Big Data Systems

Djellel Difallah

Spring 2023

Lecture 11 – Spark SQL, Hive, and Big Data File Formats

Outline

- Spark SQL
- An Overview of Apache Hive
- Big Data File Formats
  - Avro
  - ORC
  - Parquet

SPARK SQL

Spark SQL

- Module for structured data processing
- More expressive than simple RDD processing
- Internal optimizations: Utilizes extra information for efficiency (thanks to the data format)
- Unified execution engine for structured data processing.

Spark DataFrame

- A DataFrame is a distributed collection of data with named columns.

- A Spark DataFrame Conceptually similar to a relational database table or a “dataframe” in R/Python.
  - Note: The implementation differ!

- Can be created from various sources, such as structured data files: csv, external DB (with connector), or existing RDDs.

OVERVIEW ON APACHE HIVE

What is Apache Hive?

- Developped by Facebook
- A data warehousing infrastructure based on Hadoop
- A system for managing and querying structured data built on top of
  - Map-Reduce for execution
  - HDFS for storage
- Has HiveQL (HQL), a SQL-like query language
- Interprets HiveQL and generates MapReduce jobs that run on the cluster
- Not designed for OLTP

Hive Data Organization

- Databases: namespaces that separate tables and other objects
- Tables: homogeneous units of data with the same schema
  - Analogous to tables in an RDBMS
- Partitions: determine how the data is stored
  - Allow efficient access to subsets of the data
- Buckets/clusters
  - For subsampling within a partition
  - Join optimization

HiveQL

- HiveQL (HQL) provides the basic SQL-like operations
  - Column projection using SELECT
  - Filter rows using WHERE
  - JOIN between tables
  - Aggregate using GROUP BY
  - Manage tables and queries with CREATE, DROP, and ALTER
- Missing large parts of full SQL specification:
  - HAVING clause in SELECT
  - Correlated sub-queries
  - Sub-queries outside FROM clauses
  - Updatable or materialized views
  - Stored procedures
  - Because it is only a wrapper for map-reduce execution, complex queries can be hard to optimise

BIG DATA FILE FORMATS

Data File Format

- In RDBMs
  - Proprietary storage engines: extremely well engineering and tightly integrated with the rest of the DBMS.
    - E.g., MySQL (InnoDB, MyISAM)
  - NSM vs DSM

- In big data:
  - Hadoop’s HDFS does not have the notion of schema: *It’s just a file system*
  - Serializability and storage: open source (for the most part)
    - Data exchange: Hot topics (Protobufs, Thrift, Arrow). Focus is on Interoperability
    - Data storage: our focus next

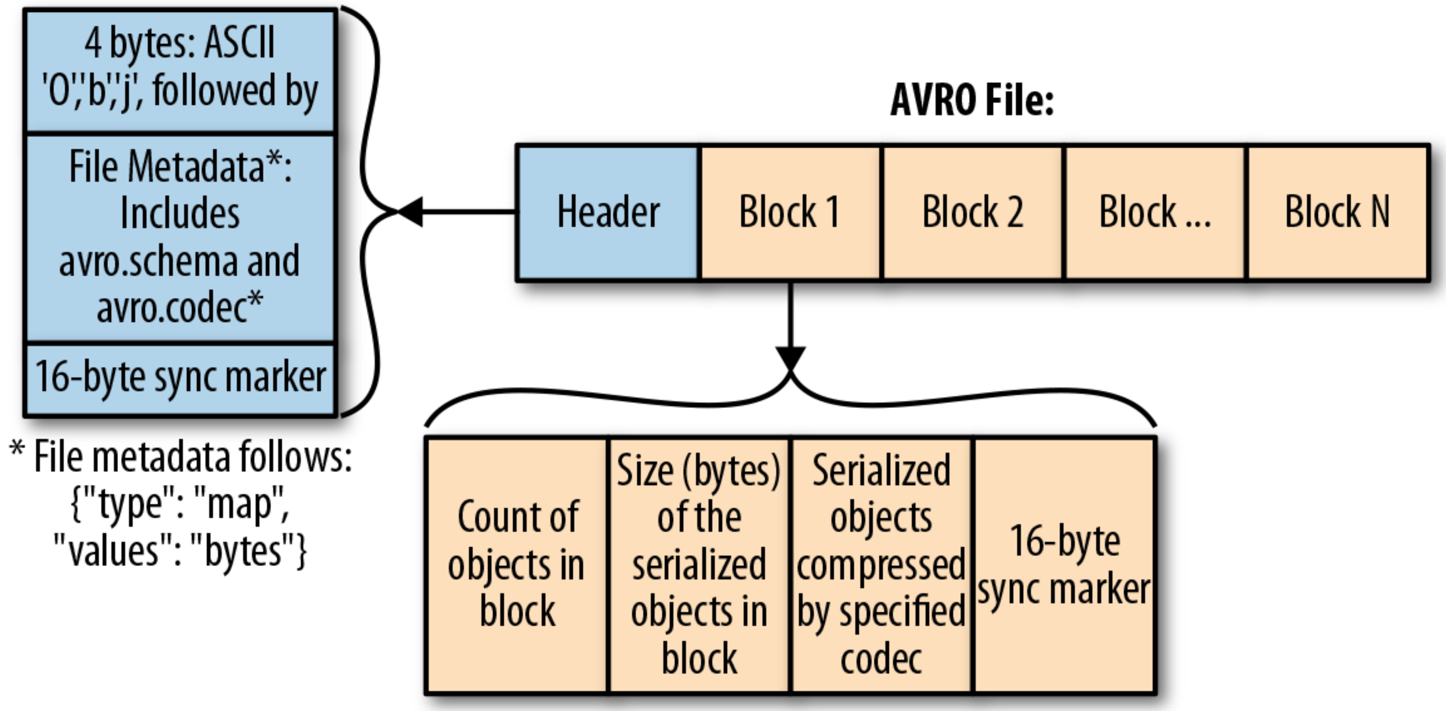
Apache Avro

- Avro is a data serialization system.
- A container file, to store persistent data.
- Compact, fast, binary data format.
- Remote procedure call (RPC).
- Code generation is not required to read or write data files nor to use or implement RPC protocols.
- Code generation as an optional optimization, only worth implementing for statically typed languages.

Avro Schema

- Not meant to be self-describing (at the record level), but rather for long term storage.
- When Avro data is written, the schema is stored within the file, so that files may be processed later by any program.
- When Avro data is read, the schema used when writing it is loaded.
- When Avro is used in RPC, the client and server exchange schemas in the connection handshake.
- Avro schemas are defined with JSON .

The Avro File Format


4 bytes: ASCII 'O','b','j', followed by
File Metadata*: Includes avro.schema and avro.codec*
16-byte sync marker
* File metadata follows: {"type": "map", "values": "bytes"}
AVRO File:
Header
Block 1
Block 2
Block ...
Block N
Count of objects in block
Size (bytes) of the serialized objects in block
Serialized objects compressed by specified codec
16-byte sync marker

Apache Orc

- Optimized Row Columnar is a compat column-oriented storage format
- Originally developed for Apache Hive (a database system that runs on Hadoop)
- Replaces the RCFile format
- ORC is designed to improve perfomance for read-heavy workloads
- Features:
  - Organize data by columns for better compression and analytics query performance
  - Lightweight compression (Zlib, Snappy)
  - Vectorized query execution
  - Predicate pushdown

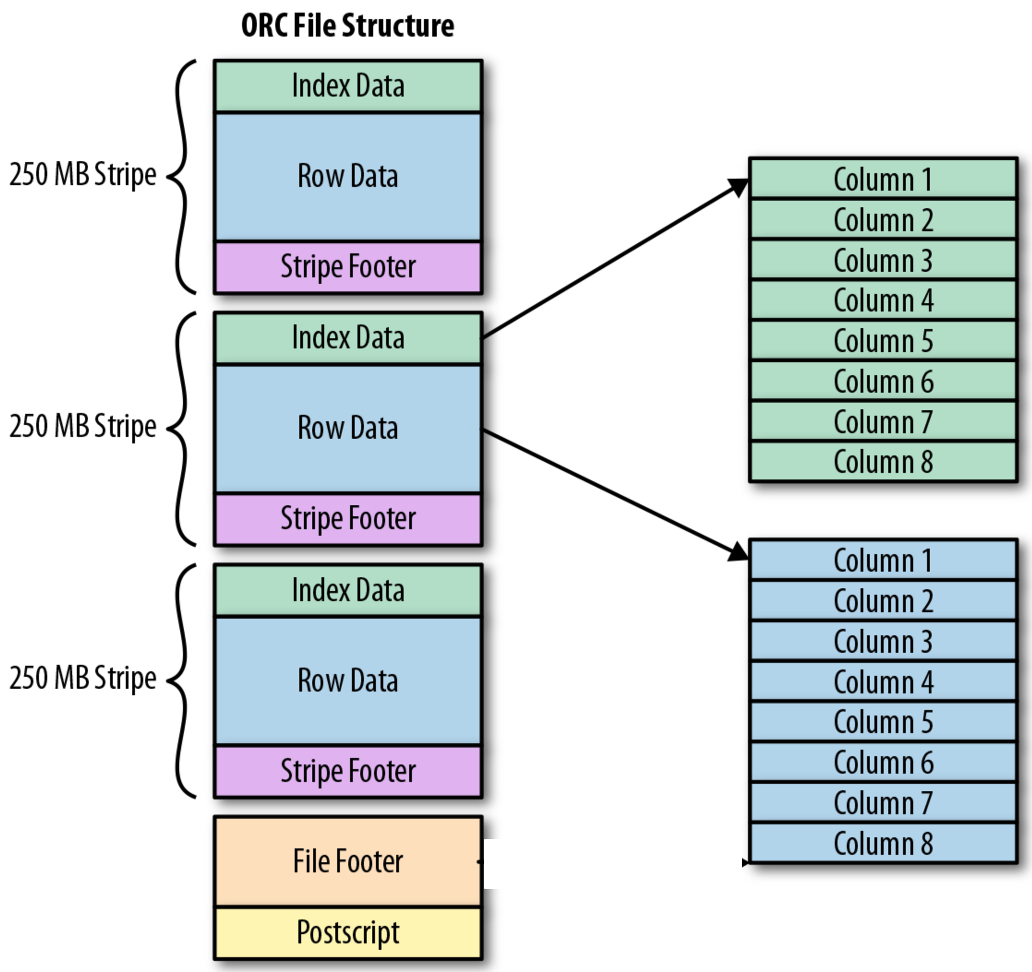
File Format Characteristics

- ORC files are organized into independent stripes of data. Each stripe consists of an index, data, and a footer.
  - Index: (every 10,000 rows) Index to jump to particular column, Min/Max per column. Useful for skipping.
  - Data: data is stored column-wise, with each column divided into streams
  - Stripe Footer: Meta information e.g., Column encoding information,

- File Footer:
  - List of stripe location
- Postscript:
  - Compression parameters

- Why at the end?
  - HDFS append only

The ORC File Format



Operationalizing the Data Lake by Holden Ackerman, Jon King. July 2019
Publisher(s): O'Reilly Media, Inc.

Predicate Pushdown

- The system checks a predicate against file metadata to decide whether rows must be read.
- This increases the potential for skipping.

- Independent reading: https://engineering.fb.com/2014/04/10/core-data/scaling-the-facebook-data-warehouse-to-300-pb/

Apache Parquet

"Apache Parquet is a file format designed for efficient data storage and retrieval. It provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk"

- Parquet was inspired by Google's Dremel paper and developed by Cloudera and Twitter.
- An open-source, columnar storage file format optimized for analytical querying.
  - More precisely: "Hybrid Storage"
- It offers nested data support and versatile compression options.
- Parquet is self-describing. It contains metadata that includes file schema and structure.

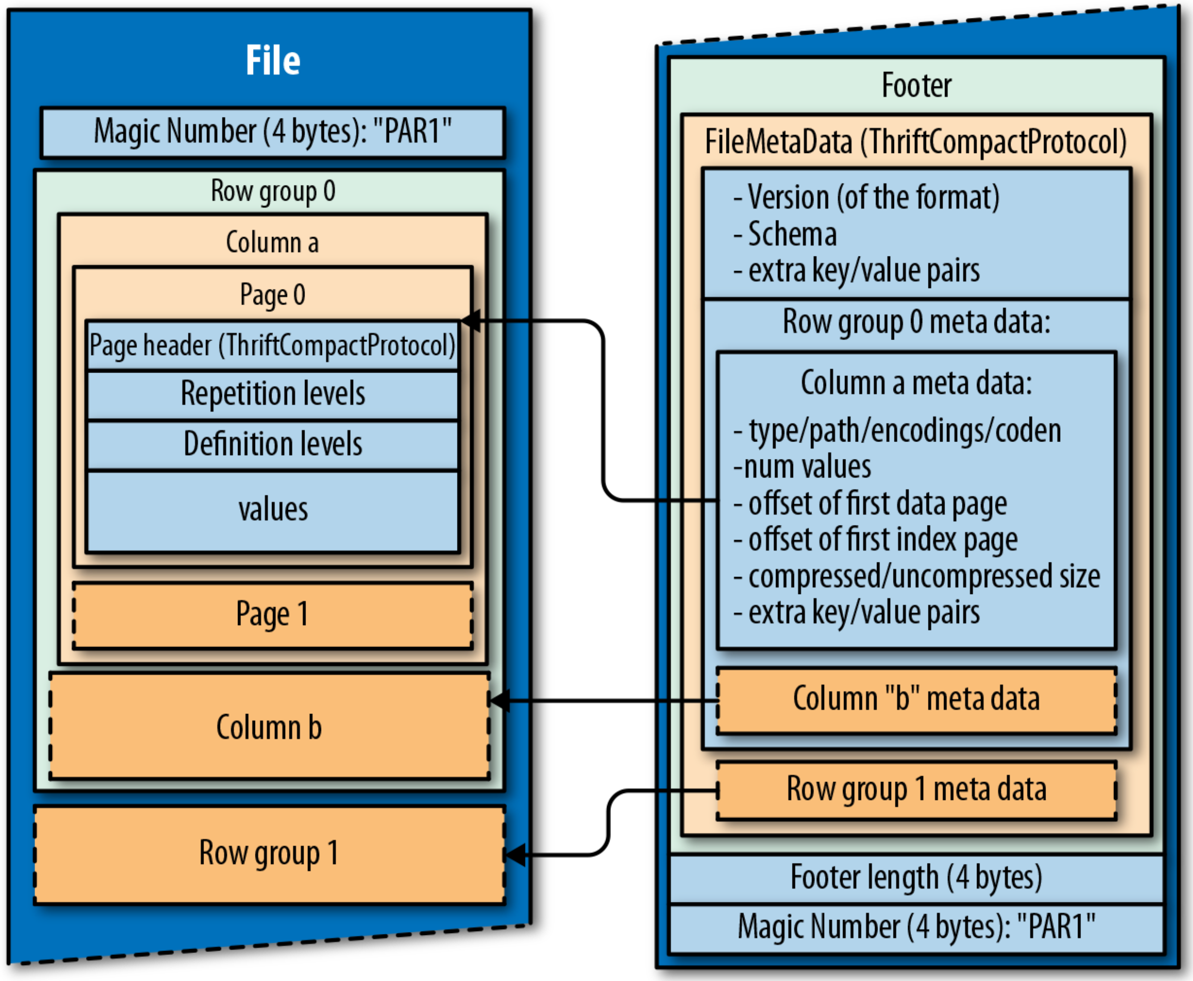
Key Features

- Fast data processing and storage efficiency:
  - Designed for complex nested data structures
  - Highly efficient columnar compression and flexible encoding schemes
  - Future-proof and compatible with various query engines
- Splittable and extensible:
  - File footer metadata for efficient reads and parallel processing
  - Automatic schema merging for schema evolution
- Ideal for analytics (OLAP) use cases:
  - Highly-efficient data compression and decompression
  - Increased data throughput and performance with techniques like data skipping

File Format Characteristics

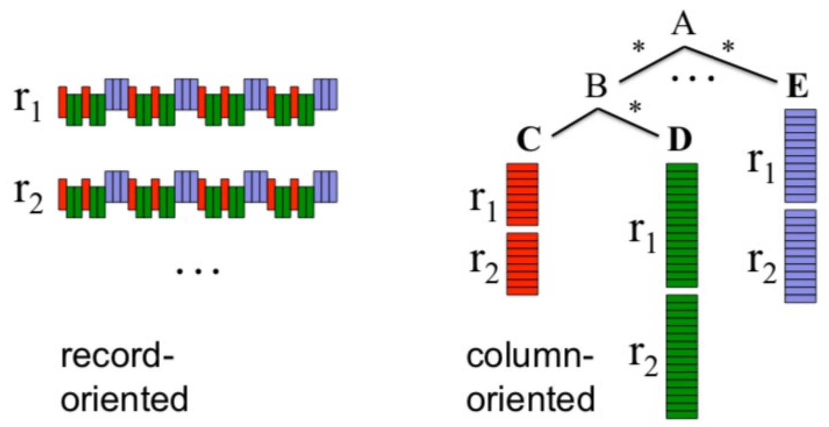
- Parquet files are composed of row groups, header and footer.
- Row groups (default 128MB)
  - Multiple columns (chunks)
- Column Chunks
  - Multiple Pages
- Pages
  - Metadata (min, max, count)
  - Repetition and definition levels
  - Encoded values

The Parquet File Format



Parquet support for Nested Data Types

- Parquet's distinct characteristic is its ability to store nested data structures in a columnar manner.
- This allows for individual reading of nested fields in a Parquet file format without requiring access to every field within the nested structure.
- Inspired by the Google Dremel Paper:



[Dremel: Interactive Analysis of Web-Scale Datasets]