Published in Dev Genius

Nivedita Mondal Follow

Oct 5, 2022 · 11 min read · Listen

Save

Walkthrough the Big Data File Formats

In this article, we are going to discuss different file formats that we use to store the data in our storage area.

Why Selection of File Format is Important?

File format play a major role in optimizing the performance of code developed. Following measurement should be considered by a developer to select any one file format before starting with the development. From the file format that we choose, we should be able to,

- **Read** the data **faster for processing.**
- **Write** the data into the target location **faster** without any data loss.

- Data stored as any type of file should provide a property of **Splittablity**, which help in **parallel processing.**
- **Schema** changes should be made easily without any impact to the existing data.
- Should have enhanced **compression** which makes an add on significance while processing the data.

Two ways of storing files in HDFS are

The file format in *Hadoop* roughly divided into two categories: row-oriented and column-oriented:

Row-oriented:

The same row of data stored together that is continuous storage: SequenceFile, MapFile, Avro Datafile. In this way, if only a small amount of data of the row needs to be accessed, the entire row needs to be read into the memory. Delaying the serialization can lighten the problem to a certain amount, but the overhead of reading the whole row of data.

- Fields of a row are stored contiguously

Benefits:

- Quick and easy:

— Retrieve an entire row
— Insert, update

Drawbacks:

- Without indexing, filtering is slower
- Entire row has to be read even if we only need a few columns

Column-oriented

The entire file cut into several columns of data, and each column of data stored together: Parquet, RCFile, ORCFile. The column-oriented format makes it possible to skip unneeded columns when reading data, suitable for situations where only a small portion of the rows are in the field. But this format of reading and write requires more memory space because the cache line needs to be in memory .

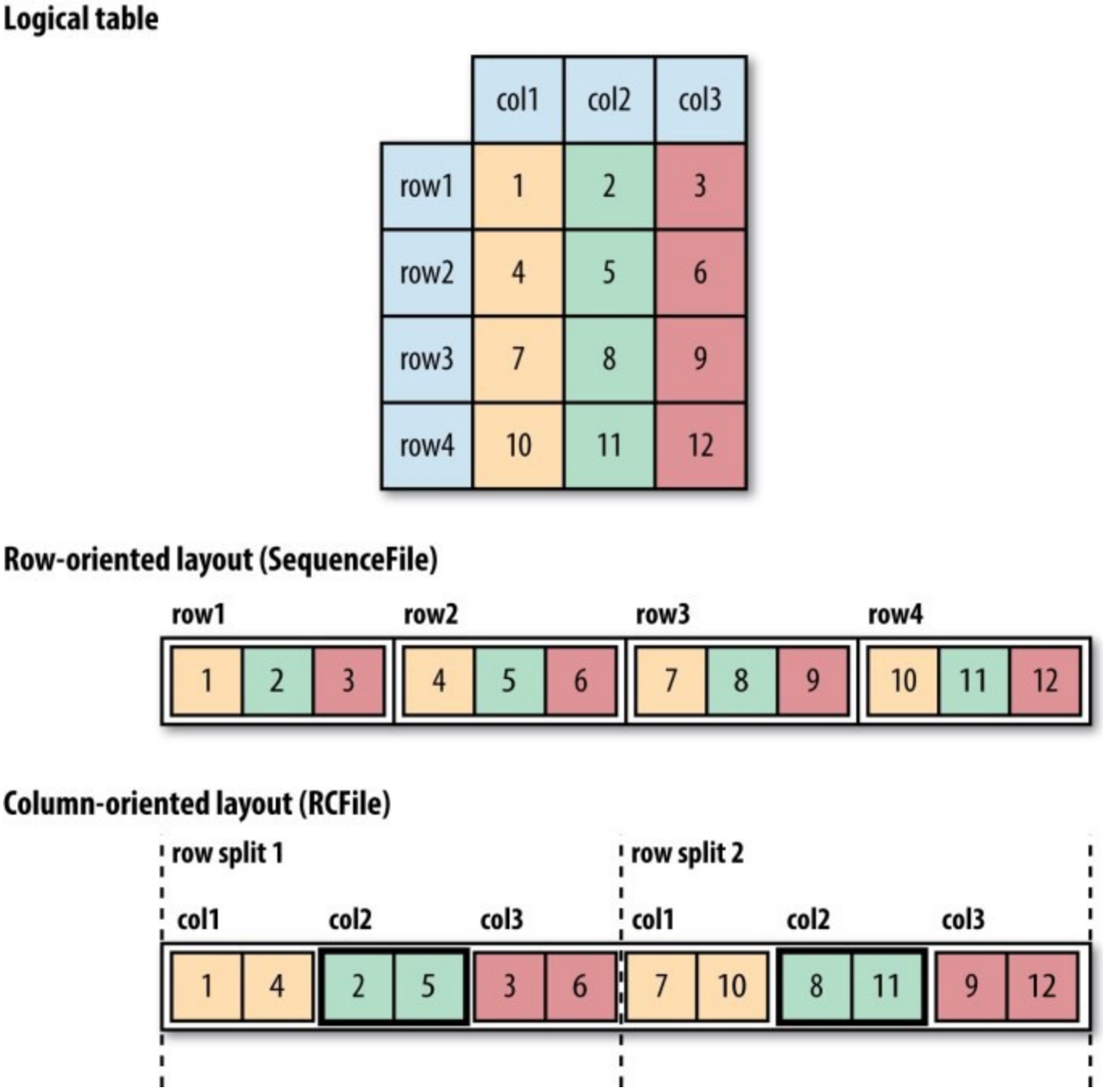
- Fields of a column are stored contiguously

Benefits:

- Each column can serve as an index (fast filtering operations on the whole dataset)
- Only selected columns are read

Drawbacks:

- Whole-row operations require a lot of disk I/O
- Slow and hard inserting and updating
- The same row can be stored on different nodes in a distributed environment

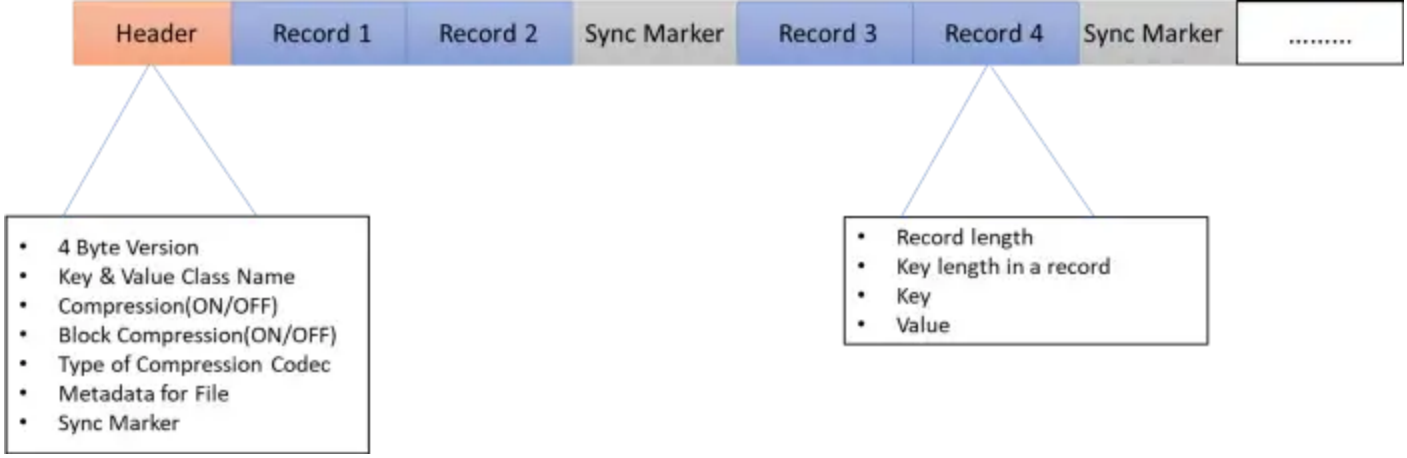


File Storage formats can be broadly classified into two categories —

1. *Traditional or Basic File Formats* — Text(CSV/JSON), Key-Value or Sequence File Format.
2. *Hadoop Specific File Formats* — Avro, Parquet, RC(Row Columnar) or ORC(Optimized Row Columnar) Format.

Sequence File Format

A Sequence file is a key-value binary file format that has a structure similar to a CSV file with differences. Each row can be delimited and split but due to its binary format, it is used as intermediate storage. Sequence files are more compact than text files.



Advantage:

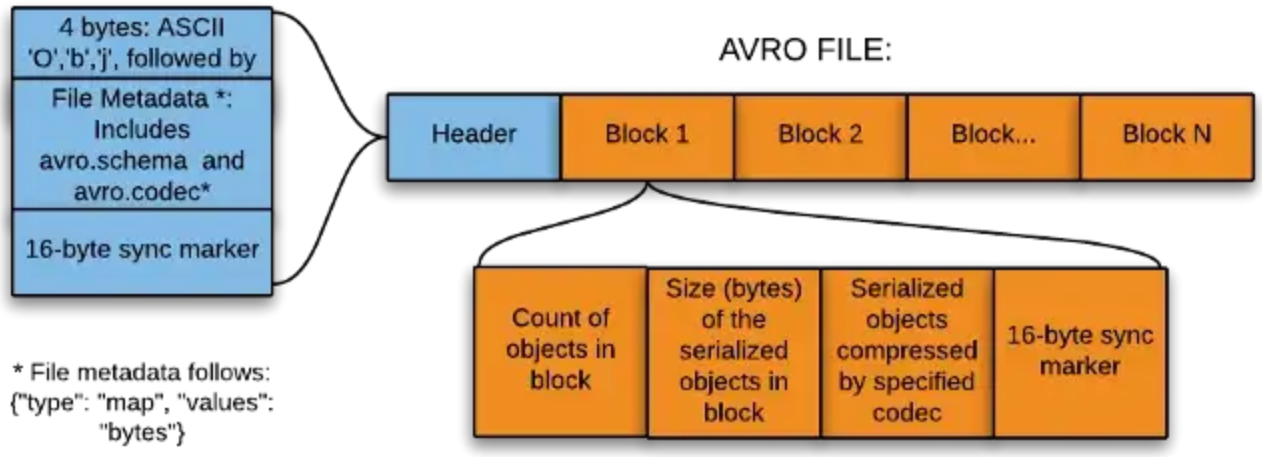
- Files can be split and merged easily. Supports merging of small files.
- Supports record or block compression.

Drawback:

- Data is not in a readable format.
- Limited support for schema evolution as changing the order of fields is not possible but appending new fields at the end of the line is plausible.

Avro

Avro is framework developed within Apache's Hadoop project. It is a row-based storage format which is widely used as a serialization process. AVRO stores its schema in JSON format making it easy to read and interpret by any program. The data itself is stored in binary format by doing it compact and efficient. A key feature of AVRO is related to the fact that it can easily handle schema evolution. It attach metadata into their data in each record.



Feature & characteristics:

- Avro is a row-based format file.
- Avro has contained 2 parts, schema which is in JSON, and data which is in binary
- Avro is designed to support Schema Evolution because it has schema in itself, thus the program is able to handle schema changes easily.
- Arvo is supported by compression and splittable, thus it can be used in the Big Data system.

Advantage:

- Avro is a data serialization system.
- It is splittable (AVRO has a sync marker to separate the block) and compressible.
- Avro is good file format for data exchange. It has a data storage which is very compact, fast and eficient for analytics.
- It highly supports schema evolution (at different time and independently).
- It supports batch and is very relevant for streaming.
- Avro schemas defined in JSON, easy to read and parse.
- The data is always accompanied by schema, which allow full processing on the data.

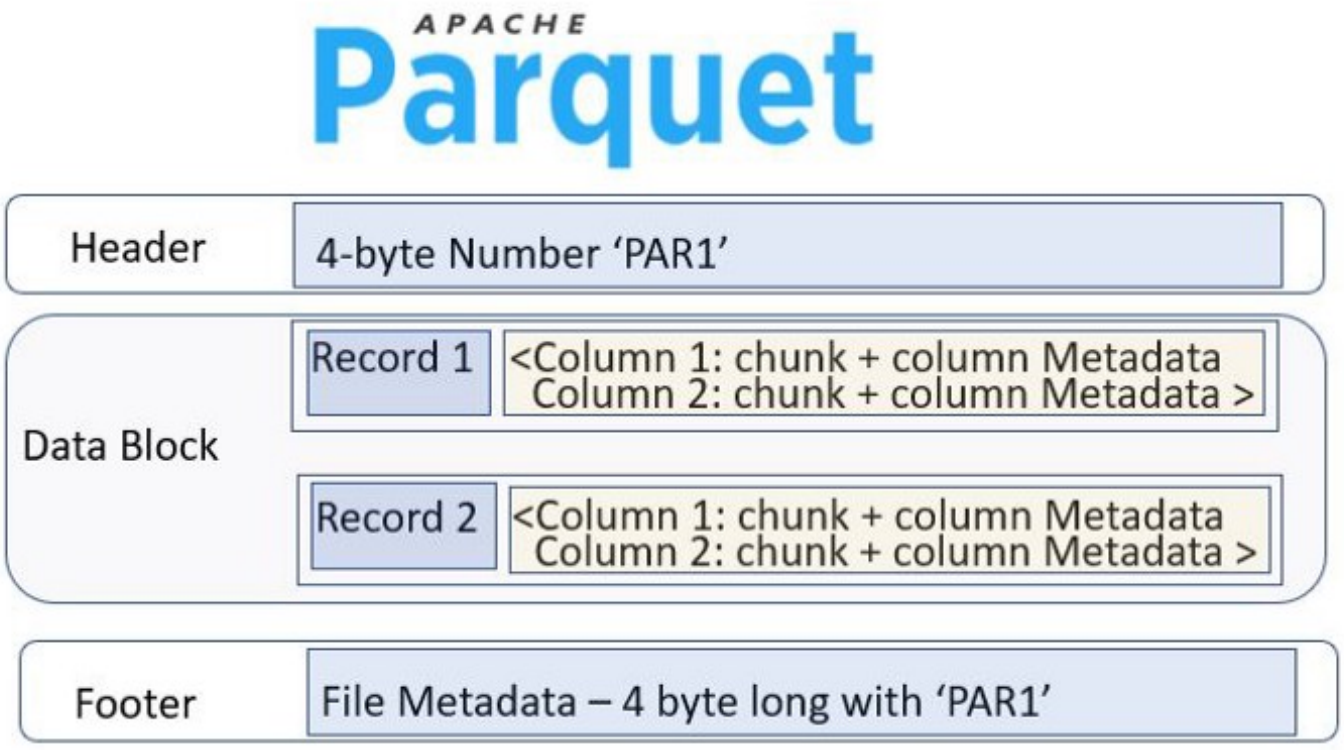
Drawback:

- Its data is not readable by human.

- Not integrated to every languages.

Parquet

Parquet is an open source file format for Hadoop ecosystem. Its design is a combined effort between Twitter and Cloudera for an efficient data storage of analytics. Parquet stores data by column-oriented like ORC format. It provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk. Parquet is a binary file containing metadata about their content. The column metadata is stored at the end of the file, which allows for fast, one-pass writing. Parquet is optimized for the paradigm of write once read many (WORM).



Parquet file format is broadly classified as three component, namely

- Parquet Header
- Data Block
- Parquet Footer

Feature & characteristics:

- Parquet is the columnar based format.
- It is suitable for read-heavy workloads, it can query fast because it is column-based.
- Parquet is supporting data compression.
- Parquet can be expensive for schema evolution. Because, in order to figure out schema, you basically have to read all of your parquet files and reconcile/merge their schemas during the reading time which can be expensive depending on how many files or/and how many columns are in there in the dataset.
- Parquet also supports complex data types and nested data structure
- Suitable when using Apache Spark because it can boost processing performance greatly.

Advantage:

- Splittable files.
- Organizing by column, allowing a better compression, as data is more homogeneous.
- High efficiency for OLAP workload.
- Supports schema evolution.
- Restricted to batch processing.

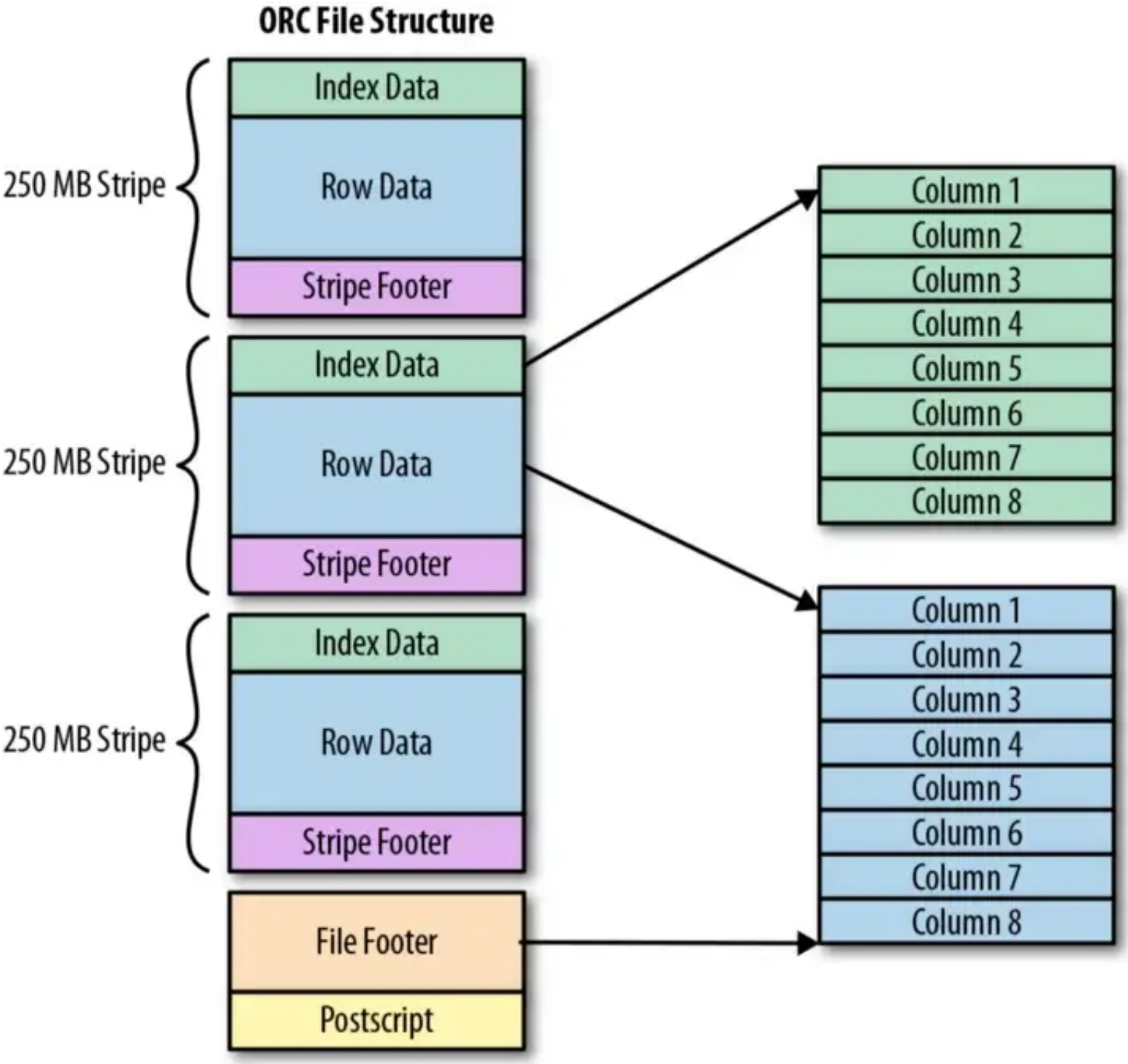
Drawback:

- Not human readable.
- Difficulties to apply updates unless you delete and recreate it again.

ORC(Optimized Row Columnar)

ORC format was created and open sourced by Hortonworks in collaboration with Facebook. It is a column-oriented data storage format similar to Parquet. ORC files contain groups of row data called stripes, along with auxiliary information in a file footer. At the end of the file, a postscript

holds compression parameters and the size of the compressed footer. The default stripe size is 250 MB. Large stripe sizes enable large efficient reads from HDFS.



Feature & characteristics:

- ORC is **row-columnar format** (= just like Parquet)
- It is suitable for read-heavy workloads, it can query fast because it is column-based.
- Schema evolution can be expensive the same as the Parquet file.
- ORC is the best compression among these three types.

- Suitable when using Apache Hive.

Advantage:

- Highly compressible, reducing the size of the original data up to 75%.
- More efficient for OLAP than OLTP queries.
- Restricted to batch processing.

Drawback:

- Unable to add data without having to recreate the file.
- Does not support schema evolution.
- Impala currently does nt support ORC format.

ORC Vs Parquet

ORC	Parquet
ORC was inspired from the row columnar format which was developed by Facebook.	Parquet was inspired from the nested data storage format outlined in the Google Dremel paper.
ORC is developed by Hortonworks in collaboration with Facebook.	Parquet is developed and backed by Cloudera, in collaboration with Twitter.
ORC only supports Hive and Pig.	Parquet has a broader range of support for the majority of the projects in the Hadoop ecosystem.
ORC is better optimized for Hive.	Parquet works really well with Apache Spark.
ORC files are organized into stripes of data, which are the basic building blocks for data.	It stores data in pages and each page contains header information, information about definition levels and repetition levels, and the actual data.

Takeaways:

CSV: Good option for compatibility, spreadsheet processing and human readable data. ...

JSON: Heavily used in APIs. ...

Avro: Great for storing row data, very efficient. ...

Protocol Buffers: Great for APIs, especially for gRPC. ...

Parquet: Columnar storage. ...

ORC: Similar to Parquet, it offers better compression.

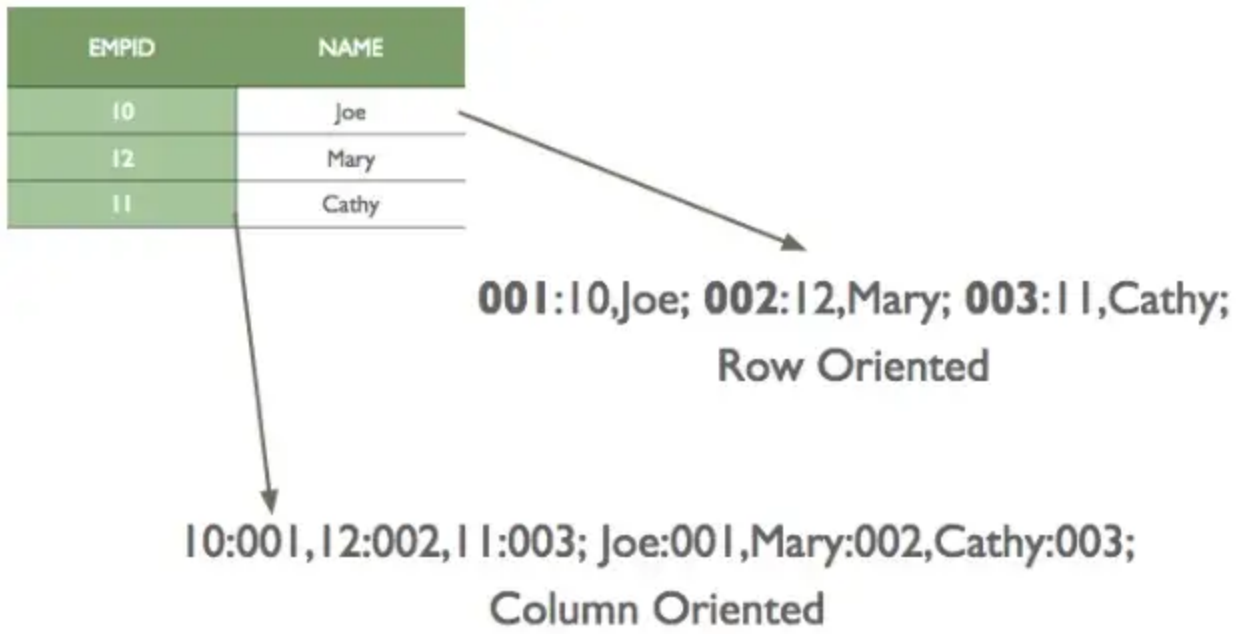
FAQs

What do we mean by Parquet?

Apache Parquet is a columnar format for storage of data available in Hadoop ecosystem. It is space efficient storage format which can be used in any programming language and framework. Apache Spark supports reading and writing data in Parquet format.

What does it mean by Row & Columnar wise Storage Format?

While converting a tabular or structured data into the stream of bytes we can either store row-wise or we could store column-wise.
In row-wise, we first store the first row and then store the second row and so on. In column-wise, we first store first column and second column.



Why ORC file format is faster?

ORC stands for Optimized Row Columnar which means it can store data in an optimized way than the other file formats. **ORC reduces the size of the original data up to 75%.** As a result the speed of data processing also increases and shows better performance than Text, Sequence and RC file formats.

How do I choose between ORC and Parquet?

While Parquet has a much broader range of support for the majority of the projects in the Hadoop ecosystem, ORC only supports Hive and Pig. One key difference between the two is that **ORC is better optimized for Hive, whereas Parquet works really well with Apache Spark.**

Which is the best file format for big data?

The Optimized Row Columnar (ORC) file format provides a highly efficient way to store data.

Is JSON good for big data?

Although JSON is referred to as comparatively better than CSV when dealing with massive data sets and in terms of scalability of files or applications, you should avoid this format when working with big data.

How large can a parquet file be?

The official Parquet documentation recommends a disk block/row group/file size of 512 to 1024 MB on HDFS.

How much does Parquet compress?

The final test, disk space results, are quite impressive for both formats: With Parquet, the 194GB CSV file was compressed to 4.7GB; and with Avro, to 16.9GB. That reflects an amazing 97.56% compression ratio for Parquet and an equally impressive 91.24% compression ratio for Avro.

Does Parquet store data type?

Parquet is a binary format and allows encoded data types. Unlike some formats, it is possible to store data with a specific type of boolean, numeric(int32, int64, int96, float, double) and byte array.

Is Parquet file compressed by default?

By default, in Hive, Parquet files are not written with compression enabled. However, writing files with Impala into a Parquet table will create files with internal Snappy compression (by default).

What compression does Parquet use?

Snappy and Gzip are the most commonly used ones and are supported by all implementations. LZ4 and ZSTD yield better results the former two but are a rather new addition to the format, so they are only supported in the newer versions of some of the implementations.

Are Parquet files immutable?

Historically Parquet files have been viewed as immutable, and for good reason. You incur significant costs for structuring, compressing and writing out a parquet file. It is better to append data via new parquet files rather than incur the cost of a complete rewrite.

Are Parquet files Splittable?

Yes, Parquet files are splittable. S3 supports positioned reads (range requests), which can be used to read only selected portions of the input file (object).

Can we zip Parquet files?

But you should not that storing Parquet files in a ZIP just for compression reasons is removing a lot of benefits of the Parquet format itself. By default Parquet is already compressed with the Snappy compression code (but you can also use GZip, ZStandard, and others).

Is parquet file structured or unstructured?

Avro and Parquet file formats are considered structured data as these can maintain the structure/schema of the data along with its data types.

Is gzip a splittable?

While GZIP offers the highest compression ratio, it is not splittable and takes longer to encode and decode. GZIP files can only use a single core to process.

Is Snappy a splittable?

According to this Cloudera post, Snappy IS splittable. For MapReduce, if you need your compressed data to be splittable, BZip2, LZO, and Snappy formats are splittable, but GZip is not. Splittability is not relevant to HBase data.

Why Parquet is best for Spark?

Parquet has higher execution speed compared to other standard file formats like Avro,JSON etc and it also consumes less disk space in compare to AVRO and JSON.

What is a SNAPPY file?

Snappy is a compression format and program library to implement it, introduced by Google. It is designed for very fast compression and decompression. Snappy is defined as a raw stream format, plus a higher-level “framing format” that can be used as a file format.

What is zlib library?

zlib is a software library used for data compression. zlib was written by Jean-loup Ga.

Is ORC file compressed?

The ORC file format provides the following advantages: Efficient compression: Stored as columns and compressed, which leads to smaller disk reads.

Which is best file format for Hive?

Using **ORC files** improves performance when Hive is reading, writing, and processing data comparing to Text,Sequence and Rc. RC and ORC shows better performance than Text and Sequence File formats.

What is the default storage format in hadoop?

Sequence file
Reason — high performance

What is default compression for parquet?

Snappy

What is default compression for orc?

Zlib

What is default file format for Spark?

Spark's default file format is Parquet. Parquet has a number of advantages that improves the performance of querying and filtering the data.

What is Protocol Buffers?

Protocol Buffers (Protobuf) is a free and open-source cross-platform data format used to serialize structured data. It is useful in developing programs to communicate with each other over a network or for storing data. The method involves an interface description language that describes the structure of some data and a program that generates source code from that description for generating or parsing a stream of bytes that represents the structured data.

Comparisons Between Different File Formats

AVRO vs. PARQUET

- AVRO is a row-based storage format, whereas PARQUET is a columnar-based storage format.
- PARQUET is much better for analytical querying, i.e., reads and querying are much more efficient than writing.
- Writiing operations in AVRO are better than in PARQUET.
- AVRO is much matured than PARQUET when it comes to schema evolution. PARQUET only supports schema append, whereas AVRO supports a much-featured schema evolution, i.e.,

adding or modifying columns.

- PARQUET is ideal for querying a subset of columns in a multi-column table. AVRO is ideal in the case of ETL operations, where we need to query all the columns.

ORC vs. PARQUET

- PARQUET is more capable of storing nested data.
- ORC is more capable of Predicate Pushdown.
- ORC supports ACID properties.
- ORC is more compression efficient.

Characteristics of Each File Formats

	Avro	Parquet	ORC
Schema Evolution	Best	Good	Better
Compression	Good	Better	Best
Splitability	Good	Good	Best
Row or Column	Row	Column	Column
Read or Write	Write	Read	Read

Characteristics of Each Compression Formats

Properties	Compression Format			
	gzip	bzip2	lzo	snappy
File extension	.gzip	.bz2	.lzo	.snappy
Compression level	Medium	High	Avargae	Average
Speed	Medium	Slow	Fast	Fast
CPU usage	Medium	High	Low	Low
Is Splittable	No	No	Yes, if indexed	No

File Size Comparison