Lecture #17: System Analysis (Google Dremel / BigQuery)

15-721 Advanced Database Systems (Spring 2024)
https://15721.courses.cs.cmu.edu/spring2024/
Carnegie Mellon University
Prof. Andy Pavlo

1 Introduction

Google Dremel [1] is an interactive in-situ data processing and analysis tool with the capacity to analyze web-scale datasets. Dremel incorporates excellent architectural design choices that are widely recognized as industry-leading standards, including disaggregated compute and storage, disaggregate memory, in situ data analysis, on-demand serverless compute, columnar storage, etc.

Topics covered throughout this lecture:

- Resource Disaggregation
- Columnar Data Format
- Shuffle-based Distributed Query Execution
- Query Optimizations

2 Resource Disaggregation

Dremel offers on-demand serverless compute by incorporating the concept of decoupling compute, storage, and memory. Disaggregation ensures the scalability of each resource and enhance cost-performance.

2.1 Distributed File System

Dremel utilizes distributed file system to scale out its storage capacity. Adopting the distributed file system brings lots of benefits, including increased reliability, scalability, performance [2], robustness, and elasticity to the system.

2.2 Self-describing

Everything that a database requires to analyze a file is contained within the file itself.

3 Columnar Data Format

Although columnar storage had been recognized prior to the publication of the Dremel paper, it was the Dremel paper that first adopted columnar storage specifically for semi-structured data.

3.1 Nested and Semi-structured Data

Dremel needs to support nested and semi-structured data while ensuring performance. To achieve this, data must be accessed without needing to access all ancestor fields. This necessity introduces the concepts of repetition levels and definition levels. The use of both repetition levels and definition levels enables Dremel to track repeated and optional fields efficiently.

3.2 Vectorized Execution

The columnar data format is inherently suited to block-oriented vectorized processing methodologies, greatly enhancing execution efficiency.

4 Shuffle-based Distributed Query Execution

Dremel's architecture transitioned to centralized scheduling and a shuffle persistence layer. The query coordinator constructs the query plan and forms a DAG. Subsequently, it manages the execution of the query by coordinating with workers provided by the scheduler. Workers are allocated as a flexible pool without a predefined structure. The coordinator dispatches a ready-to-execute local query execution plan to the workers. More shuffle details are shown below.

4.1 Deterministic and Idempotent Tasks

A logical plan can be segmented into multiple stages, each of which can then be broken down into multiple parallel tasks. For each task, the result for running tasks needs to be deterministic and idempotent to ensure correctness when restarts happen.

4.2 In-memory Shuffle

The process of in-memory shuffle is listed below:

1. Coordinator retrieves metadata
    - The coordinator retrieves all metadata needed and binds it with the query plan
    - Prevent all worker nodes requesting data from the distributed file system at the same time
2. Coordinator assigns work
    - The coordinator assigns workers and distributes work to workers
    - Dremel employs shuffle persistent layer to decouple scheduling and execution
3. Workers retrieve data
    - Workers retrieve data from the distributed file system
    - Workers perform the computation
4. Workers send output data
    - Workers send output to shuffle nodes which store the intermediate outputs in the memory
    - Shuffle nodes store data in the memory in hash partitions (Shuffle nodes only spill the data to disk when necessary)
    - During the next stage, data can be read directly from the memory of those shuffle nodes
5. Workers store final output data
    - When all stages are completed, the output data will be written back to the distributed file system

4.3 Straggler

In order to prevent stragglers from impacting overall performance, when a worker's execution time exceeds a certain threshold, the coordinator will execute a redundant task. This mechanism ensures that failed tasks are eventually executed and completed, while ensuring the overall system performance.

4.4 Dynamic Resource Allocation

The coordinator can scale up or scale down the number of workers dynamically.

5 Stratified and Dynamic Query Optimization

Due to the lack of statistics, it is difficult for query optimizers to utilize cost models based on statistics. As a result, Dremel employs a hybrid approach that combines rule-based and cost-based optimization strategies.

5.1 Stratified Approach

Dremel employs the stratified approach that incorporates both rule-based and cost-based optimizer in order to generate a basic physical plan. In cost-based optimizations, only the data with available statistics is used. Cost model estimation can thus be inaccurate, to address this problem, Dremel utilizes adaptive query optimization.

5.2 Dynamic Query Execution

Dremel dynamically optimizes queries, adjusting the query plan before each stage begins based on the performance and results from the previous stage. Below are some examples:

1. Number of workers
    - Change the number of workers for a specific stage
2. Adaptive join strategy: Broadcast v.s. hash join
    - Broadcast join is faster as it does not need to shuffle data
    - Broadcast join can only work when the size is small enough to fit in memory
3. Physical operator implementation
4. Dynamic repartitioning
    - When data is skewed, repartitioning can help to maintain balance
    - Repartitioning happens when the database detects a shuffle partition is getting too full

6 Related Projects

Below are projects that are inspired by or related to Dremel:

1. Apache Drill
    - An open-source implementation of Google Dremel
    - Apache Drill is built on top of Hadoop
2. Presto
    - Java-based execution engine designed specifically for datalakes
3. Apache Impala
    - An executor component located on each data node to avoid sending it through the network
4. Dremio
    - Dremio is built on top of Apache Arrow
    - Utilize ”reflections” to accelerate query execution on external files

References

[1] A. Gubarev, D. Delorey, G. M. Romer, H. Ahmadi, J. Shute, J. J. Long, M. Tolton, M. Pasumansky, N. Shivakumar, S. Melnik, S. Min, and T. Vassilakis. Dremel: A decade of interactive sql analysis at web scale. pages 3461–3472, 2020. URL http://www.vldb.org/pvldb/vol13/p3461-melnik.pdf.

[2] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems (TOCS)*, 6(1):51–81, 1988.