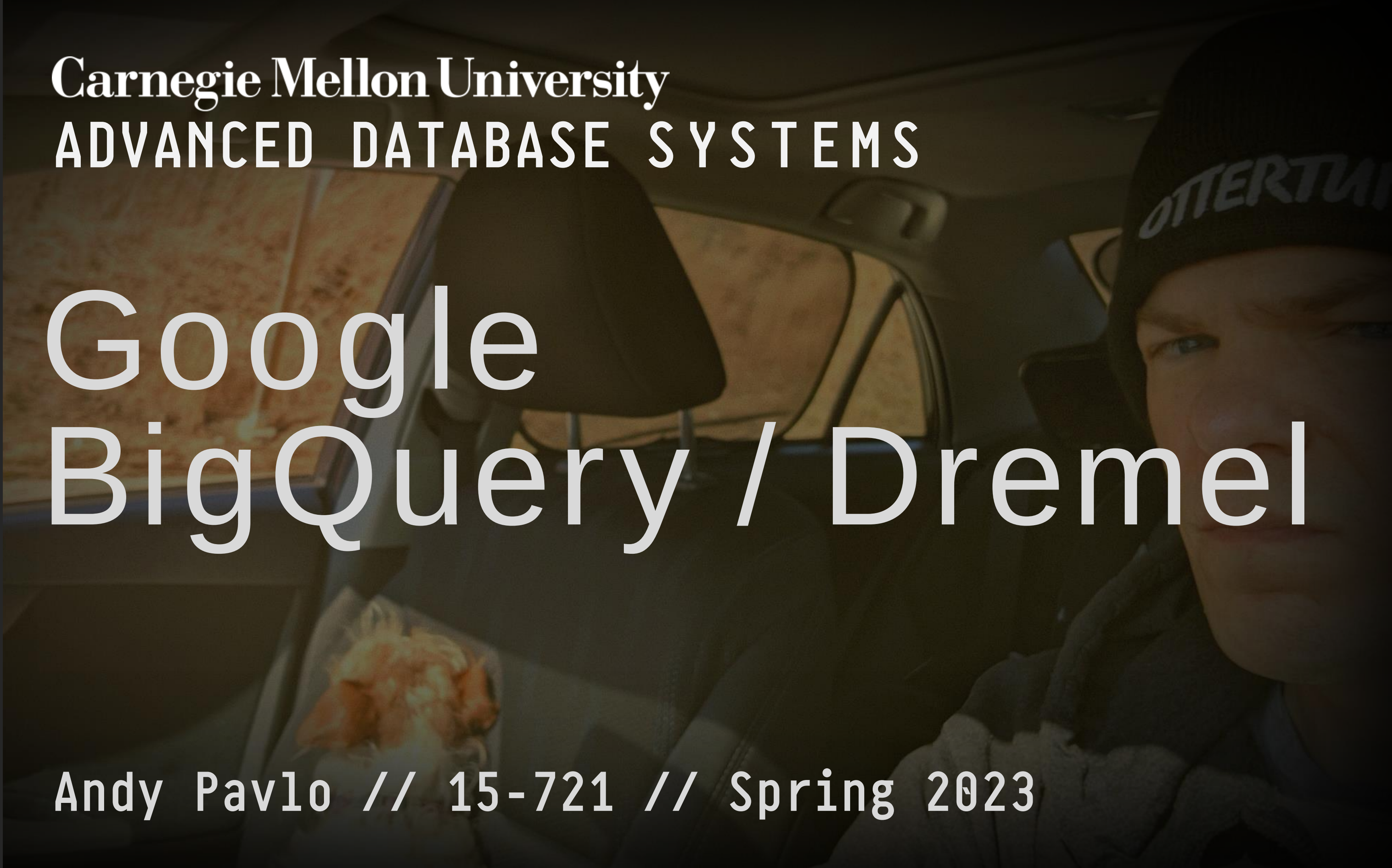
Lecture #19

Carnegie Mellon University

ADVANCED DATABASE SYSTEMS

Google BigQuery / Dremel

Andy Pavlo // 15-721 // Spring 2023



SYSTEMS DISCUSSION

We will spend the rest of the semester discussing real DBMSs that use the methods that we covered this semester.

Three goals of this exercise:

→ Learn how to map concepts and ideas from real-world systems to fundamental techniques.
→ Construct a mental catalog of design justifications for different real-world scenarios.
→ Find out that Andy isn't making stuff up.

SYSTEMS DISCUSSION

Google BigQuery / Dremel

Databricks Spark SQL / Photon

Snowflake

DuckDB

Velox

Amazon Redshift

REOCCURRING THEMES

Resource Disaggregation

Lack of Statistics

Columnar & Non-Relational Data

Vectorized Execution

OBSERVATION

In the 2000s, Google had the most influence in industry trends in development of database systems.

Whenever Google released a research paper describing an internal system, other tech companies would write open-source clones.

→ People assumed that whatever Google did was the way it should be done because they are successful.

DATA SYSTEMS AT GOOGLE

NoSQL

- MapReduce (2004)
- BigTable (2005)
- Chubby (2006)

SQL

- Megastore (2010)
- Vitess (2010)
- Dremel (2011)
- Spanner (2011)
- F1 (2013)
- Mesa (2014)
- Napa (2021)

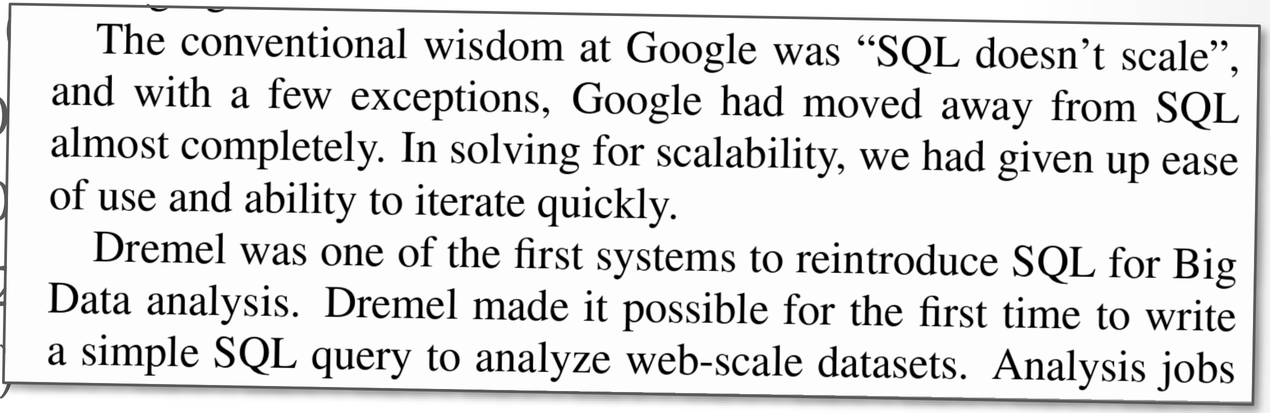
DATA SYSTEMS AT GOOGLE

NoSQL

- MapReduce (
- BigTable (20
- Chubby (200

SQL

- Megastore (2
- Vitess (2010)
- Dremel (2011)
- Spanner (2011)
- F1 (2013)
- Mesa (2014)
- Napa (2021)


The conventional wisdom at Google was "SQL doesn't scale", and with a few exceptions, Google had moved away from SQL almost completely. In solving for scalability, we had given up ease of use and ability to iterate quickly.

Dremel was one of the first systems to reintroduce SQL for Big Data analysis. Dremel made it possible for the first time to write a simple SQL query to analyze web-scale datasets. Analysis jobs

DATA SYSTEMS AT GOOGLE

NoSQL
- MapReduce (2004) → ***Hadoop, Spark***
- BigTable (2005) → ***HBase, Accumulo, Hypertable***
- Chubby (2006) → ***Zookeeper, etcd***

SQL
- Megastore (2010)
- Vitess (2010) → ***Vitess, Planetscale***
- Dremel (2011) → ***Drill, Impala, Dremio***
- Spanner (2011) → ***CockroachDB*, TiDB****
- F1 (2013)
- Mesa (2014)
- Napa (2021)

(*) Only for marketing…

GOOGLE DREMEL

Originally developed in 2006 as a side-project for analyzing data artifacts generated from other tools.
→ The "interactive" goal means that they want to support ad hoc queries on **in-situ** data files.
→ Did not support joins in the first version.

Rewritten in the late 2010s to shared-disk architecture built on top of GFS.

Released as public commercial product (BigQuery) in 2012.

DREMEL: A DECADE OF INTERACTIVE SQL ANALYSIS AT WEB SCALE
VLDB 2020

GOOGLE DREMEL

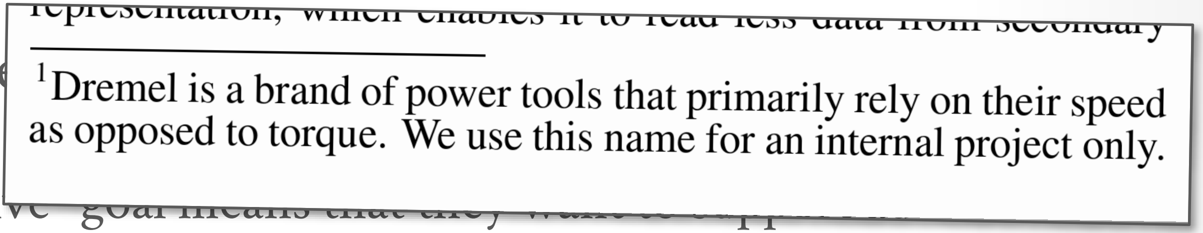
Originally deve
analyzing data

→ The "interactive" goal means that they want to support ad
hoc queries on **in-situ** data files.

→ Did not support joins in the fi

Rewritten in the late 2010s
architecture built on top of

Released as public commer
in 2012.



representation, which enables it to read less data from secondary

[1]Dremel is a brand of power tools that primarily rely on their speed as opposed to torque. We use this name for an internal project only.

DREMEL: A DECADE OF INTERACTIVE SQL ANALYSIS AT WEB SCALE
VLDB 2020

IN-SITU DATA PROCESSING

Execute queries on data files residing in shared storage (e.g., object store) in their original format without first ingesting them into the DBMS (i.e., managed storage).

→ This is what people usually mean when they say **data lake**.

→ A **data lakehouse** is the DBMS that sits above all this.

The goal is to reduce the amount of prep time needed to start analyzing data.

→ Users are willing to sacrifice query performance to avoid having to re-encode / load data files.

GOOGLE DREMEL

Shared-Disk / Disaggregated Storage

Vectorized Query Processing

Shuffle-based Distributed Query Execution

Columnar Storage
→ Zone Maps / Filters
→ Dictionary + RLE Compression
→ Only Inverted Indexes

Hash Joins Only

Heuristic Optimizer + Adaptive Optimizations

DREMEL: QUERY EXECUTION

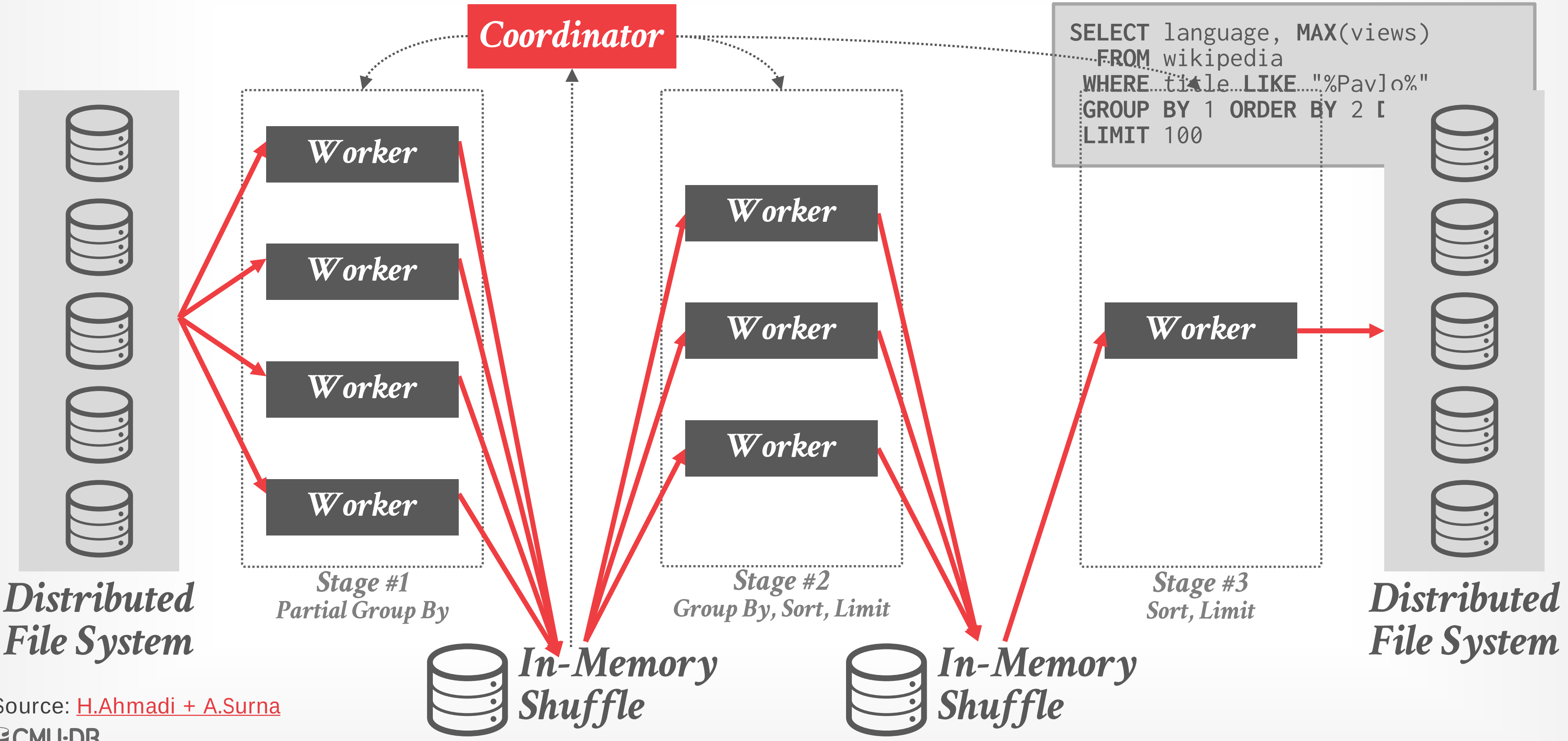
DBMS converts a logical plan into **stages** (pipelines) that contain multiple parallel **tasks**.

→ Each task must be deterministic and idempotent to support restarts.

Root node (Coordinator) retrieves all the meta-data for target files in a batch and then embeds it in the query plan.

→ Avoids 1000s of workers hitting up distributed file system for meta-data at the same time.

DREMEL: QUERY EXECUTION



```
SELECT language, MAX(views)
  FROM wikipedia
 WHERE title LIKE "%Pavlo%"
 GROUP BY 1 ORDER BY 2 D
 LIMIT 100
```

Source: H.Ahmadi + A.Surna

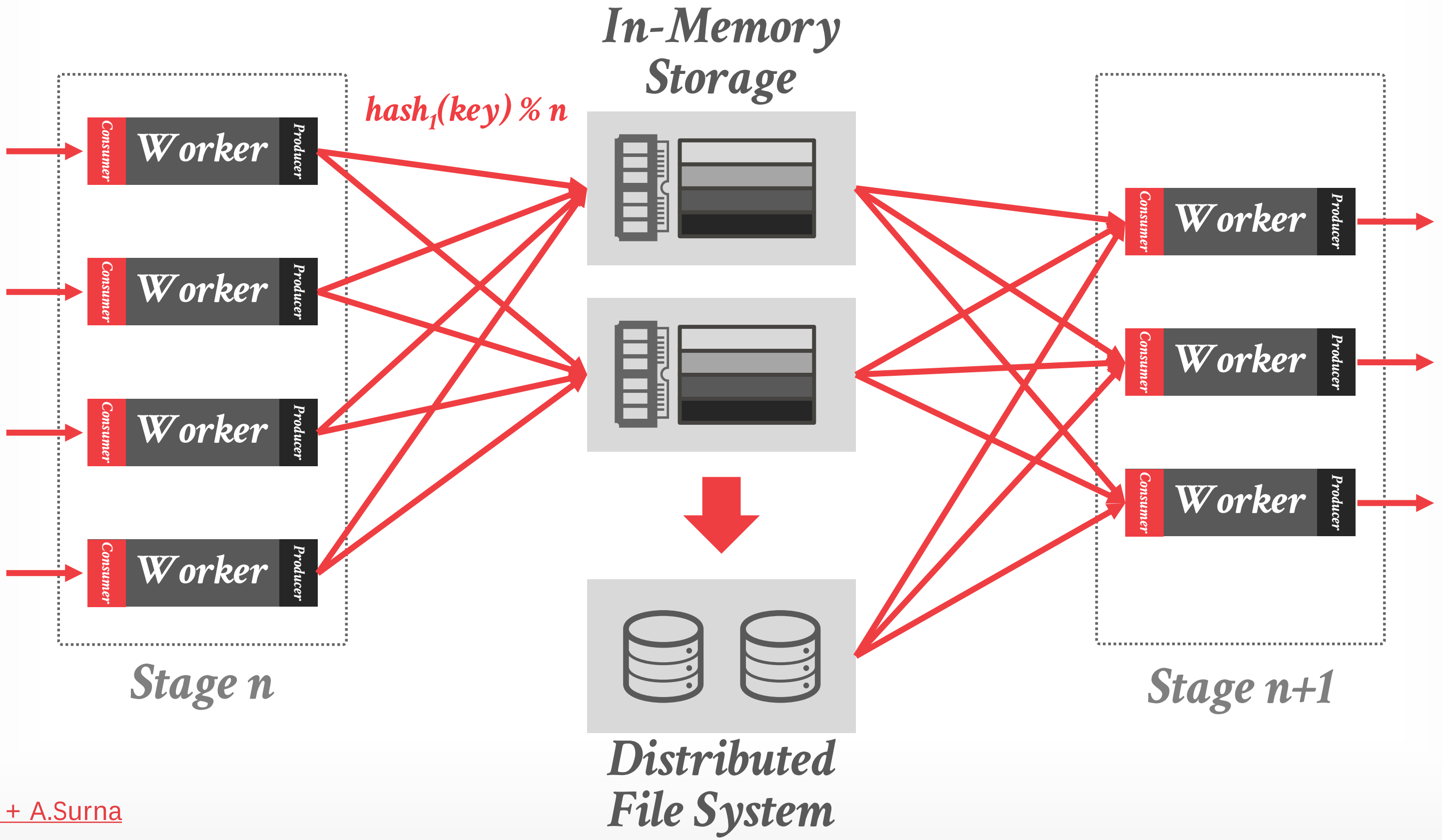
DREMEL: IN-MEMORY SHUFFLE

Producer/consumer model for transmitting intermediate results from each stage to the next using dedicated nodes.

→ Workers send output to shuffle nodes.
→ Shuffle nodes store data in memory in hashed partitions.
→ Workers at the next stage retrieve their inputs from the shuffle nodes.

Shuffle nodes store this data in memory and only spill to disk storage if necessary.

DREMEL: IN-MEMORY SHUFFLE



Source: H.Ahmadi + A.Surna

DREMEL: IN-MEMORY SHUFFLE

The shuffle phases represent checkpoints in a query's lifecycle where that the coordinator makes sure that all tasks are completed.

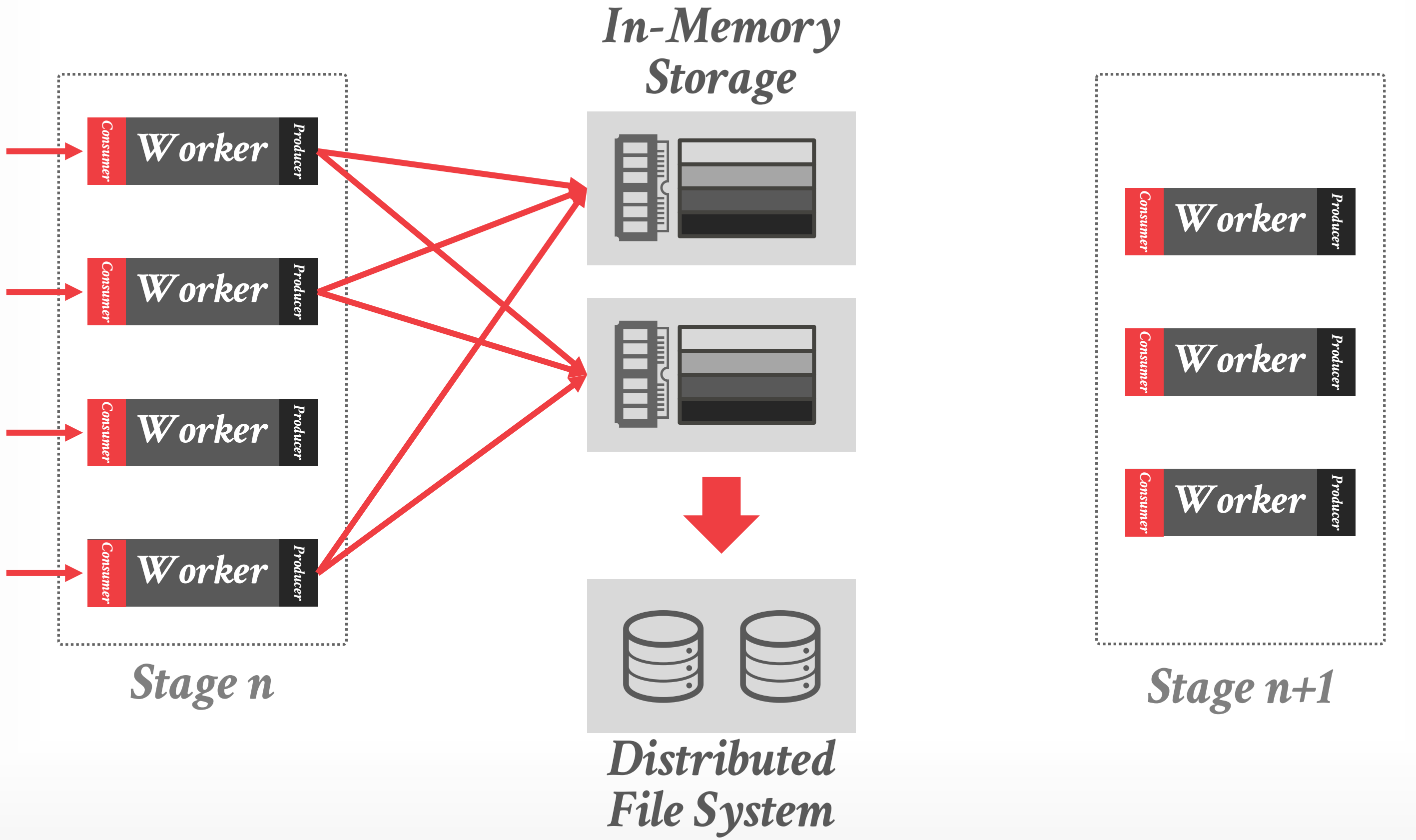
Fault Tolerance / Straggler Avoidance:

→ If a worker does not produce a task's results within a deadline, the coordinator speculatively executes a redundant task.

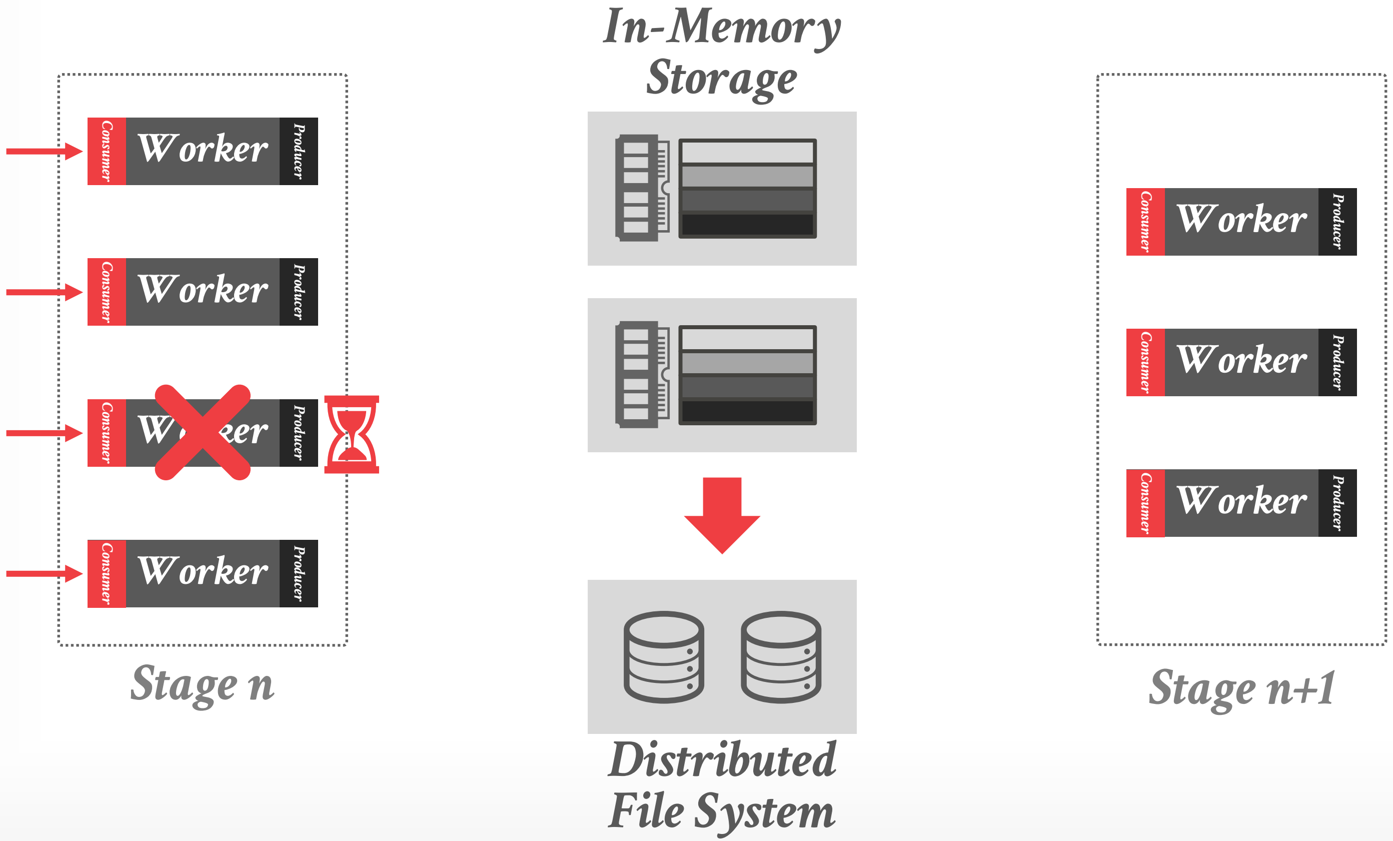
Dynamic Resource Allocation:

→ Scale up / down the number of workers for the next stage depending size of a stage's output.

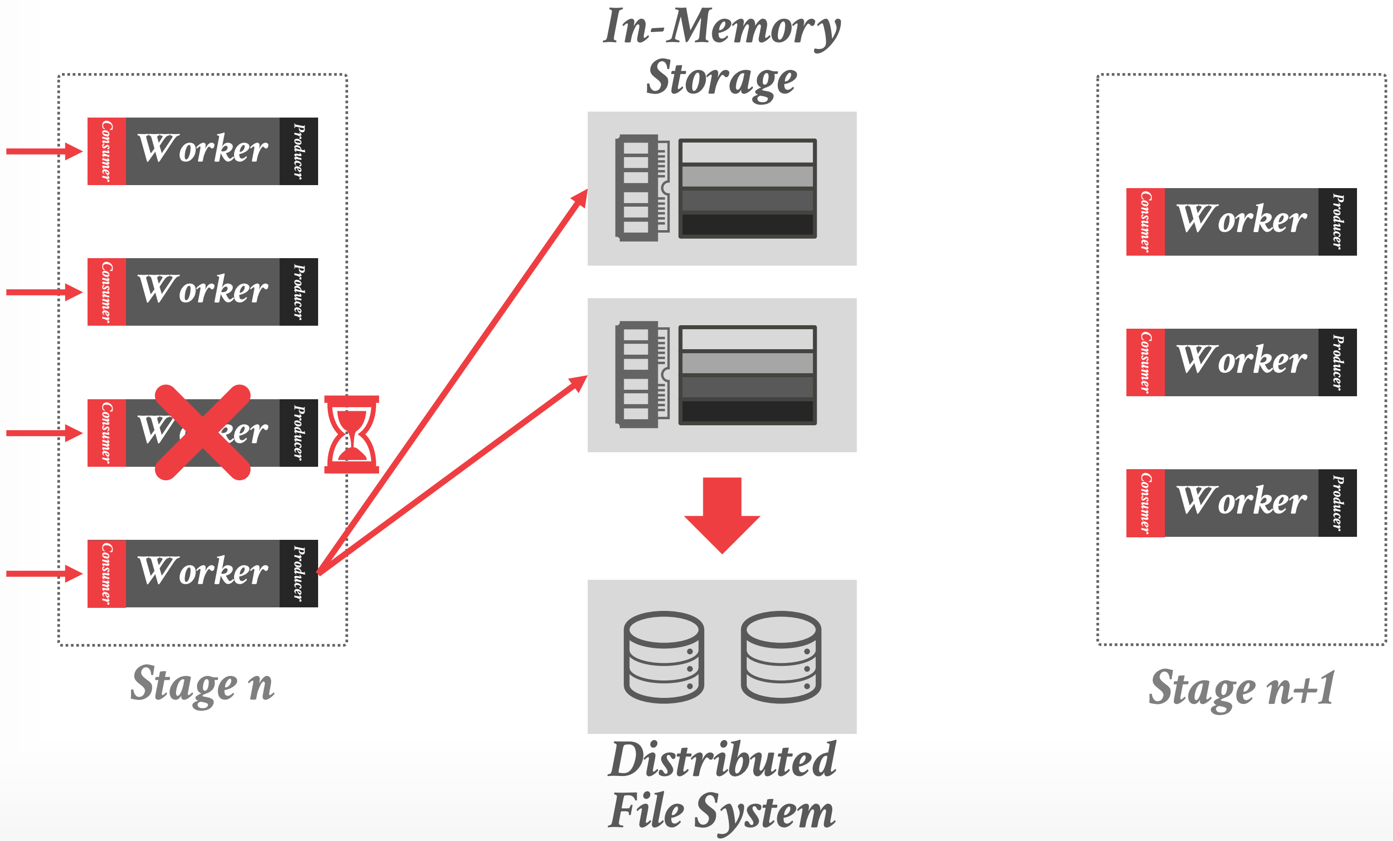
DREMEL: IN-MEMORY SHUFFLE



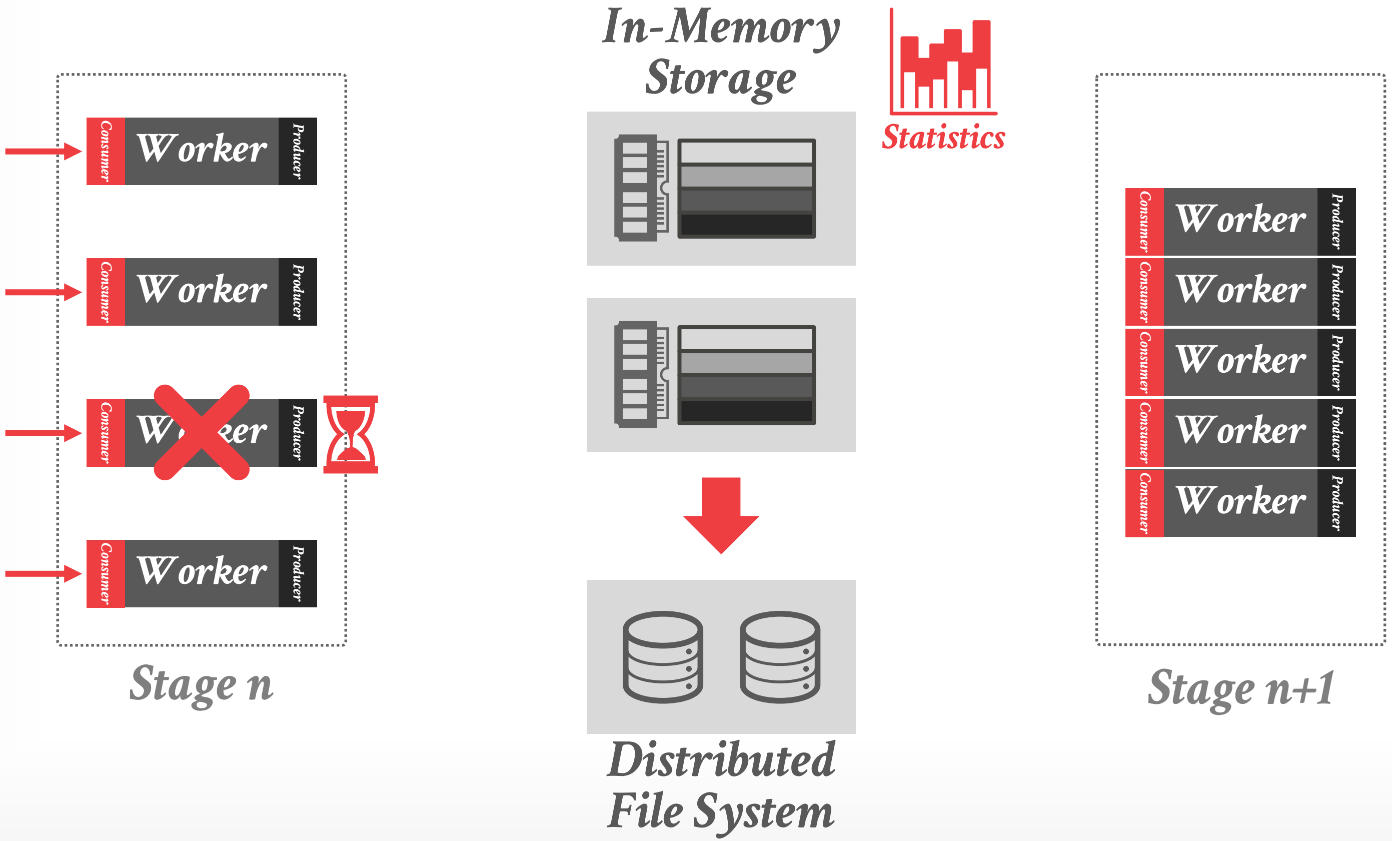
DREMEL: IN-MEMORY SHUFFLE



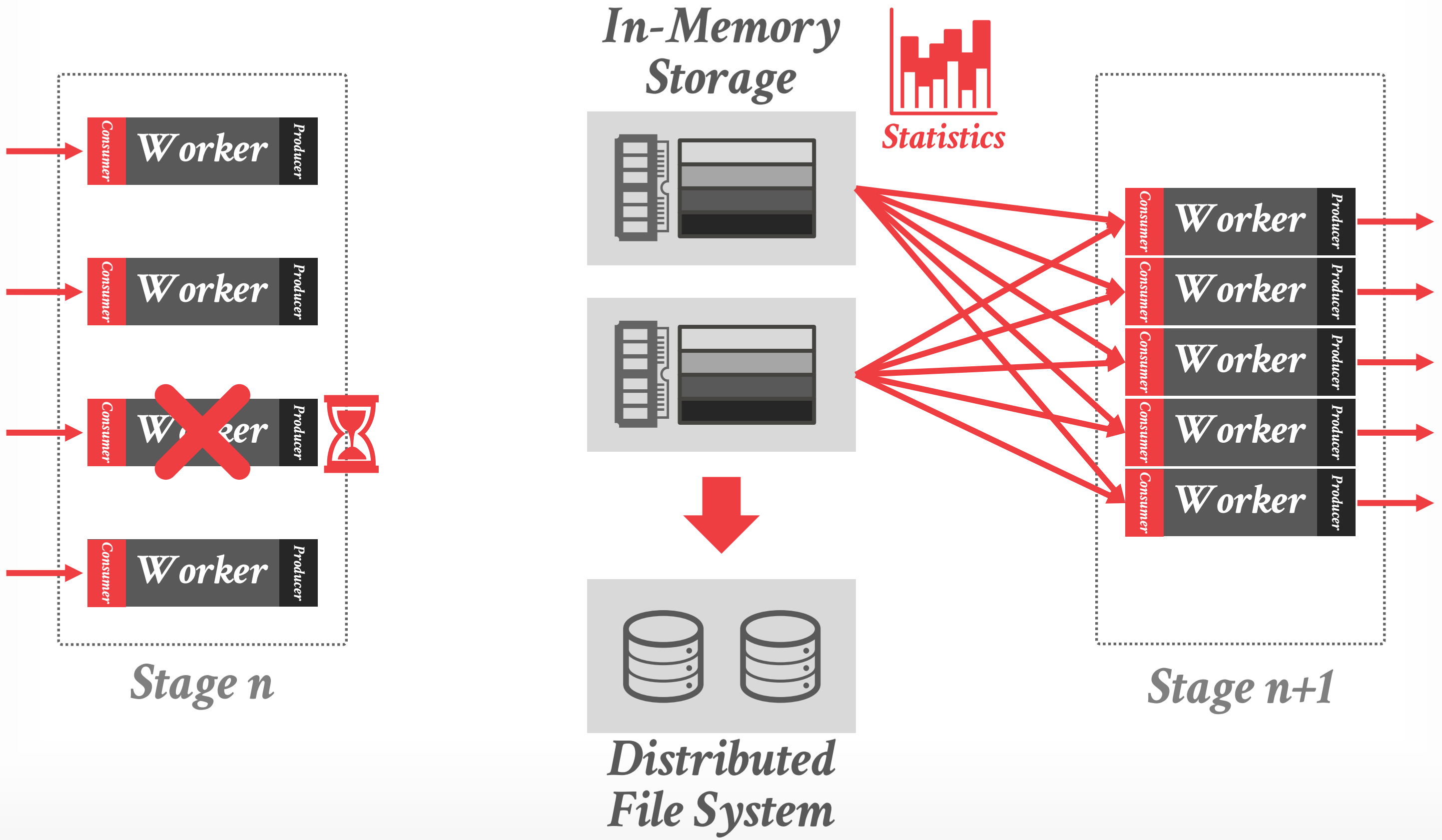
DREMEL: IN-MEMORY SHUFFLE



DREMEL: IN-MEMORY SHUFFLE



DREMEL: IN-MEMORY SHUFFLE



OBSERVATION

We discussed how query optimizers rely on cost models derived from statistics extracted from data.

But how can the DBMS optimize a query if there are no statistics?

- → Data files the DBMS has never seen before.
- → Query APIs from other DBMSs (connectors).

DREMEL: A DECADE OF INTERACTIVE SQL ANALYSIS AT WEB SCALE
VLDB 2020

DREMEL: QUERY OPTIMIZATION

Dremel's optimizer uses a stratified approach with rule-based + cost-based optimizer passes to generate a preliminary physical plan to start execution.

→ Rules for predicate pushdown, star schema constraint propagation, primary/foreign key hints, join ordering.

→ Cost-based optimizations only on data that the DBMS has statistics available (e.g., materialized views).

To avoid the problems with bad cost model estimates, Dremel uses dynamic query optimization…

Source: H.Ahmadi + A.Surna

DREMEL: DYNAMIC QUERY OPTIMIZATION

Dremel changes the query plan before a stages starts based on observations from the preceding stage.

→ Avoids the problem of optimizer making decisions with inaccurate (or non-existing) data statistics.

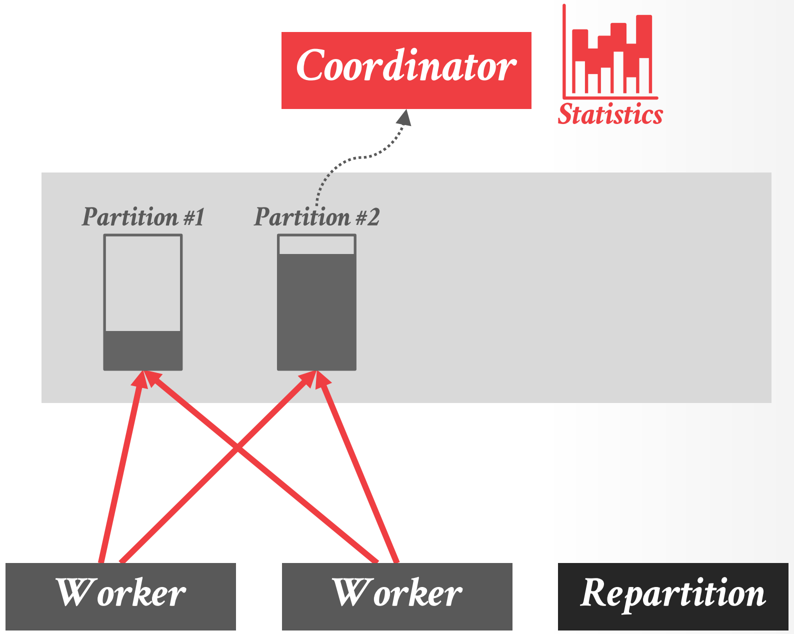
Optimization Examples:

→ Change the # of workers in a stage.
→ Switch between shuffle vs. broadcast join.
→ Change the physical operator implementation.
→ Dynamic repartitioning.

DREMEL: DYNAMIC REPARTITIONING

Dremel dynamically load balances and adjusts intermediate result partitioning to adapt to data skew.

DBMS detects whether shuffle partition gets too full and then instructs workers to adjust their partitioning scheme.

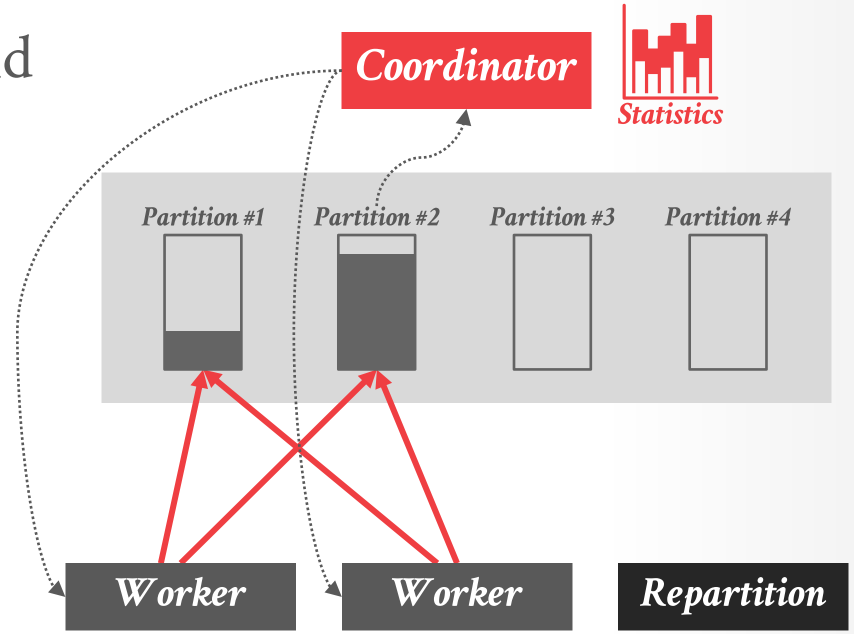


Source: H.Ahmadi + A.Surna

DREMEL: DYNAMIC REPARTITIONING

Dremel dynamically load balances and adjusts intermediate result partitioning to adapt to data skew.

DBMS detects whether shuffle partition gets too full and then instructs workers to adjust their partitioning scheme.

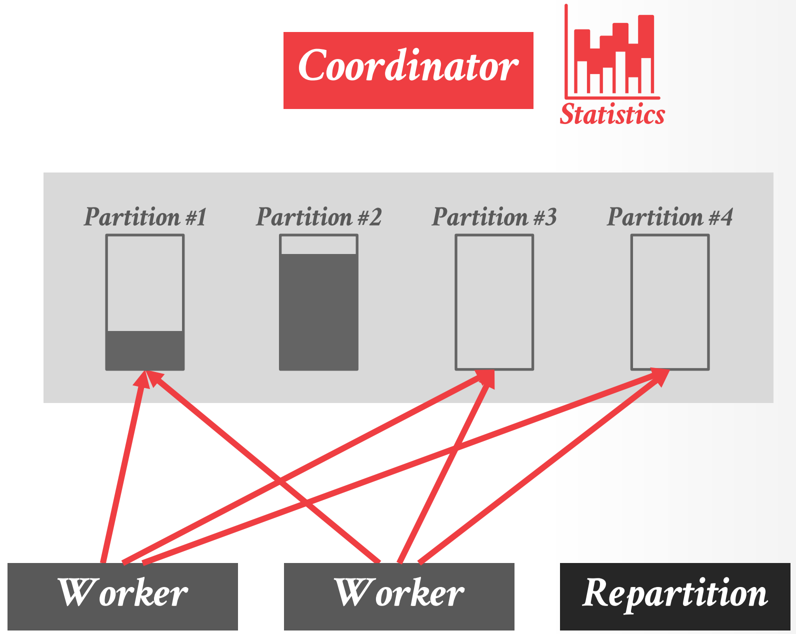


Source: H.Ahmadi + A.Surna

DREMEL: DYNAMIC REPARTITIONING

Dremel dynamically load balances and adjusts intermediate result partitioning to adapt to data skew.

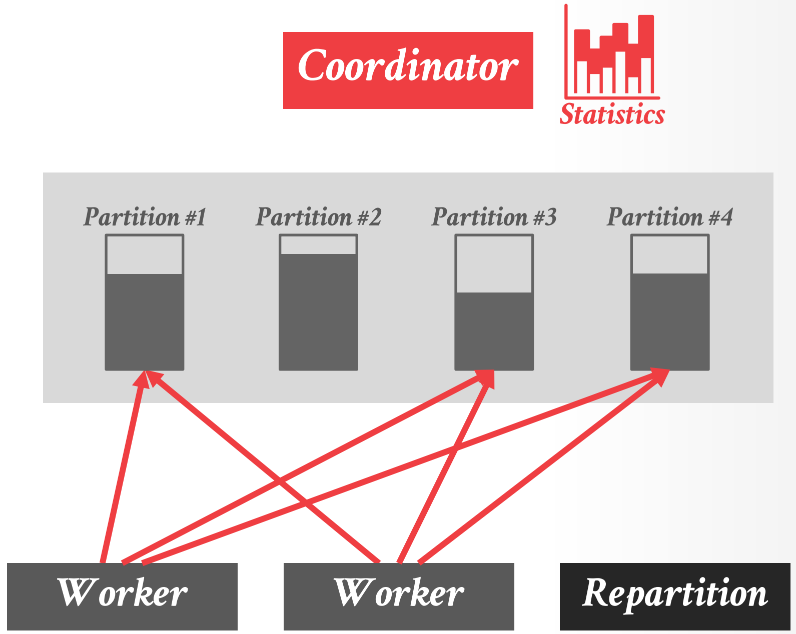
DBMS detects whether shuffle partition gets too full and then instructs workers to adjust their partitioning scheme.



DREMEL: DYNAMIC REPARTITIONING

Dremel dynamically load balances and adjusts intermediate result partitioning to adapt to data skew.

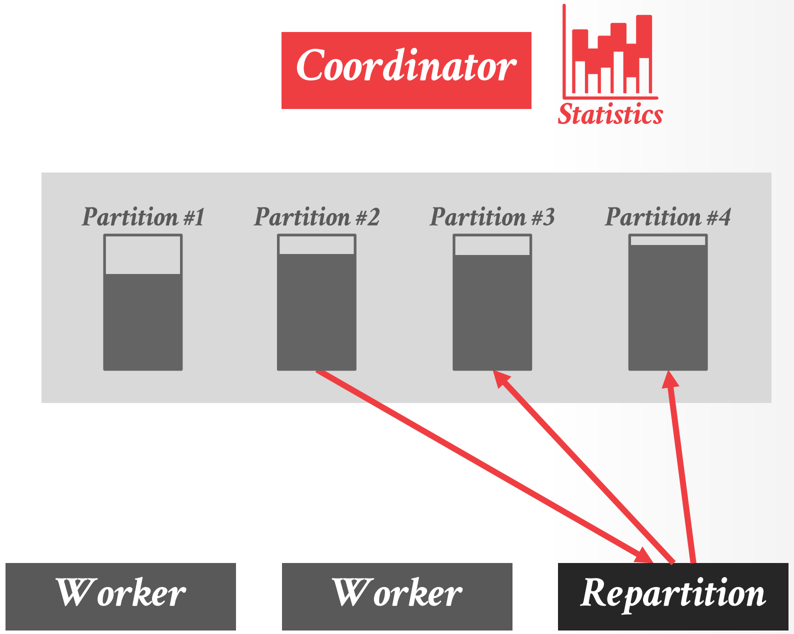
DBMS detects whether shuffle partition gets too full and then instructs workers to adjust their partitioning scheme.



DREMEL: DYNAMIC REPARTITIONING

Dremel dynamically load balances and adjusts intermediate result partitioning to adapt to data skew.

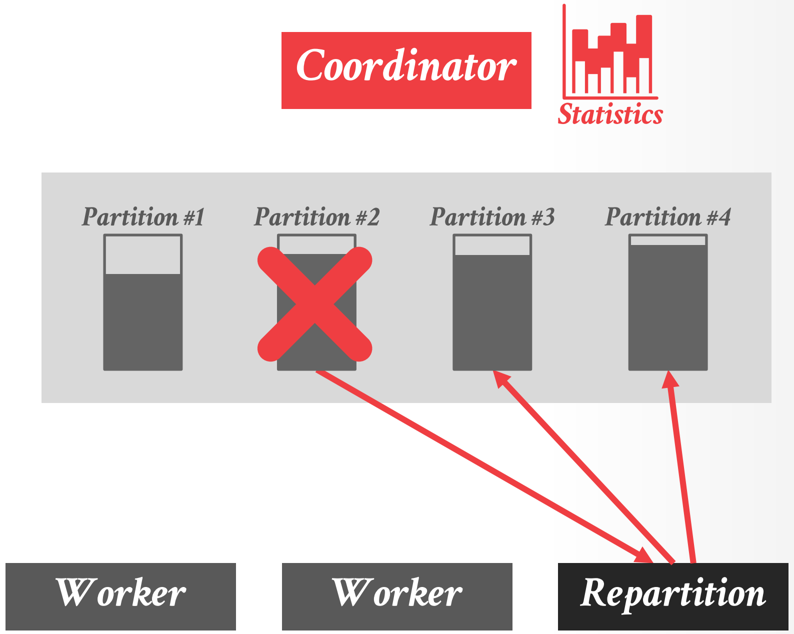
DBMS detects whether shuffle partition gets too full and then instructs workers to adjust their partitioning scheme.



DREMEL: DYNAMIC REPARTITIONING

Dremel dynamically load balances and adjusts intermediate result partitioning to adapt to data skew.

DBMS detects whether shuffle partition gets too full and then instructs workers to adjust their partitioning scheme.



Source: H.Ahmadi + A.Surna

DREMEL: STORAGE

DBMS relies on Google's distributed file system (Colossus) to scale out storage capacity.

Relies on Capacitor's columnar encoding scheme for nested relational and semi-structured data.

→ Think of it as JSON/YAML without the slowness.
→ Capacitor also provides access libraries with basic filtering.
→ Similar to Parquet vs. Orc formats.

Repetition and definition fields are embedded in columns to avoid having to retrieve/access ancestor attributes.

DREMEL: SCHEMA REPRESENTATION

Dremel's internal storage format is self-describing
→ Everything the DBMS needs to understand what is in a file is contain within the file.

But the DBMS must parse a file's embedded schema whenever it wants to read that a file.
→ Tables can have thousands of attributes. Most queries only need a subset of attributes.

DBMS stores schemas in a columnar format to reduce overhead when retrieving meta-data.

DREMEL: SQL

In the early 2010s, many of Google's internal DBMS projects each had their own SQL dialect.

The GoogleSQL project unified these redundant efforts to build a data model, type system, syntax, semantics, and function library.

Open-Source Version: ZetaSQL

SPANNER: BECOMING A SQL SYSTEM
SIGMOD 2017

OBSERVATION

Since the 2011 VLDB paper, there are DBMS projects that are copies or inspired by Dremel.
→ **Apache Drill**
→ **Dremio**
→ **Apache Impala**

There is also Apache Uniffle (Tencent) that provides distributed shuffle as a service.

APACHE DRILL

Drill is an open-source implementation of Dremel built on top of Hadoop.

→ Project started in 2012 at MapR.

Supports query codegen via Janino embedded Java compiler.

HPE announced they will no longer support Drill development in 2020.

DREMIO

Open-source / commercial DBMS inspired by Dremel based on Apache Arrow.

→ Started in 2015 by CMU alum.

Leverages user-defined materialized views ("reflections") to speed up query execution on external data files.

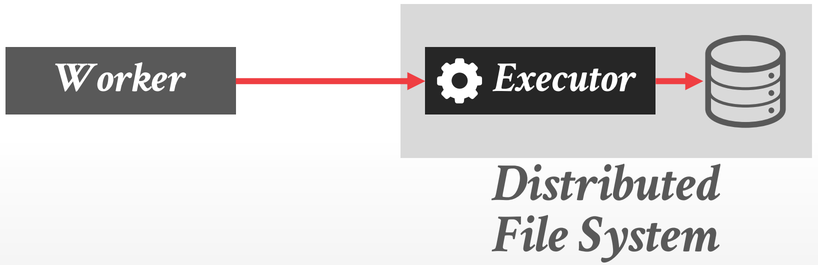
Also relies on Java-based codegen and vectorization.

APACHE IMPALA

Impala is another Dremel inspired DBMS for executing queries on distributed filesystems.
→ Started in 2012 at Cloudera by ex-Google DB people.

Supports codegen of filters and parsing logic.

Co-locate executor component on each data node to provide parsing and predicate pushdown.



PARTING THOUGHTS

Dremel is an innovative DBMS that predates all other major cloud-native OLAP DBMSs.

The shuffle phase seems wasteful but it simplifies engineering and can improve performance.

It is also a good example of the benefit of decomposing a DBMS's components into individual services to abstract raw resources.

NEXT CLASS

Spark SQL / Photon Engine