Global Illumination for Fun and Profit

Fig. 1. In the Clouds: Vancouver from Cypress Mountain. Note that the teaser may not be wider than the abstract block.

Abstract—Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

Index Terms—Radiosity, global illumination, constant time

1 Introduction

The distributed database system is becoming increasingly pervasive due to the explosive growth of data in science, industrial and life. Meanwhile, many management tools such as Hive, Flink and Vertical are developed to optimize the query, translate traditional query language to execution plan based on the map-reduce framework and dispatch multiple tasks to clusters to perform the data acquisition in parallel.

To maximally leverage the distributed systems, it is crucial for users to understand and evaluate how the query runs across the clusters. The frequently asked questions include "Where does the time go?", "What is the bottleneck of my query?", "Can we improve the performance of the specific query?". Many research work devoted to evaluating and improving the performance of data analytics frameworks, but most of them try to reveal the performance by making high-level statistics about the correlated metrics collected from the accumulated logs or experiment conducted on benchmarks, which cannot be used for the understanding of the special case and provide the details answer for these questions. Specific methods which can look into the query execution process are required.

There are two challenges to facilitate the fine-grained inspection of query executions. **Non-transparent translation** makes it difficult for database users to inspect the query behaviors for a given abstract query. As shown by Figure**, the query issued by users is highly abstract which hides the detailed executed logic on a distributed system. The tools such as Hive can translate the query to a physical execution plan as shown by Figure 3. The execution plans have hundreds to thousands of lines of description, which is difficult for users to build a mental map for the overall execution plan. Existing work tries to bridge the gap between the execution logic and human perception by visualizing the execution plan as directed acyclic graph and allow users to interactively narrow down to any detailed operator as demand. However, the visualization of execution plan is always independent from the visualization of execution process. During the exploration, the users have to switch between multiple views which break the continuity of the plan-execution analysis. **Unexpected behavior of distributed system** also increases he difficulty to understand the model execution process. For instance, the developer find that the same query execution plan run today may be different from that of yesterday. In general, four aspects are considered to affect the performance of clusters: CPU usage, memory usage, network IO and disk status. Existing work studies try to reveal how these metrics related to the system performance or quantify the impact and significance of these features. These studies are conducted based on the observed performance data from the experiment or logs collected from the production environment. One work inspired us is VQA which tries to linkage the resources status to query performance and resource usages. However, these work fail to provide the fine-grained execution traces for users to inspect the reasons of model behavior.

In this work, we develop a visual analytics system called DQSVis (Figure 1) for database users to monitor, understand and diagnose query behavior across the distributed system. The system can be run

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

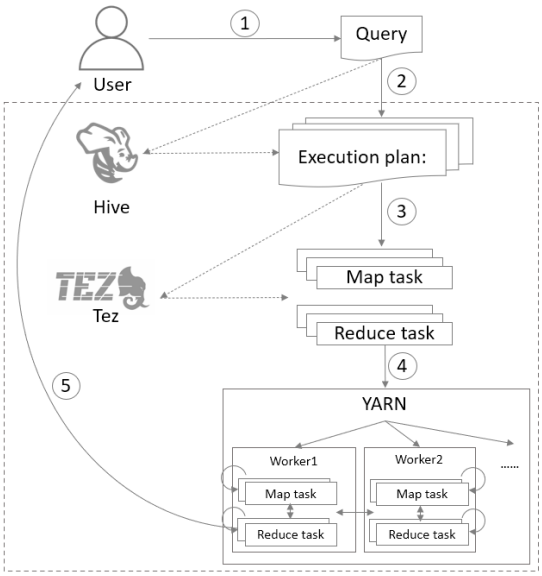

Fig. 2. Overview of Hive2.0 distributed query architecture.

```
Stage-0
  Fetch Operator
    limit:100
    Stage-1
      Reducer 6 vectorized
      File Output Operator [FS_80]
        Limit [LIM_79] (rows=100 width=1612)
          Number of rows:100
          Select Operator [SEL_78] (rows=186008031 width=1612)
            Output:["_col0","_col1","_col2","_col3","_col4"]
          <-Reducer 5 [SIMPLE_EDGE] vectorized
            SHUFFLE [RS_77]
              Select Operator [SEL_76] (rows=186008031 width=1612)
                Output:["_col0","_col1","_col2","_col3","_col4"]
                Group By Operator [GBY_75] (rows=186008031 width=1612)
                  Output:
["_col0","_col1","_col2","_col3","_col4","_col5","_col6","_col7","_col8"],aggregations:['
(VALUE._col0)","count(VALUE._col1)","sum(VALUE._col2)","count(VALUE._col3)","sum
(VALUE._col4)","count(VALUE._col5)","sum(VALUE._col6)","count(VALUE._col7)"],keys:KEY._cc
                  <-Reducer 4 [SIMPLE_EDGE]
                    SHUFFLE [RS_29]
                      PartitionCols:_col0
                      Group By Operator [GBY_28] (rows=372016062 width=1612)
                        Output:
["_col0","_col1","_col2","_col3","_col4","_col5","_col6","_col7","_col8"],aggregations:['
(_col4)","count(_col4)","sum(_col5)","count(_col5)","sum(_col7)","count(_col7)","sum
(_col6)","count(_col6)"],keys:_col18
                        Merge Join Operator [MERGEJOIN_58] (rows=372016062 width=1612)
                          Conds:RS_24._col1=RS_74._col0(Inner),Output:
["_col4","_col5","_col6","_col7","_col18"]
                        <-Map 10 [SIMPLE_EDGE] vectorized
                          SHUFFLE [RS_74]
                            PartitionCols:_col0
                            Select Operator [SEL_73] (rows=533517 width=1094)
                              Output:["_col0","_col1"]
                              Filter Operator [FIL_72] (rows=533517 width=1094)
                                predicate:i_item_sk is not null
                                TableScan [TS_12] (rows=533517 width=1094)
                                  tpcds_text_100@item,item,Tbl:COMPLETE,Col:NONE,Output:
["i_item_sk","i_item_id"]
                        <-Reducer 3 [SIMPLE_EDGE]
                          SHUFFLE [RS_24]
```

Fig. 3. The hive execution plan.

with three modes: 1) monitoring mode: the system runs with the query execution process, collects and visualizes the query status in real-time; 2) simulation mode: the system will replay the execution process with given simulation rate; 3) analysis mode: the system will directly show the final results for users to explore the final results. In the visualization component, we design a temporal DAG (directed acyclic graph) diagram to display the execution plan and execution process dynamically and seamlessly. To enable the scalable visual analysis of the large number of tasks executed on the computing nodes, we implement the compound trace diagram which integrates the point cloud form and progress bar form together to meet the different analysis requirements. The monitoring results are visualized in the monitoring view and linked with the other analysis views through a suit of flexible interactions.

- A framework to systematically analyze the query execution on distributed database by integrating three components: query analyzing component, machine monitoring component and analytic component.
- Well-established visualization front-end to support the interactive investigating, comparing and diagnosing the query process. The system includes a set of novel designs for visualizing the temporal DAG and sequence group.
- Case studies on the analysis of query process performed on the Hive platform.

2 Related Work

2.1 Distributed query analysis

2.2 Visual analytics for distributed systems

2.3 Visualization for temporal data

3 Background

The architecture and terms are instroduced in this section to serve the as basis for the further discussions. The figure 2 demonstrates how a query is processed by a distributed query system. In this case, we use the Hadoop2.0(HIive+Tez architecture) as an example.

When user issues a query (shown as Figure 2(1)) through the interface such as a web UI or SQL terminal, Hive optimizes it and translate it as the detail logic execution plan shown as Figure 2. The logic exaction plan may contains hundreds of lines of description, which usually describe the execution process as a Directed Acyclic Graph(DAG). The DAG includes two types of vertex: map vertex and reduce vertex. Each vertex contains a sequence of logic operators such as filter, aggregate, merge, etc. Moreover, there are edges connecting the verteices. The edges define the data movement between vertecies, the source vertices are the producers and the destinat vertices are the consumers. Tez further generates the physical tasks according to the work flow of vertices and YARN dispatches these tasks on the Hadoop worker machines. The task is the atomic process of the query. All tasks of the a specific vertex are dispatched to multiple machines and processed according to operators of the vertex.

4 System Design

4.1 Requirement analysis

4.2 System overview

5 Visualization Design

5.1 Query execution overview

5.1.1 Execution plan view

5.1.2 Execution progress view

5.2 Task view

5.3 System profiling visualization

5.4 Linkage and interactions

6 Evaluation

6.1 Case study

6.2 Expert interview

7 Conclusion

8 Conclusion

Acknowledgments

The authors wish to thank A, B, and C. This work was supported in part by a grant from XYZ (# 12345-67890).

References

[1] Kitware, Inc. *The Visualization Toolkit User's Guide*, January 2003.

[2] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Computer Graphics*, 21(4):163–169, Aug. 1987. doi: 10.1145/37402.37422

[3] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995. doi: 10.1109/2945.468400