Discussion: Dremel

Some History

- Parallel DB Systems have been around for 30 years prior
- Historical DB companies supporting parallelism include:
  - Teradata, Tandem, Informix, Oracle, RedBrick, Sybase, DB2

NoSQL

- Along came NoSQL (early-mid 2000s)
  - The idea that databases are slow
  - Complaints included
    - Too slow
    - Too much loading time
    - Too monolithic and complex
      - Instruction manuals of ~500 pages
    - Too much heft for “internet scale” applications
    - Too expensive
    - Too hard to understand

NoSQL

- The story of NoSQL and its intimate relationship with Google
- This is the OLAP story, not the OLTP story
- OLTP story
  - BigTable (‘06) => MegaStore (‘11) => Spanner, F1 (‘12)
  - Less consistency => More consistency
  - Contemporaries:
    - PNUTS, Cassandra, HBase, CouchDB, Dynamo

NoSQL

- OLAP story
  - MapReduce (04) => Dremel (10)
  - Less using pdb principles => More using pdb principles
  - By 2010, Google had restricted MapReduce to complex batch processing, with Dremel for interactive analytics
  - Contemporaries:
    - MapReduce: Hadoop (Yahoo)
    - PSQL-on-MapReduce: Pig (Yahoo), Hive (Facebook)
    - PSQL-not-on-MapReduce: Impala

Map-Reduce

- 2004: Google published MapReduce.
  - Parallel programming paradigm
  - Pros:
    - Relatively fast
    - Imperative
    - Many real use-cases
  - Cons:
    - Checkpointing all intermediate results
    - No real logic or optimization
    - Very “rigid”, no room for improvement
    - Many bottlenecks

Along comes Dremel

- 2010:
  - Still not a full-fledged parallel database
  - PROJECT-SELECT-AGGREGATE
  - What does it lack?

Along comes Dremel

- 2010:
  - Still not a full-fledged parallel database
  - What does it lack?
    - Support for joins
    - Support for transactions (it is read-only)
    - Support for intelligent partitioning?

Fast-Forward to Today

- Dremel offered as Google BigQuery: one of the top cloud-native data warehouses
  - Won 10 year best paper award in 2020
- Similar ideas adopted by Snowflake, the #1 cloud data warehouse

- Related, but slightly different ideas employed by Spark, and then by Databricks

Conventional Wisdom Didn't Play Out

- Conventional Wisdom: "Shared Nothing" better than "Shared Memory" or "Shared Disk"

- However, cloud-native data warehouses ended up dominating the market all using a shared disk abstraction
  - Also known today as disaggregated storage and compute
  - Shared nothing = aggregated storage and compute
- Dremel set this trend

Dremel: A Decade of Interactive SQL Analysis at Web Scale*

Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis
Original authors of VLDB 2010 Dremel paper

Hossein Ahmadi, Dan Delorey, Slava Min, Mosha Pasumansky, Jeff Shute
Google LLC
dremel-tot-paper@google.com

ABSTRACT

Google's Dremel was one of the first systems that combined a set of architectural principles that have become a common practice in today's cloud-native analytics tools, including disaggregated storage and compute, in situ analysis, and columnar storage for semistructured data. In this paper, we discuss how these ideas evolved in the past decade and became the foundation for Google BigQuery.

PVLDB Reference Format:
Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis, Hossein Ahmadi, Dan Delorey, Slava Min, Mosha Pasumansky, and Jeff Shute. Dremel: A Decade of Interactive SQL Analysis at Web Scale. *PVLDB*, 13(12): 3461-3472, 2020.
DOI: https://doi.org/10.14778/3415478.3415568

1. INTRODUCTION

Dremel is a distributed system for interactive data analysis that was first presented at VLDB 2010 [32]. That same year, Google launched BigQuery, a publicly available analytics service backed by Dremel. Today, BigQuery is a fully-managed, serverless data warehouse that enables scalable analytics over petabytes of data.[1] It is one of the fastest growing services on the Google Cloud Platform.

A major contribution of papers originating from the industry in the past decade, including the Dremel paper, is to demonstrate what kind of systems can be built using state-of-the-art private clouds. This body of work both reduced the risk of exploring similar routes and identified viable directions for future research. Introducing the journal version of the paper [33], Mike Franklin pointed out that it was "eye-opening" to learn that Google engineers routinely analysed massive data sets with processing throughputs in the range of 100 billion records per second [20]. His main take-away was that simply throwing hardware at the problem was not sufficient. Rather, it was critical to deeply understand the structure of the data and how it would be used. Franklin made an on-the-mark prediction that the data volumes described in the paper would soon become relevant to more organizations, given how quickly the "bleeding edge" becomes commonplace in our field. He also called out various opportunities for optimizations and improvements.

This paper focuses on Dremel's key ideas and architectural principles. Much of the overall system design stood the test of time; some of these principles turned into major industry trends and are now considered best practices. Stated in terms of the technology trends highlighted in the recently published Seattle Report on Database Research [1], the main ideas we highlight in this paper are:

- *SQL*: [1] reports that all data platforms have embraced SQL-style APIs as the predominant way to query and retrieve data. Dremel's initial SQL-style dialect got generalized as ANSI-compliant SQL backed by an open-source library and shared with other Google products, notably Cloud Spanner.[2]
- *Disaggregated compute and storage*: The industry has converged on an architecture that uses elastic compute services to analyze data in cloud storage. This architecture decouples compute from storage, so each can scale independently.
- *In situ analysis*: Data lake repositories have become popular, in which a variety of compute engines can operate on the data, to curate it or execute complex SQL queries, and store the results back in the data lake or send results to other operational systems. Dremel's use of a distributed file system and shared data access utilities allowed MapReduce and other data processing systems at Google to interoperate seamlessly with SQL-based analysis.
- *Serverless computing*: As an alternative to provisioned resources, the industry now offers on-demand resources that provide extreme elasticity. Dremel was built as a fully-managed internal service with no upfront provisioning and pay-per-use economics. This concept was successfully ported to BigQuery.

*Invited contribution for the VLDB 2020 Test of Time Award given to the VLDB 2010 paper "Dremel: Interactive Analysis of Web-Scale Datasets" [32]
[1]https://cloud.google.com/bigquery

Google's Dremel was one of the first systems that combined a set of architectural principles that have become a common practice in today's cloud-native analytics tools, including disaggregated storage and compute, in situ analysis, and columnar storage for semistructured data. In this paper, we discuss how these ideas evolved in the past decade and became the foundation for Google BigQuery.

Dremel is a distributed system for interactive data analysis that was first presented at VLDB 2010 [32]. That same year, Google launched BigQuery, a publicly available analytics service backed by Dremel. Today, BigQuery is a fully-managed, serverless data warehouse that enables scalable analytics over petabytes of data.[1] It is one of the fastest growing services on the Google Cloud Platform.

Column Stores

- For OLAP, column stores are a lot better than row stores
- Idea from the 80s, commercialized as Vertica in 2005.
- Key idea: store values for a single column together

- All modern cloud data warehouses use columnar representations, including BigQuery, Snowflake, …

- Why is this better for aggregation/OLAP?

Column Stores

- For OLAP, column stores are a lot better than row stores
- Key idea: store values for a single column together
- Why is this better for aggregation?
  - Better compression; can pack similar values together better
  - Can skip over unnecessary columns
  - Much less data read from disk

Column Stores

- When can column stores suffer relative to row stores?

Column Stores

- When can column stores suffer relative to row stores?
  - Want to point at a certain data item (e.g., find me the year where company XXX was established)
  - Transactions can be bad:
    - Insertions, deletions can be quite terrible
    - Writes require multiple accesses

Dremel: Column Encoding

- Turns out, this now lives on as “Parquet”, universally adopted
- Are there cases where the column encoding scheme proposed doesn’t make much sense?

Dremel: Column Encoding

- Turns out, this now lives on as “Parquet”, universally adopted
- Would this column encoding make sense if:
  - All records have a rigid schema?
  - Not all records obey the schema?
    - Often the case in json/xml – mistakes in data generation
  - If most data “looks the same” with a few exceptions?

Hierarchical Trees

- What factors would you take into account while deciding the fanout for the hierarchical trees?

Hierarchical Trees

- What factors would you take into account while deciding the fanout for the hierarchical trees?
- Too small fanout may do too much unnecessary network bandwidth for too little gain
- Too large fanout may end up overwhelming one node

  - Network bandwidth
  - CPU capability
  - Local Memory
  - Local Disk