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.

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

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

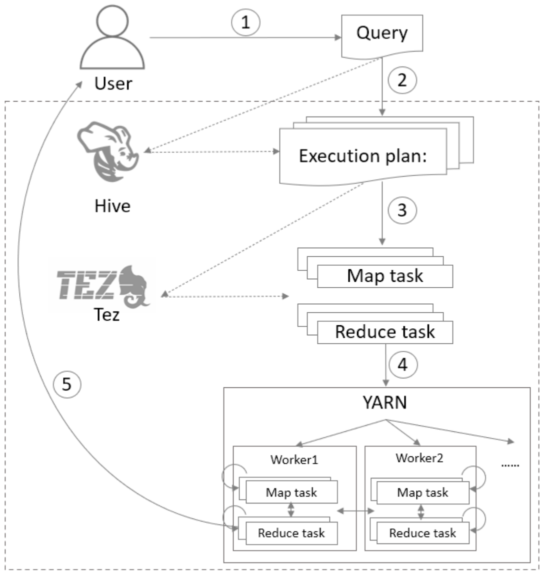
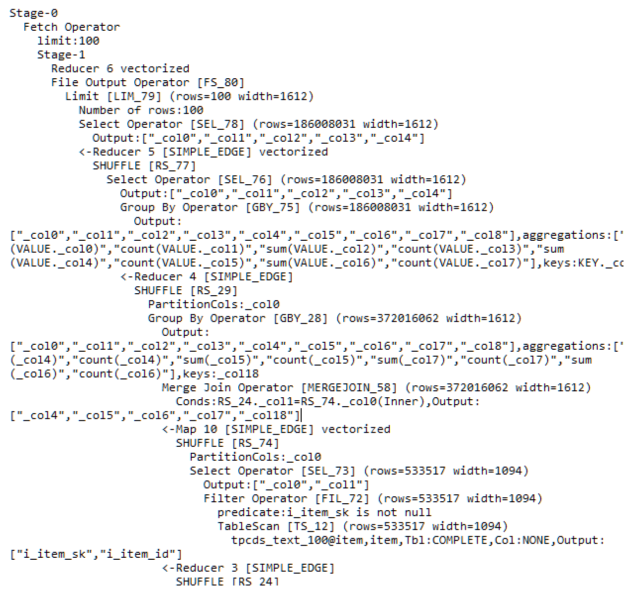


Fig. 2. Overview of Hive2.0 distributed query architecture.



```
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.

- The design and implementation of DQSVIS, a visual analytic system for understanding and analyzing the query execution across clusters.
- 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

3.1 Background of distributed query execution

The architecture and terms are introduced 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.

3.2 Requirement analysis

During the one year of collaboration, we have closely collaborated with three experts in distributed database, who are also the co-authors of this paper.

In the first month of the collaboration, we have held brainstorming to collect the most frequent raised questions when analyzing the distributed query system performance. Based the discussions with domain experts and review of existing literature, we have formulated the following design requirements.

R1 **Understand the general query execution trace and query plan structure.** Before our collaboration, the domain experts have used profiling software (TezVIS, etc) or visualization tools(Tableau, etc) to show the query progress as Ganntt chart and query plan structure as directed graph. However, these two visualizations are always displayed in separated views which require users to switch their focus and break the continuity of exploration.

R2 Understand the query process at the task level.Understand the execution of single task can be helpful to identify the bottleneck of the whole query process.However, visualize the tasks is challenge. First, to visualize the tasks in traditional way(Gantt diagram) need a large canvas. The multiple features such as the size of data and the operators should be visualized for understanding the task. Moreover, the many to many relationship among the tasks also makes it difficult to design clear and **scaleble** visualization.

R3 Provide the visual insight to reason the behaviour and pattern of a specific task.To solely visualize the tasks themselves are not enough to explain the specific pattern of tasks. Many reason about the hardware resource such as the network status, hard disk waiting list is also related to the patterns. Such kinds of information should be vitalized effectively to assist the exploration of query executions.

R4 Support interactive exploration. Other than the visualization designs, a flexible interaction should be implemented for users to

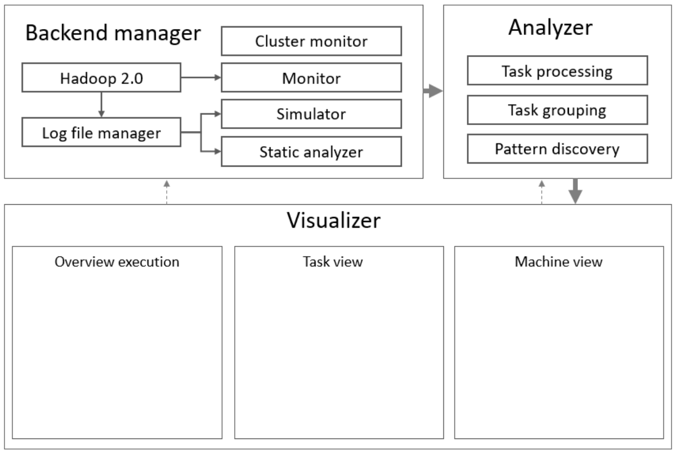


Fig. 4. DQEVis system consists of three major components: backend manager, analyzer and visualizer.

navigate to any point of interest. The linkage among the correlated visual elements are also should be designed to coordinate the information.

3.3 Task analysis

Guided by the aforementioned requirements, we discussed with the domain experts about the visualization form and distilled the following visualization tasks:

T1 **Visualize the execution process and query plan structure effectively.** To guarantee the continuity of exploration(R1), the process and plan structure should be integrated into one visualization view. Several criteria should be considered such as the minimize usage of canvas, minimize the cross of links and provide clear topology structure.

T2 **Effectively visualize the information of tasks.** To facilitate the fine grained exploration of query execution(**R1**, **R2**), the information about the tasks should be visualized, including: the size of data processed by the task; the data-flow among the tasks; the temporal information of task(start the time, end time, time usage, etc) and the corresponding sub-process. Moreover, the abnormal(tasks taking longer time) tasks should be easily observed.

T3 **Visualize the machine status.** Display the machine status such as network status, disk IO pending list, CPU usage and Memory Usage will be useful to investigate the characters of task, and reasoning the patterns of the query execution(**R3**).

T4 **Interaction and linkage.** System should provide the flexible interactions allowing users to switch the focus among the different point of interest, such as a specific time range, a vertex or a group of tasks(**R1**, **R2**, **R3**, **R4**). For example, user may select a vertex and explore if the tasks in this vertex are CPU-bound or I/O-bound. This requires the visualization to show the related tasks when choosing a vertex and highlight the corresponding CPU and dist information simultaneously.

4 System Design

As shown by Figure 4, DQEVis consists of three modules: backend manager, analyzer, and visualizer.

In our current system, we use Hadoop2.0 as the **distributed query engine**. The system can run in three modes: monitoring mode, simulation mode and analytics mode. The monitoring mode directly collect the real-time execution log from Hadoop2.0, then process it and send the result to analyzer. The simulator and log analyzer are linked with log file manager, a module to process and save logs as as structured data form. Log analyzer directly output the struscured data to the next module. The simulator simulatses the execution progress which allow users to adjust the running speed and explore the dynamic query process.

The analyzer collects the data from the backend manager and further process them for visualization.

The visualization module integrates coordinated views to support interactive exploration of and reasoning about the query behaviour at the different perspectives. Execution overview demostrates the execution process in at the vertex level. A new algorithm for the temporal DAG is proposed to visualize the structure and process at the same time. The task view visualize the temporal information of tasks as well as the dataflow relationship among the tasks. The profiling view integrates the task view with the resource status for each machines. A rich set of interactions are also supported to link different views together.

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