Performance Comparison of Apache Tez and MapReduce on Hadoop Cluster

Beepana Pokharel, Mahlet Zemedie, Sanuj Kumar
Department of Computer Science
New Mexico State University
Las Cruces, New Mexico 88001, United States

November 29, 2019

Abstract

Apache Hadoop is a good option for processing large data sets that are very complex, and not easy processes by traditional devices. Hadoop has many components that worked together to make the Hadoop ecosystem robust and efficient. Apache Tez is the one of the core execution engine of the Hadoop ecosystem. Apache tez and map reduce use Apache Hive to run tasks in the form of scripts. In our report, we analyze how these two frameworks differ from each other based on some parameters chosen. We compare both the frameworks in the theoretical and empirical ways on the multiple node cluster. In this report we tried to perform the analysis on multionode cluster which installed in our machine.

1 Introduction

Apache Tez is an application framework built on top of YARN supporting batch and interactive queries for terabytes and petabytes of data. It is used as a back-end execution engine over Map Reduce for Hive and Pig[6]. Apache Tez offers a customized execution architecture that allows users to express complex computations as data flow graphs DAGs: The vertices of the graph represent computational processes, with the edges connecting them describing input they receive and output they send out from/to other computational

vertices or data sources/sinks[9]. Tez improves the MapReduce paradigm by dramatically improving its speed while maintaining MapReduce's ability to scale to petabytes of data. Apache Hive and Apache Pig use Apache Tez, as do a growing number of third party data access applications developed for the broader Hadoop ecosystem[10]. This paper organizes including Theorotical analysis, experimental evaluation and Conclusion. This session Theoretical analysis (e.g. Hadoop, Mapreduce and Tez). In experimental evaluation, we describe about comparison of execution time of wordcount and other queries benchmark. In concluding, we discuss performance and trade-off of Apache Tez.

2 Theoretical analysis

2.1 Apache Tez

Distributed processing is the base of hadoop. Hive and Pig relies on MapReduce framework for distributed processing. But MapReduce is Batch Oriented. So it is not suitable for interactive queries. So Apache Tez is alternative for interactive query processing.Tez is prominent over map reduce by using hadoop containers efficiently, multiple reduce phases without map phases and effective use of HDFS. With Tez, The Mapper function reads data from the file system, processes it into Key-Value Pairs which is further stored temporarily on the local disk. These Key-value pairs, grouped on the key values, are sent to the reducers over the network. On nodes where Reducers are to be run, the data is received and is saved on the local disk and waits for the data from all the mappers to arrive. Then, the entire set of values for a key is read into a single reducer, processed and further writes the output which is then further replicated based on the configuration. Some of the benefits are:-

- Amortized development costs : Hive and Pig completely rewrote their engines using the Tez libraries in about 6 months.Apache Tez provides a shared library of ready to use components[12].
- Improved performance : The Distributed parallel execution with in-memory processing boosts Tez's performance to 10 times as compared to MR. A computation in map reduce comprises of multiple map reduce jobs and each map reduce consists of reading from disk, performing

some operations and again writing to disk. But Apache Tez does all computation in memory and only writes the final result to disk. Tez tries to do MR jobs eliminating writing to disk after each computation, extra map reads and queue.

- Enabling future pipelines that leverage multiple engines, to be run more efficiently because of a shared substrate. Tez also maintains a connection pool of pre-warmed containers with shared registry objects. The application can choose to store different kinds of pre-computed information in those shared registry objects so that they can be reused without having to recompute them later on, and this shared set of connections and container-pool resources can run those tasks very fast.

- Container Reuse : A container is a unit of resource allocation on a cluster node. Tez Application Master (AM) runs tasks in containers allocated to it by YARN[12]. When a task completes, the Application Master has an option to return the container to YARN and ask for another container with different capabilities or locality or reuse the same one. Negotiating different containers has overheads for fetching resources and therefore initialising the container process so the same container can be used when the data and locality for the tasks is similar and there is known connection with the locality through edges. When the data is not similar then Tez AM returns the resources gathered by the container back to YARN for getting new and different resource.

- Horizontal Scalability : Apache Tez is based on dividing each tasks into sub-tasks and executing them in parallel across multiple nodes. As you increase the resources i.e the number of nodes(systems) the run time API assigns sub-tasks to each of the new nodes dynamically and thereby improves the performance.

- Resource Elasticity : With Tez, since each task runs in its own container process, the resource allocations are much finer grained. This improves utilization by reducing allocated resources that are idle and thus provide resource elasticity to Tez applications in that they can scale up to utilize as much resources as the cluster can spare to speed up job execution time while gracefully degrading performance but still completing the job when resources are scarce.[12]

2.1.1 Use Cases of Apache Tez

Tez has replaced MapReduce as the underlying engine for many of the batch-oriented Pig and Hive workloads that Yahoo relies on to serve its 1 billion monthly users. Today, 70 of the Hadoop workloads and Yahoo run under Tez [17], use of Apache Spark has also grown, but not nearly as quickly as Tez. This switch from MapReduce to Tez and Spark has been referred as "compute shaping." What compute shaping does is, it allows to make better use of the platform.[17]

2.2 Map Reduce

MapReduce is a framework which helps in writing programs for processing of data in parallel across thousands of machines[16]. MapReduce is divided into two tasks Map and Reduce. Map phase is followed by the Reduce phase. Reduce phase is always not necessary. MapReduce programs are written in different programming and scripting languages.

3 Experimental evaluation

Apache Tez and MapReduce are two frameworks used by Apache Hive in analysis of particular Dataset. These two frameworks have their own merits and demerits.Firstly, we discuss about the dataset used in our experiment. Then we explain the experimental setup used for processing of our dataset. At last we discuss the results of analysis of data after running Apache Hive script on both the frameworks.

3.1 Dataset

We have used three different datasets used for different queries:

- Twitter dataset for using the wordcount program. This dataset is obtained from the https://archive.org website. The size of the this dataset is 734.3 MB.
- The retail dataset is extracted from the cloudera quickstart VM 5.2, also available on github [22], which is one of the example from cloudera dataset. The six tables has a total total size of CSV format 10 MB,

and SQL file size 10MB. It has different tuple counts with the largest tables having 170,000 tuples. The tables name and the attributes for this dataset are mentioned below:

  - categories
    category_id, category_department_id, category_name
  - customers
    customer_id, customer_fname, customer_lname, customer_email, customer_password, customer_street, customer_city, customer_state, customer_zipcode
  - orders
    order_id, order_date, order_customer_id, order_status
  - order_items
    order_item_id, order_item_order_id, order_item_order_date, order_item_product_id, order_item_quantity , order_item_subtotal, order_item_product_
  - departments
    department_id, department_name string, update_date string
  - products
    product_id, product_category_id, product_name , product_description , product_price, product_image

- The sales database is extracted from MariaDB database. It has four tables with different number of tuples and the total size of the data is 160MB of SQL file and 550MB of CSV file . The largest table has 6M tuples and, tables name and the attribute is mentioned below.

  - sales
    SalesID, SalesPersonID, CustomerID, ProductID, Quantity
  - employees
    EmployeeID, FirstName, MiddleInital, LastName
  - customers
    CustomerID, FirstName, MiddleInital, LastName
  - products
    ProductID, Name, Price

We used sqoop to import data from mysql to the hive table. Sqoop is a tool designed to transfer data between Hadoop and relational database servers.

It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases. An example of sqoop to import customers table of retail dataset is

sqoop import –connect jdbc:mysql://localhost:3306/retail_db –username root –password Root@123 –table customers –split-by customer_id –target-dir /user/hive/warehouse/retail_db/customers –hive-import –create-hive-table –hive-table retial_db.customers

3.2 Queries

- On the Twitter dataset, we run a WordCount example. WordCount reads text files and counts how often words occur. The input and the output are text files, each line of which contains a word and the count of how often it occured, separated by a tab. Each mapper takes a line as input and breaks it into words. It then emits a key/value pair of the word and 1. Each reducer sums the counts for each word and emits a single key/value with the word and sum.
  To perform the WordCount we followed some steps:

  1. Creating input file
  2. Create a input directory input in hdfs and Uploading the input file in DFS:
  3. Start all hadoop daemons by `start-dfs.sh && start-yarn.sh`.
  4. Executing WordCount Program: syntax of the executing command will be like this :

     ```
     $hadoop jar <location of hadoop mapreduce examples jar file>
     wordcount <input directory of dfs> <output directory of
     dfs>
     ```

     For example: hadoop jar /hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar wordcount user/hadoop/output/twitter_5_tez4

  5. The output of the program: The output can be downloaded from the browser or we can get it by using this command :

     ```
     $hadoop fs -get /usr/suto/output/part-r-00000 output.txt
     ```

- On the retail dataset, we run point queries, range queries and scanning the entire relation. The queries are listed below

  1. select $*$ from order_items;
  2. select t.customer_id from(select order_customer_id as customer_id, count(order_id) as customer_count from orders group by order_customer_id) as t where t.customer_count$\leq$10;
  3. select product_price from products where product_price between 100 and 1900;
  4. select min(product_price), max(product_price), sum(product_price), avg(product_price), count(product_id) from products;
  5. select o.order_id from orders as o, categories as c, departments as d, products as p, order_items as oi where p.product_category_id=c.category_id and c.category_department_id = d.department_id and d.department_name="Apparel" and p.product_id = oi.order_item_product_id and o.order_id =oi.order_item_order_id;

- On Sales dataset, we run the same query as above to analyze the time with different dataset with different size. We have created a view to limit the tuple size to 1M because our system was not supporting the large amount of tuples.

  create view v_1mil(SalesID, SalesPersonID, CustomerID, ProductID, Quantity) as (select $*$ from sales limit 1000000);

  1. select $*$ from sales;
  2. select s.SalesPersonID from (Select a.SalesPersonID, count(p.ProductID) as countval from v_1mil a, products p where p.ProductID = a.ProductID and p.Price $\geq$100 group by a.SalesPersonID) as s where s.countval $\leq$5;
  3. select Price from products where Price between 100 and 1900;
  4. select count(ProductID), min(Price), max(Price), avg(Price), sum(Price) from products;
  5. Select s.SalesPersonID, p.Name, p.Price from v_1mil s, products p where p.ProductID = s.ProductID;

3.3 Experimental SetUp

The virtual machines in our project are created using VMware Workstation 15 Player. We used three different configuration of the virtual machine. The three different configurations:

(Master x Slaves) :: (1x1),(1x3),(1x5)

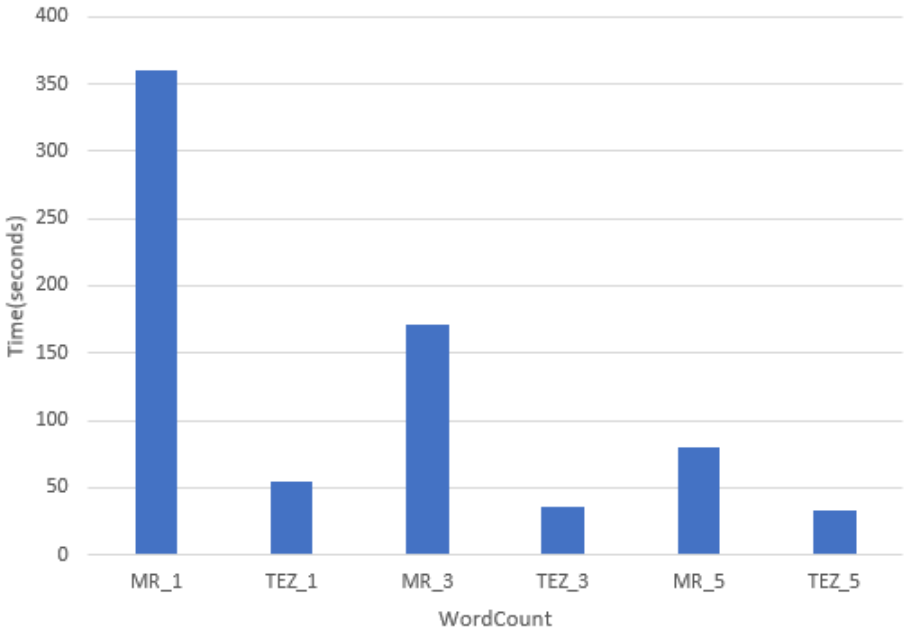
Configuration for Master: Processor(s) - 1, Physical Memory - 2GB , Disk Memory - 20GB
Configuration for Slaves: Processor(s) - 1, Physical Memory - 1GB, Disk Memory - 20GB

3.4 Experimental results

We conducted experiments to evaluate and compare the execution time between MapReduce and Tez. We run two benchmarks to measure different solution of a dataset.

3.4.1 WordCount

The average execution-time of word-count with those of the Twitter files (file size: 734.3 MB) on Hadoop cluster is analyzed above. The example was tested for five times and the average execution time for each frameworks was taken for each configuration. The graph below compares time taken between MapReduce and Tez frameworks.

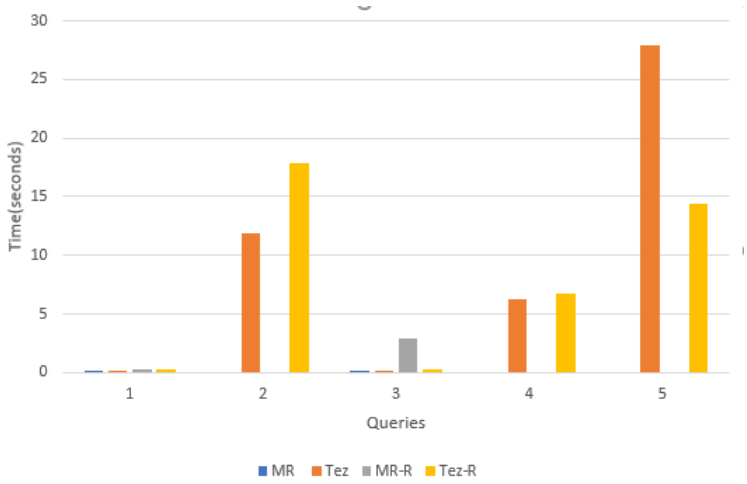


In the figure above, the WordCount example with Tez framework is the best execution-time than MapReduce. Tez framework can reduce an execution-time more than 65 % of MapReduce framework in Hadoop cluster. Also adding more nodes to the cluster increase the performance for both frameworks.

3.4.2 Other Queries

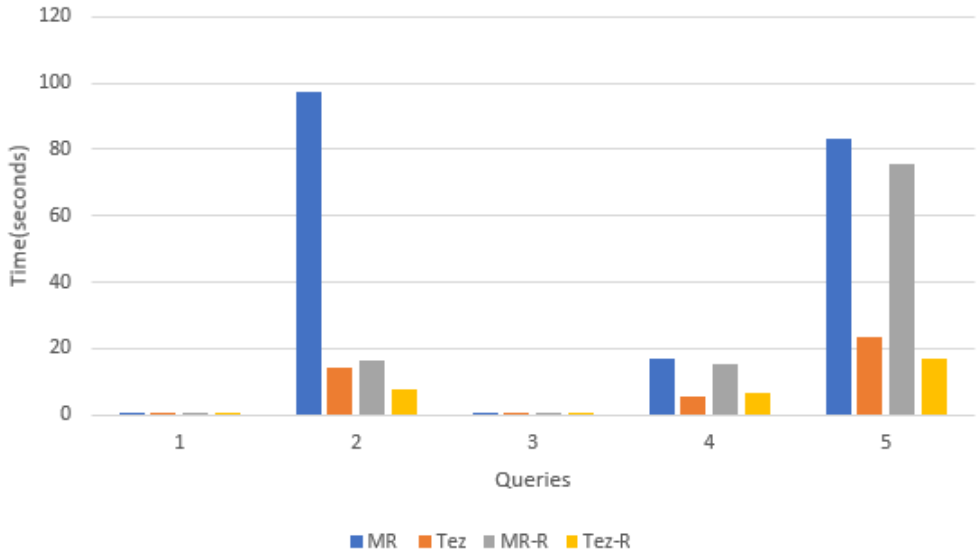
We run each queries listed above five times, and took the average for each configuration for both frameworks to analyze the result.
We run the different five queries listed in above Queries section for both data-set (retail and sales) five times each and took the average time.

- 1 Master and 1 Slave

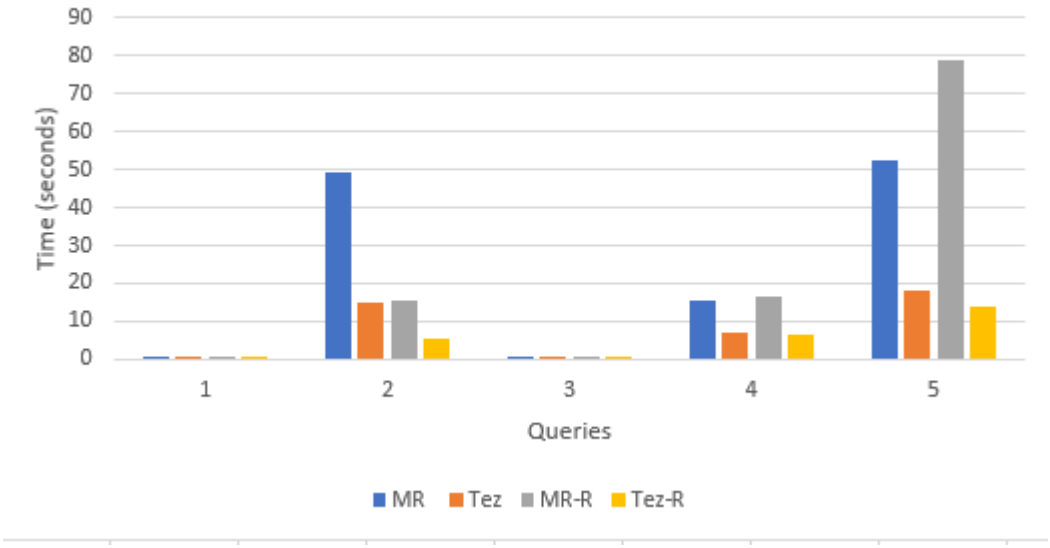


In the above configuration, we were not able to run some queries due to insufficient utilities. There is only one slave node and there is no sufficient resources for it, cannot execute more than 1 mappers. We tried to have same configuration for all the slave nodes for comparison. Therefore changing the resources size only for 1 slave node won't be helpful in analysis. Distributed join algorithms are difficult to express in map-reduce. A regular shuffle join for instance has to process different inputs in the same map task, use tags to be written to disk for distinguishing tables, use secondary sort to get the rows from each table in a predictable order, etc. Tez is a much more natural platform to implement these algorithms.[21]

- 1 Master and 3 Slaves



- 1 Master and 5 Slaves



From both of the graphs above, we can clearly see that running queries

on Apache Tez is much faster than on Mapreduce for both data sets with different configurations.

4 Limitation

- In the configuration of our virtual machines, sql query with distinct keyword is not working on hadoop
- Subqueries in the select clause is not supported with hive.
- We can not run concurrent queries within one session on Tez.[23]

5 Query Parallelization

Currently Tez only supports running one DAG per Tez-Session and each Hive-Session will only use one Tez-Session. The addition of the max_number_of_sessions configuration and associated code to Hue allows for Hue to manage multiple Hive-Sessions. For Tez, every jobs gets converted into mapreduce tasks that run in parallel in the main memory and reduces to sort merge in main memory itself, so there is no point of external sort. Hadoop can execute MapReduce jobs in parallel, and several queries executed on Hive automatically use this parallelism. However, single, complex Hive queries commonly are translated to a number of MapReduce jobs that are executed by default sequencing. Apache Tez supports Broadcast join, which is also known as asymmetric fragment and replicate join. Latest version of Hive uses Cost Based Optimizer (CBO) to increase the Hive query performance. Hive uses a cost-based optimizer to determine the best method for scan and join operations, join order, and aggregate operations. The Apache Hive `EXPLAIN` command use to display the actual execution plan that Hive query engine generates and uses while executing any query in the Hadoop ecosystem. An example can be seen in the image below:

```
OK
Time taken: 0.095 seconds
hive> EXPLAIN Select s.SalesPersonID, p.Name, p.Price from sales s, products p where p.ProductID = s.ProductID;
OK
Plan optimized by CBO.

Vertex dependency in root stage
Map 1 <- Map 2 (BROADCAST_EDGE)

Stage-0
  Fetch Operator
    limit:-1
    Stage-1
      Map 1 vectorized
      File Output Operator [FS_24]
        Select Operator [SEL_23] (rows=1 width=1738139429)
          Output:["_col0","_col1","_col2"]
          Map Join Operator [MAPJOIN_22] (rows=1 width=1738139429)
            Conds:SEL_21._col1=RS_19._col0(Inner),HybridGraceHashJoin:true,Output:["_col0","_col3","_col4"]
          <-Map 2 [BROADCAST_EDGE] vectorized
            BROADCAST [RS_19]
              PartitionCols:_col0
              Select Operator [SEL_18] (rows=1 width=160810)
                Output:["_col0","_col1","_col2"]
                Filter Operator [FIL_17] (rows=1 width=160810)
                  predicate:productid is not null
                  TableScan [TS_3] (rows=1 width=160810)
                    salesdb@products,p,Tbl:COMPLETE,Col:NONE,Output:["productid","name","price"]
          <-Select Operator [SEL_21] (rows=1 width=1580126720)
              Output:["_col0","_col1"]
              Filter Operator [FIL_20] (rows=1 width=1580126720)
                predicate:productid is not null
                TableScan [TS_0] (rows=1 width=1580126720)
                  salesdb@sales,s,Tbl:COMPLETE,Col:NONE,Output:["salespersonid","productid"]

Time taken: 3.17 seconds, Fetched: 31 row(s)
hive>
```

6 Conclusion

In our experiment and results of Hadoop benchmarks, we installed Hadoop cluster in a virtual machine using VMware Workstation 15 Player, and evaluate an execution-time of MapRedue and Apache Tez. We used three different data-set to analyze the performance of each frameworks. We run the Word-Count example and other different type of queries (Range, point, scanning the entire table) on the differnt data-set. In word-count benchmark, Tez framework has a better performance than MapReduce because it does not need to temporary store data in HDFS during process. As the result, word-count output is lower than 100 MB. On the other hand, MapReduce always store data in HDFS that it has a direct effect of an execution-time. However,when the data is very large data size has some effects on a performance of reduce phase in Tez. Tez must keep data in memory that it has a direct effect with a large output of reduce phase. Therefore, it can cause the lower performance of Tez comparing with MapReduce in this case. All of results

can refer to a performance of Tez framework that it not only uses for data structure on Hadoop cluster.

7 Task Division

1. Beepana Pokharel -Gathering Dataset, designing queries
2. Mahlet Zemedie - Documentation and gathering other resources
3. Sanuj Kumar - Installing the systems on VMs.

All of us contributed in preparing the report , installing and configuring VMs and designing presentation slides, and helped each other in understanding the systems.

References

[1] *D*emystify Apache Tez Memory Tuning - Step by Step. (2019, August 17). Retrieved from https://community.cloudera.com/t5/Community-Articles/Demystify-Apache-Tez-Memory-Tuning-Step-by-Step/ta-p/245279.

[2] *H*ortonworks Follow. (2013, November 15). Apache Tez: Accelerating Hadoop Query Processing. Retrieved from https://www.slideshare.net/hortonworks/apache-tez-accelerating-hadoop-query-processing.

[3] *D*ataWorks. (2014, June 17). Hive Tez: A Performance Deep Dive. Retrieved from https://www.slideshare.net/Hadoop_Summit/w-235phall1pandey

[4] Siva, Siva, Hadoop, →, S., Avinash, & Siva Post. (2014, November 28). Apache Tez - Successor of Mapreduce Framework. Retrieved from http://hadooptutorial.info/apache-tez-successor-mapreduce-framework/.

[5] Apache Tez: Accelerating Hadoop Query Processing. Retrived from https://www.slideshare.net/Hadoop_Summit/murhty-saha-june26255pmroom212

[6] Hadoop, A., Inc, & Apache Software Foundation. (2019, September 11). Apache Tez. Retrieved from https://www.cloudera.com/products/open-source/apache-hadoop/apache-tez.html.

[7] The Tragedy of Tez — Big Data Page by Paige. (n.d.). Retrieved from http://bigdatapage.com/tragedy-tez/.

[8] Myers, A. (2018, November 28). Database Terminologies: Partitioning. Retrieved from https://towardsdatascience.com/database-terminologies-partitioning-f91683901716.

[9] Requeno, José Ignacio, Iñigo Gascón, and José Merseguer. "Towards the Performance Analysis of Apache Tez Applications." Companion of the 2018 ACM/SPEC International Conference on Performance Engineering - ICPE 18, 2018. https://doi.org/10.1145/3185768.3186284.

[10] Stop Thinking,Just Do!(n.d.).Retrieved from https://sungsoo.github.io/2014/04/27/comparing-apache-tez-and-microsoft-dryad.html.

[11] Hadoop, Apache, Inc, and Apache Software Foundation. "Apache Tez." Cloudera, September 11, 2019. https://www.cloudera.com/products/open-source/apache-hadoop/apache-tez.html.

[12] "Apache Tez: A Unifying Framework for Modeling and Building ..." Accessed October 18, 2019.http://web.eecs.umich.edu/~mosharaf/Readings/Tez.pdf..

[13] "The Apache Software Foundation Blogging in Action." The Apache Software Foundation Announces Apache™ Tez™ as a Top-Level Project : The Apache Software Foundation Blog. Accessed October 18, 2019. https://blogs.apache.org/foundation/entry/the_apache_software_foundation_announces60.

[14] hjamali52hjamali52 61533 gold badges1010 silver badges1919 bronze badges, Wes FloydWes Floyd 32633 silver badges55 bronze badges, abhijeet dhumalabhijeet dhumal 1, GeekyGeeky 3555 bronze badges, ketankkketankk 1, & Kiran Teja AvvaruKiran Teja Avvaru 1. (1964, December 1). Apache Tez architecture Explanation. Retrieved from https://stackoverflow.com/questions/25521363/apache-tez-architecture-explanation?rq=1

[15] The Cascading 3.0 Query Planner. (2014, September 18). Retrieved from https://www.cascading.org/2014/09/18/the-cascading-3-0-query-planner/.

[16] Maitrey S, Jha CK. Handling Big Data efficiently by using MapReduce technique. In: IEEE international conference on computational intelligence & communication technology (CICT); 2015. p. 703–8.

[17] Yahoo's Massive Hadoop Scale on Display at Dataworks Summit. (2017, June 16). Retrieved from https://www.datanami.com/2017/06/16/yahoos-massive-hadoop-scale-display-dataworks-summit/.

[18] (n.d.). Retrieved from https://ieeexplore.ieee.org/document/8025950.

[19] Raghav, et al. "Commissioning and Decommissioning of Data Node in Hadoop Cluster." AcadGild, 22 Nov. 2018, https://acadgild.com/blog/commissioning-and-decommissioning-of-datanode-in-hadoop.

[20] Pandya, Hardik. "Sqoop: Import Data From MySQL to Hive - DZone Big Data." Dzone.com, 2 Jan. 2018, https://dzone.com/articles/sqoop-import-data-from-mysql-to-hive.

[21] "Apache Software Foundation." Hive on Tez - Apache Hive - Apache Software Foundation, https://cwiki.apache.org/confluence/display/Hive/Hive on Tez.

[22] Dgadiraju. (n.d.). dgadiraju/code. Retrieved from https://github.com/dgadiraju/code/tree/master/hadoop/edw/database.

[23] Support Portal, https://mapr.com/support/s/article/Enable-Hue-to-run-concurrent-query-per-user-in-TEZ?language=en_US.

[24] S, V. (2019, January 11). Apache Hive EXPLAIN Command and Example. Retrieved from http://dwgeek.com/apache-hive-explain-command-example.html/.