SQL TO Flink Translator

Fawzya Ramadan Sayed and Mohamed Helmy Khafagy

Department of Computer Science, Fayoum University, Fayoum, Egypt

Abstract

Flink has become an effective approach to big data analytics in large cluster systems, and with increasing needs of analysis this data Flink has become an effective technique for analysis Big Data and the need of optimized translator for SQL Query is very necessary to build efficient and flexible SQL to Flink because SQL based data processing has limited scalability. So Hive supports queries like called HiveQL however HiveQL offers the same features with SQL,it is still Still a low performance With respect to Flink. A system named SQL TO Flink Translator is developed in this paper to design a new system to define and add SQL Query language to Flink, this translator improve the performance of SQL without any change in Flink framework and give the possibility of execute SQL Query on Flink by generate Flink algorithm that execute SQL Queries . SQL TO Flink Translator has the capability of run SQL when other systems support a low-performance Query and also our system has the best performance when tested in TPC Benchmark.

Keywords: *Flink, Hadoop, MapReduce, HiveQL, SQL, Translator*

.

1. Introduction

Today, we are deep with data that is generated in several scientific areas, on the web, or in social media, or in business applications, or in sensors and mobile devices. For example, Facebook has more than one billion users a day, with more than 618 million of users Producing more than 500 terabytes of new data each day [1].Today available hardware, storage and processing requirements are different from the past when the Traditional data processing and storage approaches designed in the past on old hardware and storage

Traditional data processing and storage technique were resolved when existing hardware, storage and processing requirements were very changed from what it is today. Thus, those techniques are facing many challenges in addressing Big Data requirements.

The keyword "Big Data" [2, 3] Indicate to broad and complex data sets consists of a group of variety of structured and semi-structured which is too large, too fast, or too tough to be managed by traditional techniques.

MapReduce [4] is a programming model that can process and generate massive data and automatically parallelizes through clusters of machines, fixes machine failures, and schedules inter-machine communication to produce an efficient use of the network and disks. The essential idea of

MapReduce is based on two second- order functions, Programming applications were written by this low-level hand coding offers a high flexibility but it increases the difficulty in program debugging [5, 6].

Map and Reduce [2] both functions apply the user function on divisions of their input dataset. All subsets are independently processed by the user defined function MapReduce run above Distributed File System [7–9]. Map function uses a key-value pair and a Reduce function consolidate all intermediate values linked to the same intermediate key. Today Apache Hadoop is the greatest important open-source implementation to MapReduce Moreover this technique has a solution of availably using Replication [10] and has load balance [11, 12] to guarantee the optimum use of resources. Hadoop creates a Hadoop Distributed File System (HDFS) [9] as the global file system running on a cluster [13].

Apache Flink is a project pass through incubation in the apache software foundation that is originated from the stratosphere research project Began in TU Berlin [14]

Newly, numerous SQL- similar to declarative languages and their translators have been constructed and combined with MapReduce to support these languages. These systems contain Pig Latin/Pig [15, 16], SCOPE [17], HiveQL/Hive [18], YSmart[19], S2mart[20], and Qmapper [21]. Practically, these languages play an important role more than programs that have been written in hand-coded.For example, we find that about 95% Hadoop jobs in Facebook are not hand-coded but produced by Hive. These languages and translators have dramatically enhanced the productivity of writing MapReduce programs. Though, we have observed that for many queries, still, auto-generated MapReduce programs for many queries are inefficient compared to hand-optimized programs by expert programmers. We get that ineffectual SQL-to- MapReduce translations have two dangerous problems. First, auto-generated jobs reason that some queries to have long execution times in some operations. Second, for massive sites, the programs generated from inexpert SQL-to-MapReduce translations would create many unnecessary jobs, which is a waste of cluster resources.

Also Apache Flink doesn't support SQL Query language but it has many advantages Fast, which is exploited in

memory data flowing and integrates reiterated processing intensely into the system runtime. That is made the system enormously fast for data intensive and iterative jobs. Reliable and Scalable, this is designed to achieve well when memory is running out. Expressive, which is wonderful writing, type-safe, and its code is negotiable to maintenance in Java or Scala. Easy to use, which it needs a few configuration parameters and convenient with Hadoop, which also support all Hadoop input and output formats and all data types.

This advantages motivate us to Design and develop SQL-to- Flink translator to generate highly optimized Flink programs for SQL queries in Flink programming model.This paper proposes a new Flink translator that translates Some query to Flink algorithm such as Query that contain FILTER ,AGGREGATION, GROUP BY, UNION, JOIN, DISTINCT.

Our paper is organized as follows. Section 2 introduces Literature Survey on a related Hive, Ysmart, S2mart, and Qmapper; Section 3 presents the Proposed System Architecture and Proposed System Methodology; Section 4 displays Experimental Results and a comparison between Flink and Hive, and Section 5 concludes this paper and proposes the future work.

2. Literature Survey

2.1 HIVE

The Apache Hive data warehouse software that is enable querying and handling vast data exist in distributed storage. Hive provides Queries expressed in an SQL-like language called HiveQL. which are compiled into the MapReduce jobs that are performed by Hadoop. Also, we find that HiveQL allows users to create customs MapReduce scripts into queries. Though HiveQL offers the same features in SQL, it is still solid to deal with complex SQL queries into HiveQL. That manual translation is often leading to low performance.

2.2 YSMART

A correlation aware SQL-to-MapReduce translator to enhance complex queries without modification to the MapReduce.Ysmart is a tool built on top of Hadoop that is widely used by Facebook. It has three types of intra query correlations defined based on the key/value pair model of the MapReduce framework[19].YSmart produces optimized MapReduce jobs by applying a set of rules, this rules apply Common MapReduce Framework (CMF) in YSmart, so that it can use the minimal jobs to perform several correlated operations in SQL Query. YSmart can decrease redundant calculations, I/O processes and network transfers compared to current translators.

2.3 Y2MART

A Smart Sql to MapReduce Translator, S2MART transforms the Sql queries into MapReduce jobs by the inclusion of intra-query correlation to minimize redundant operation, the generation and spiral modeled database have been proposed to reduce data transfer cost and network transfer cost SubQuery [20].S2MART applies the concept of views in the database to execute parallelization of big data easy and streamlined.

2.4 QMAPPER

A QMapper is a tool for utilizing query rewriting rules and cost-based and MapReduce flow evaluation, and improved the performance of Hive significantly [21]. Although, Qmapper support complex SQL queries such Exists and Not Exists in where the clause it cannot support All-Any-In-Not In subQuery in where the clause, and also missing features UNION- INTERSECT-MINUS.

2.5 SQL TO FLINK

SQL To Flink Translator is Query language that concentrates on analysis of large-scale datasets, SQL To Flink Translator is a tool built on top of Apache Flink to support SQL Queries. Users send a SQL Query to SQL TO Flink Translator that produces the equivalent code to these query that it can be run on Flink. SQL TO Flink Translator provides high flexibility to deal with FLink without intervention in Flink algorithm that improves the performance of SQL Query language on Big data.

3. SQL to Flink Translator

3.1 System Architecture

The full system architecture is shown in Figure 1 which is divided into four phases. The SQL Query entered into the system from the user.In the first phase The SQL query is given to the Sql Query Parser which assure that SQL is written correctly. At the second phase extract FILTER ,AGGREGATION, GROUP BY, UNION, JOIN, DISTINCT from the given Query , from the input Query extract all columns in the SQL Query and then make scanning on the dataset tables and build a new dataset that contains only new columns and except the non exists columns in the input Query. At the third phases produce the Flink algorithm that execute the input Query At the fourth phases execute the flink algorithm and Finally, return the result to the user.

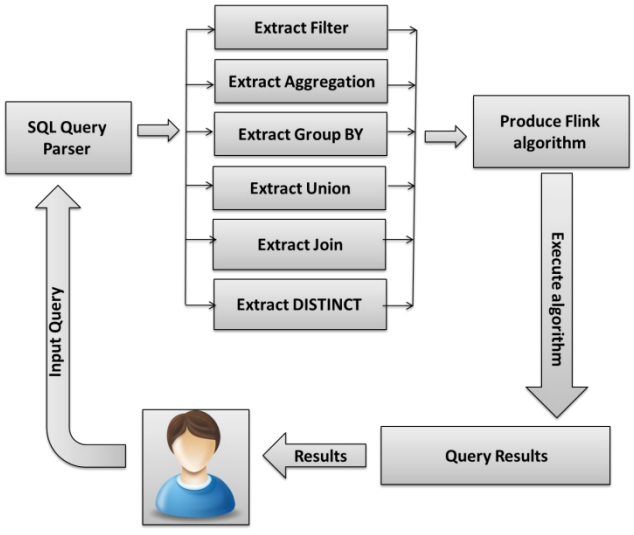

Fig 1. System Architecture

3.2 Methodology

The SQL Query Parser is a module which takes SQL queries as input from the user, parse it to and if it is to have this operation FILTER, AGGREGATION (SUM- MAX-MIN), GROUP BY, UNION, JOIN, and DISTINCT operations we will start to explain in each case of them what our translator will do in each case.

3.2.1 Filter

Filter transformation applies a filter function to every element of a DataSet and keeps only those elements for which the function returns true. When the parser finds Simple condition in the input Query in Figure 2, the proposed system Start to make the code by calling filter function and return from it the result, see Figure 3.

```
SELECT CustomerID, CustomerName, City ,CusTax FROM
Customers WHERE CusTax > 500;
```

Fig 2. SQL Query Contain Filter operation

```
DataSet<Data> customers = getCustomerDataSet(env);
DataSet<Data> suppliers = getSupplierDataSet(env);
DataSet<Data> Result = customers.filter(new FilterFunction<Data>() {
                    @Override
                    public boolean filter(Data cus) {
                        return cus.f3>500; }});
```

Fig 3. Filter Flink code

3.2.2Aggregation

There are some aggregation operations that are commonly used. The Aggregate transformation offers the following built-in aggregation functions

- Aggregation SUM

Aggregate transformation supports the SUM aggregation functions that return the sum of particular field , When the parser finds Aggregation SUM in the input Query In Figure 4, the proposed system Start to make the code by calling aggregate (Aggregations. SUM, Field number) function and return from it the result, see Figure 5.

```
SELECT CustomerID, CustomerName, City ,SUM(CusTax)
FROM Customers WHERE CusTax > 500;
```

Fig 4. SQL Query Contain Aggregation SUM operation

```
DataSet<Data> customers = getCustomerDataSet(env);
DataSet<Data> result = customers.aggregate(Aggregations.SUM,3)
.filter(new FilterFunction<Data>() {
                    @Override
                    public boolean filter(Data cus) {
                        return cus.f3>500; }});
```

Fig 5. Aggregation SUM Flink code

- Aggregation MIN

  Aggregate transformation supports the MIN aggregation functions that return the Minimum number for specific field, when the parser finds Aggregation MIN in the input Query in Figure 6, the proposed system Start to make the code by calling aggregate (Aggregations. MIN, Field number) function and return from it the result, see figure 7.

```
SELECT CustomerID, CustomerName, City ,MIN(CusTax)
FROM Customers WHERE CusTax > 500;
```

Fig 6. SQL Query Contain Aggregation MIN operation

```
DataSet<Data> customers = getCustomerDataSet(env);
DataSet<Data> result = customers.aggregate(Aggregations.MIN,3)
.filter(new FilterFunction<Data>() {
                    @Override
                    public boolean filter(Data cus) {
                        return cus.f3>500; }});
```

Fig 7. Aggregation MIN Flink code

- Aggregation MAX

  Aggregate transformation supports the MAX aggregation functions that return the Maximum number for specific field ,When the parser finds Aggregation MAX in the input Query In Figure 8, the proposed system Start to make the code by calling aggregate(Aggregations.MAX, Field number) function and return from it the result, see figure 9.

```
SELECT CustomerID, CustomerName, City ,MAX(CusTax)
FROM Customers WHERE CusTax > 500;
```

Fig 8. SQL Query Contain Aggregation MAX operation

```
DataSet<Data> customers = getCustomerDataSet(env);
DataSet<Data> result = customers.aggregate(Aggregations.MAX,3)
.filter(new FilterFunction<Data>() {
                    @Override
                    public boolean filter(Data cus) {
                        return cus.f3>500; }});
```

Fig 9. Aggregation MAX Flink code

3.2.3 GROUP BY

The GROUP BY transformation is applied on a grouped Datasets to group the result-set by one or more columns, When the parser finds GROUP BY in the input Query In Figure 10, the proposed system Start to make the code by calling GROUP BY function and return from it the result, see Figure 11.

```
SELECT CustomerID, CustomerName, City ,MAX(CusTax)
FROM Customers WHERE CusTax > 500
GROUP BY CustomerName;
```

Fig 10. SQL Query Contain GROUP BY operation

```
DataSet<Data> customers = getCustomerDataSet(env);
DataSet<Data> result = customers.filter(new FilterFunction<Data>() {
 @Override
public boolean filter(Data cus) {
 return cus.f3>500; }})
. GROUP BY(1). aggregate(Aggregations.MAX,3);
```

Fig 11. GROUP BY Flink code

3.2.4 UNION

The UNION transformation output the union of two Datasets, which have to be of the similar type, when the parser discoveries UNION in the input Query In Figure 12, the proposed system Start to make the code by calling UNION function and return from it the result, see Figure 13.

```
SELECT CustomerID, CustomerName, City , CusTax
FROM Customers
UNION
SELECT SupplierID, SupplierName, City , SupTax
 FROM Suppliers;
```

Fig 12. SQL Query Contain UNION operation

```
DataSet<Data> customers = getCustomerDataSet(env);
DataSet<Data> suppliers = getSupplierDataSet(env);
DataSet<Data> result = customers.union(suppliers);
```

Fig 13. UNION Flink code

3.2.5 JOIN

The Join transformation joins two Datasets to single dataset; when the parser finds JOIN in the input Query in Figure 14, the proposed system Start to make the code by

calling JOIN function and return from it the result, see Figure 15.

```
SELECT SupplierID, SupplierName, City , SupTax
FROM suppliers
JOIN Customers
ON suppliers.CustomerID=Customers.CustomerID;
```

14. SQL Query Contain JOIN operation

```
DataSet<Data> customers = getCustomerDataSet(env);
DataSet<Data> suppliers = getSupplierDataSet(env);
DefaultJoin<Data, Data> result =
customers.join(suppliers).where(1).equalTo(0);
```

Fig 15. JOIN Flink code

3.2.6 DISTINCT

The DISTINCT transformation can be used to return only distinct (different) values, When the parser finds DISTINCT in the input Query In Figure 16, the proposed system Start to make the code by calling DISTINCT function and return from it the result, see Figure 17.

```
SELECT DISTINCT City FROM Customers;
```

Fig 16. SQL Query Contain DISTINCT operation

```
DataSet<Data> customers = getCustomerDataSet(env);
DataSet<Data> result = customers.distinct(2);
```

Fig 17. DISTINCT Flink code

4. Experimental Results

4.1 DATA SET AND QUERIES

We use datasets from TPC-H Benchmark. This benchmark support DSS systems that test large volumes of data to perform queries with a high degree of difficulty and provide answers to critical business questions [19]. We .have split the data set to Variety sizes to run queries on it

4.2 ENVIRONMENT SETUP

We used a Hadoop Single node, Ubuntu 9.0.3 virtual machines, and each one running Java(TM) SE Runtime Environment on Netbeans IDE. We have installed Hadoop version1.2.1 and configured one Namenode and 2 Datanodes. The Namenode and Datanodes have 20 GB of RAM, 4 cores and40GB disk. For the convenience, we installed Hive 0.11.0 on the Hadoop Namenode and Datanodes and also For Flink we installed flink-0.7.0-.incubating-bin-hadoop1

4.3 RESULTS

4.3.1 Flink Translator vs Hive

In Figure 18, we show a comparison between Flink translator and HiveQl. We show that Flink translator .enhance the performance by 55 % in TPCH Query 3

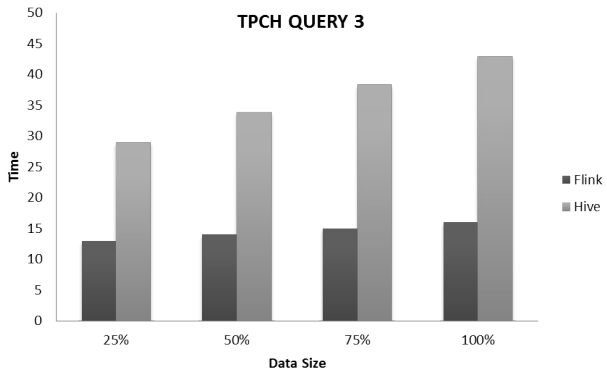

Fig 18.Compare between Flink and Hive IN TPCH Query 3 on Variety size of data

In Figure 19, we show a comparison between Flink translator and HiveQl. We show that Flink translator enhance the performance by 81 % in TPCH Query 10.

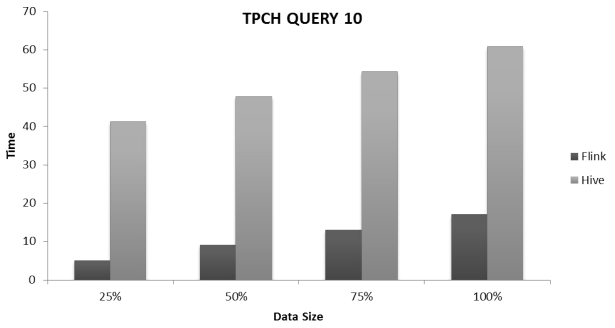

Fig 19.Compare between Flink and Hive in TPCH Query 10 on Variety size of data

4. Conclusions

SQL to Flink Translator can Execution SQL queries on Flink platform. We verified the correctness of our translator by performing various experimental results using different TPCH Queries and we achieved efficiency in all the experimental results. The intended applications for our translator are all data-intensive applications which use big data. In the future, we can build a system with more efficient query optimization for a particular application rather than being generic and enable secure cloud storage for confidentiality and integrity of data. Future work we can also test our technique using different Benchmark [22].

Acknowledgments

The authors would like to thank Prof.Dr. Volker Markl and ALL DIMA Group in TU-Berlin Specially (Kostas, Fabian and Stephean) for providing technical support and advice.

References

[1] Zadrozny, Peter and Kodali, Raghu, "Big Data Analytics Using Splunk ", pp.1-7, Springer, 2013.
[2 K. Grolinger, M. Hayes,W.A. Higashino, A. LHeureux, D. Allison, and M.A.M. Capretz, "Challenges for MapReduce in Big Data", In Proceedings of the IEEE 10th 2014 World Congress on Services (SERVICES 2014), June 27-July 2, 1014, Alaska, USA.
[3] J. Dittrich and J.-A. Quian e-Ruiz,"Efficient Big Data Processing in Hadoop MapReduce "PVLDB, vol.5,no.12,pp.2014{2015,2014.
[4] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters", Commun ACM, vol.51, no.1, pp. 107-113, 2008.
[5] J. Tan, S. Kavulya, R. Gandhi, and P. Narasimhan, "Visual, log-based causal tracing for performance debugging of mapreduce systems", Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on , pp.795,806, 21-25 June 2010.
[6] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin,"MapReduce and parallel DBMSs: friends or foes?", Commun ACM, vol. 53, no. 1, pp.6471, 2010.
[7] Khafagy, M.H. ; Feel, H.T.A., "Distributed Ontology Cloud Storage System", In Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on, pp.48-52,IEEE,2012.
[8] al Feel, H.T.; Khafagy, "OCSS: Ontology Cloud Storage System",In Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on, pp.9-13,November(2011).
[9] Haytham Al Feel, Mohamed Khafagy, "Search content via Cloud Storage System", International Journal of Computer Science, Volume 8 Issue 6, 2011.
[10] Ebada Sarhan, Atif Ghalwash,Mohamed Khafagy. "Agent-Based Replication for Scaling Back-end Databases of Dynamic Content Web Sites", In Proceedings of the 12th WSEAS international conference on Computers WSEAS, Pages 857-862, GREECE 2008.
[11] Ebada Sarhan, Atif Ghalwash,Mohamed Khafagy , "Queue Weighting Load-Balancing Technique for Database Replication in Dynamic Content Web Sites ", In Proceedings of the 9th WSEAS International Conference on APPLIED COMPUTER SCIENCE, Pages 50-55,Genova, Italy, 2009.
[12] Hefny, Hesham A., Mohamed Helmy Khafagy, and Ahmed M. Wahdan. "Comparative Study Load Balance Algorithms for Map Reduce Environment."International Journal of Computer Applications 106.18 (2014): 41-50.
[13] Apache Hadoop, http://hadoop.apache.org, January 2015.
[14] http://flink.apache.org/, January, 2015.
[15] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A.Tomkins, "Pig latin:a not-so-foreign language for data processing,", In Proceedings of the 2008 ACM SIGMOD international conference on Management of data,pp.1099-1110,2008.
[16] A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston,B. Reed, S.Srinivasan, and U. Srivastava, "Building a highlevel dataow system on top of MapReduce:The Piexperience,",PVLDB,vol.2,no.2,pp.1414-1425,2009.
[17] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver,and J.Zhou, "SCOPE: easy and efficient parallel processing of massive data sets,", PVLDB, vol.1, no. 2, pp. 12651276, 2008.
[18] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive - a warehousing solution over a MapReduce framework,"PVLDB, vol. 2, no. 2, pp. 16261629, 2009.
[19] Rubao Lee,Tian Luo,Yin Huai,Fusheng Wang, Yongqiang He,and Xiaodong Zhang,"YSmart: Yet Another SQL-to-MapReduce Translator",In Distributed Computing Systems (ICDCS), 2011 31st International Conference on,pp. 25-36,IEEE, 2011.
[20] Narayan Gowraj, Prasanna Venkatesh Ravi, Mouniga V, and M.R.Sumalatha, "S2MART: Smart Sql to Map-Reduce Translators",In Web Technologies and Applications. LNCS, vol. 7808, pp. 571582, Springer, Berlin (2013).
[21] Yingzhong Xu,and Songlin Hu, "QMapper: A Tool for SQL Optimization on Hive Using Query Rewriting ",In Proceedings of the 22nd international conference on World Wide Web companion, pp. 212221. ACM, Geneva (2013).
[22] TPC Transaction Processing Performance Council, http://www.tpc.org/tpch/.