
1-Computer Software Engineering M.Sc.Department of Computer Engineering, Shiraz Branch, Islamic Azad University, Marvdasht, Iran, Azhdari.ebrahim@gmail.com
2- Computer Software Engineering M.Sc.Department of Computer Engineering, Shiraz Branch, Islamic Azad University, Marvdasht, Iran, Elhamghaffari88@yahoo.com

Abstract: Today, one of the most important concerns of information systems, is the processing of massive database queries. Parallel processing system of Google called Map Reduce and open source version of it called hadoop have the ability to work on multiple concurrent execution processing systems. Despite the relatively high performance still it needs to improve the processing method when dealing with the processing of the giant databases. SQL is a famous and powerful database that can store large amounts of data, maintain and make access to the information required for the processing. The database, contain an order called Hash Join that is one of the ways of implementation of join between the tables of a database. In this study, a method is suggested for executing Hash Join in Map Reduce that increases the speed of the join between the tables and the outputs. By comparing this technique with the usual implementation of Hash Join on Map Reduce system, it can be seen that approximately 30 percent improvement is achieved through the proposed approach in running the system.
Keywords: distributed system of Map reduce - hadoop Software - query processing - SQL Database, hash join

1. Introduction

Obviously, one of the most time-consuming phases in a database system is query evaluation engine particularly when there are high volumes of data. Rapid and effective assessment of information queries in a space which is faced with an explosion of information, increases system performance. The use of higher processing power and exploitation of the parallel processing is one of the primary mechanisms. Although many efforts have been done to parallelize query evaluation, the investigation is still open in this regard. But when the volume of data increases, For example, when dealing with the data on the volume of Tera or Peta, the issue is made more complex. But with such massive amounts of data, we are needed to evaluate the query algorithms from different perspectives. Query algorithms, are extremely important in the database management and can help connect the tables by adopting special conditions.

2. A review of literature

An article titled Google's programming model in Map reduce was conducted in [1]. Then it was revised in article [2] which examined the model and provided suggestions for improving it. However, despite the usefulness of this article (in terms of programming model for processing actions such as join analysis and structural use of it) it has not mentioned anything about improving processes in the structure. An article is offered entitled as an optimized framework for queries [3] of Map Reduce. In this paper, a framework is suggested for effectively optimizing semi SQL queries of Map Reduce. It is a new query based on algebra, and it uses a small number of high level physical operators that are independent from existing systems of Map Reduce, such as hadoop. Algebra used in this article, and therefore its processes are relatively simple and effective and this is its strength. But due to the small number of these operators query times are higher and a greater number of machines are distributed and involved in implementing the join. An article entitled Hive: a

warehousing solution over a map-reduce framework, introduces Hive. Hive is a structure that improves query processing in the structure of hadoop and Map Reduce. This article is a little weak from the point of view of the amount of computation and processing [5]. An article entitled Processing Theta-Joins using Map Reduce evaluates the issue of mapping join conditions in the Map and Reduce functions in the structure of Map Reduce [6]. More links in this article describe how to apply mapping in the Map Reduce and improving the query performance is not considered much. Paper [7] introduces a new Translator which is better than the Hive and Pig. Its most important advantage over Hive or Pig is that it can translate complex queries which have correlations between queries. But this method is not a perfect applied database system and does not support table or loading data. It only supports a subset of the features of SELECT query in SQL. There are many things that try to improve queries on Map Reduce and have been focused on bilateral links and it is left to the reader to develop multilateral links. Paper [8] reviews Broadcast Join algorithm (broadcast link) which is a part of this idea. The study deals with bilateral and multilateral Joins. This case is powerful from the perspective of bilateral and multilateral joins. The disadvantage of it is a long processing time than expected. Also in the article [9] the results of broadcast join's algorithm is used for matrix factorization. The strength of this paper is the use of a join method in a useful operation and its weakness is the lack of improvement of broadcast joints' weaknesses.

In another article [10] the researcher suggested bloom filter to process joins on Map Reduce. This filter filters most of the results and records of temporary extension when joining and during the final processing, it doesn't send them to distributed processing systems. The paper also proposed a method by processing the input data set obtained on the basis of their cost which ultimately enhances the process and reduces the response time. This is article's strengths and its weaknesses is the need to an additional processing in order to perform filtering. Although it may not take a very long time, but it allocates the resources. Article [11] suggests procedures for implementing certain types of joins in the databases on the distributed platform so that the joins may have higher efficacy and lower response time. These include policies based on asymmetric joins. The idea has been proposed as a result of a skew-join in which commonly used join keys are processed from the values of less used key. Separation of high used keys from the low used keys, is a relatively effective suggestion in this article. Also reducing the impact of its implementation on some cases due to the retaining of some additional information for different amounts of keys is the weakness of the article. Another weakness is also not possible to perform it for processing symmetrical links. Another weakness is also the lack of possibility to perform processing of symmetrical joins. In [12] distribution algorithms are used based on Map reduce framework. In this study, a distributed algorithm based on model Map Reduce is used to extract the gamma categories, as a sort of dense subgraphs.

3. The suggested method

Hash joins are an example of the join algorithms in the implementation of relational database management systems. This algorithm links any specified amount of joint features and finds a collection of tuples in each relation that has value. Hash join needs a join proposition on condition of equality, which compares the values of two table with operator of "is equal to" (=). Hash joins can be used for to actions that deal with set matching operations. This algorithm has two phases, namely the construction and probe. In the first phase of a hash table based on a hash function is used for smaller relationships. In the second phase of a larger dynamic relationship, referring to the hash table of first relation, values associated with each row, are quickly found and appear in the final output. In order to improve SQL queries and Hash Join algorithm to the Map Reduce systems, we need to provide a solution that would reduce the number of disk memory's references. This algorithm must be applicable in a distributed and parallel way. To run A∞B, two hash tables are created in the main memory; one for A and the other for B. The hash tables are initially empty. At any moment, one of the multiples of A and B are processed. For processing a multiple of A, hash B table is searched to find those multiples of B which match the multiples of A. A and adapted multiples of B are displayed immediately as outputs. After that, multiples of A are added to A hash table in order to adapt to multiples of B that have not been processed yet. Algorithm is ended when all multiples of A and B are processed. If the hash table grows in a way that could not fit in the main memory, it will need to perform specific actions. To avoid this situation, hybrid hashing algorithms and partitioning schema are used.

The use if this technique allows the preliminary results of a query be provided soon. It also provides the possibility of using parallelism; As a result, the overall query response time is reduced in distributed systems

such as systems based on Map Reduce. The use of two hash tables in both relationships involved in hash joins has an advantage; if both are about the same size, it would be an efficient algorithm. Another advantage is its rapid implementation. The Weakness of this method is in its implementation. But may require additional processing or create a new job because of two flaws its hash table. But it may require additional processing or create a new task because of the existence of two hash tables; these are the flaws of this method. Of course, both these disadvantages are negligible comparing with the advantages obtained because the system's performance is increased as a whole.

4. Implementation

The basis of simulation and evaluation of methods are present in many of the distributed networks. So in this study, hadoop distributed simulation environment is used to evaluate the proposed approach. Simulation software that is used in this research, is hadoop simulation software under Eclipse. The software is used to simulate and evaluate the operating system and is a very reliable and efficient system which is accredited by the universities and the scientific community [13].

To demonstrate the applicability of the proposed framework for evaluating energy efficiency, studies have been conducted on alternatives in a message encrypted format. The table describes the parameters used in the simulation.

Table 1 shows that the average cost per node in the Map Reduce indicates the consumption of different encryption records. As expected, it is seen that using Java format based on object-orientation, results in energy transfer and consumed energy be almost 9 times more than the energy consumption in the encryption approach.

Table 1. Simulation parameters

row	Parameter	Value
1	The number of worker nodes	10-50
2	Reading from disk	66584576 (Bytes/second)
3	Writing on the disk	61027123 (Bytes/second)
4	HDFS writing speed	46137344 (Bytes/second)
5	Each node of the working memory	100 MB
6	Speed of network transmission	44040192 (Bytes/second)
7	Size of memory blocks	67108864 (Bytes)
8	The maximum number of mappings	5-35

9	The maximum number of reducing	5-35

On the other hand the use of compressed binary format of Java encryption (with a predetermined isolated pattern of Java) as custom encrypted format, consumes energy. As described in the previous sections, custom records have the highest levels of energy efficiency, but many of them provide little flexibility and performance. While Java is a text that represents the other end of the spectrum. The use of Java has many benefits but Verbose increases the energy costs during communications (Almost 9 times to the energy formats based on order 1). However, the use of binary Java is a compromise between text Java and custom formats. The results show the simulation framework as proposed by the method and the amount of its usefulness.

5. Performance evaluation

In this section we review the results of the implementation of the proposed method for SQL hash join on Map Reduce system. For this segment, we needed to run a query. For the first query namely Q1, the following orders were placed.

Q1:

```
SELECT a.FirstName, a.LastName, b.CreditCardID

FROM Person.Person AS a

INNER HASH JOIN Sales.PersonCreditCard AS b

ON a.BusinessEntityID = b.BusinessEntityID

ORDER BY a.LastName ASC;
```

In the tests, the number of records of the table is Person 19972 and the number of records of credit card is PersonCreditCard 19118. Given that the number of records of hash link is very close, it is logical that the number of worker nodes be equal to the hash table for these tables. The results are shown in the following. The proposed method is compared with other modes. In One case, the hash join is normally done by all nodes. Another case is when the hash joins are not conducted and a regular join is done using Map Reduce. The increase in the number of worker nods reduces the time of implementation. Of course, this could happen up to a fixed number of worker nodes, and then it had a very little impact on the output and even in some cases it had the opposite effect. The response time would be more due to the more complex Communication and the need to coordinate and exchange.

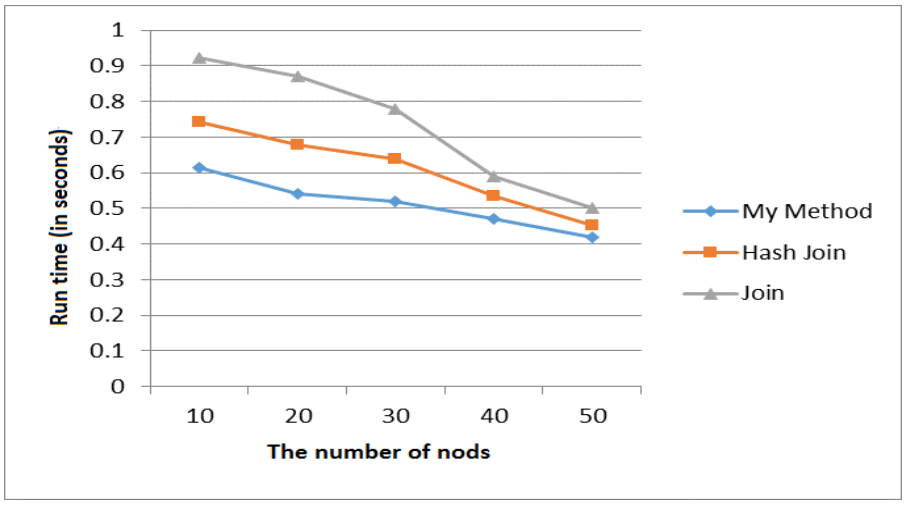


Figure 1: Comparison of the runtime of query Q1

Q2:

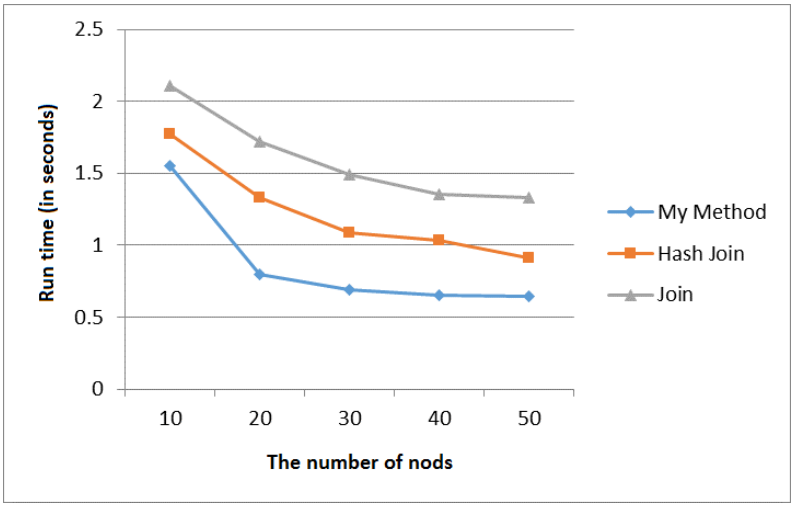
SELECT a.ProductID, a.OrderQty, a.UnitPrice, b.PersonID

FROM Sales. SalesOrderDetail AS a

Left Outer HASH JOIN Sales.Customer AS b

ON a.CustomerID = b.CustomerID

In this query, two tables of Customer and Sales Order Detail are hash joint and their records are 121 317 and 19 820 respectively. The Condition for the join is the equality of CustomerID field from the tables. In the Suggested hash query, it is logical that the number of nodes responsible for making Customer hash tables be less than the other one, since the number of records in the hash tables are so different from each other, and one record is about 6 times the other one. But in this project, as mentioned above, the number of worker nodes are assumed to be equal for both tables. Figure 2 show the implementation of the results of query Q2 and comparisons on Map Reduce.



According to the runtimes in both Figures 1 and 2 it is clear that increasing worker nodes, reduces the run time. But their impact gradually become less and less. Also comparison of the two charts indicate that the rising number of processed records in the second query, increases the processing time. But not as much as the processing of the number of records. This is particularly important due to the use of distributed system of Map Reduce and overlapping of scanning the tables. It is shown in the diagram that the proposed algorithm is much better than hash join and at the end of regular join. As it is shown in the diagram the improvement of proposed method over the usual hash join can at least be about 10 and at most about 25 to 30 percent. In Figure 2, the process is slightly different, the reason is in increased number of records that generally makes processes long. Here the impact of the proposed method is more specific and it is clear throughout the entire graph. Also, up to about 50 to 60 percent difference can be seen. But the average of this improvement can be considered as 30% that in the first case showed 15 to 20%. In general, the gap between the methods is further characterized; In fact, with a higher record number the effect of the performance of algorithms gets higher.

6. Conclusion

The method in this research which was offered in order to run the query of Hash Join in the frame work of Map Reduce, have a faster and better accuracy than the previous algorithms and also affects main memory less than the others. These advantages are resulted from the fact that the proposed method and Hash join algorithm first create two Hash tables in the memory and place their tables in them and therefore, the tables

of the bank are turned into two hash tables. The results indicate that the proposed approach offers more accurate results in less time. The most developed applications in this research mainly focuses on the discovery of records in large and heavy databases; Breaking large tables in the database and turning the tables into Hash and assigning them to any clients made the information processing to be less used from the main memory and thus; the speed increased [14]. Also the difference between this algorithm and previous methods is in that in previous one getting access to the data required so much time and the large tables may not fit in the main memory. The proposed algorithms for distributed systems is appropriate even in small networks or limited systems. The advantage of this algorithm is that with any system and with any number of memory, working with bank and information searching is possible.

References

[1] Hinrichs, Christian, Sebastian Lehnhoff, and Michael Sonnenschein , "*Paving the Royal Road for Complex Systems: On the Influence of Memory on Adaptivity*", International Symposium Selforganization in Complex Systems, The Past, Present, and Future of Synergetics,2013.

[2] ark Kingwell, M.Frank's Motel, "*The Ends of History: Questioning the Stakes of Historical Reason*", Vol.103,2013.

[3] Lindholm, Erik, et al, NVIDIA Tesla: "*A unified graphics and computing architecture*", Micro, IEEE 28.2 ,pp.39-55,2008.

[4] Kent, Mark,Url M. Landau,and Elahe Sharifnejad Toosi, "*Direct sequence CDMA device and method for using the same*", U.S. Patent No.6,173,006,2001.

[5]Broder,Andrei,et al,Graph,"*structure in the web, Computer networks*",33.1,pp.309-320,2000.

[6] Izsvák,Zsuzsanna,Zoltán Ivics,and Ronald H. Plasterk,"*Sleeping Beauty ,a wide host-range transposon vector for genetic transformation in vertebrates*",Journal of molecular biology 302.1, pp.93-102,2000.

[7] De Moraes, Paulo JU, et al, "*Life cycle assessment (LCA) and environmental product declaration (EPD) of an immunological product for boar taint control in male pigs*", Journal of Environmental Assessment Policy and Management 15.01,2013.

[8] Cutting,Doug,and Jan Pedersen,"*Optimization for dynamic inverted index maintenance*" ,Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval,ACM,1989.

[9] Barrett,Jeffrey C.et al," *Haploview: analysis and visualization of LD and haplotype maps*",Bioinformatics 21.2,pp.263-265,2005.

[10] Bakshi,Rashmi,Philippe Cudre-Mauroux,and Marcin Wylot,"*A Comparison of Different Data Structures to Store RDF Data*",2013.

[11] Burbidge,Robert, et al,"*Drug design by machine learning: support vector machines for pharmaceutical data analysis*",Computers & chemistry 26.1,pp.5-14,2001.

[12] Arash Khosraviani, "algorithm Design and implementation to extract distribution of network - categories of Gama from massive scale graphs, Thesis Masters", University of Science and Technology, 2011.

[13] Pawlikowski,Krzysztof,H-DJ Jeong,and J-SR Lee,"*On credibility of simulation studies of telecommunication networks*"Communications Magazine, IEEE 40.1,pp.132-139,2002..

[14] Viglas,Stratis D,"*Write-limited sorts and joins for persistent memory*",Proceedings of the VLDB Endowment 7.5,2014.