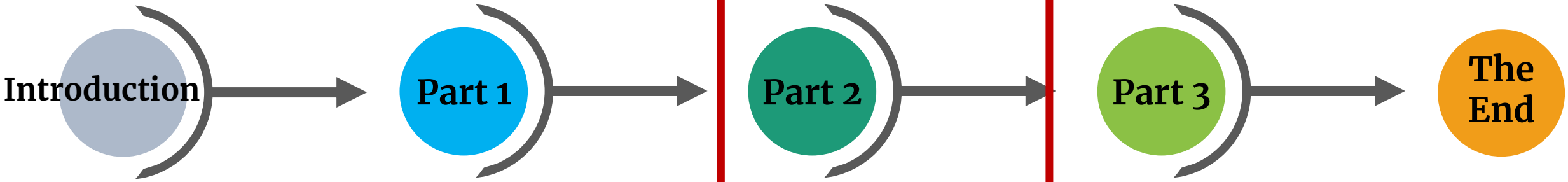
YCBS 257 – Data at Scale

Instructor: **Khaled Tannir PhD**

McGill | School of Continuing Studies

Course Learning Path (*CLP*)



Data at Scale Characteristics

- *Review Big Data 5Vs: Volume, Velocity, Variety, Veracity and Visualization*
- *Introducing Hadoop Ecosystem and Hadoop Distributions*

Data Storage & Processing

- *Learn data characteristics*
- *Learn how to store large volume datasets*
- *Learn to process large volume datasets*

Data Analysis

- *Learn to analyze large volume structured data format*
- *Learn to analyze large volume unstructured data format*
- *File Formats*

Data ingestion

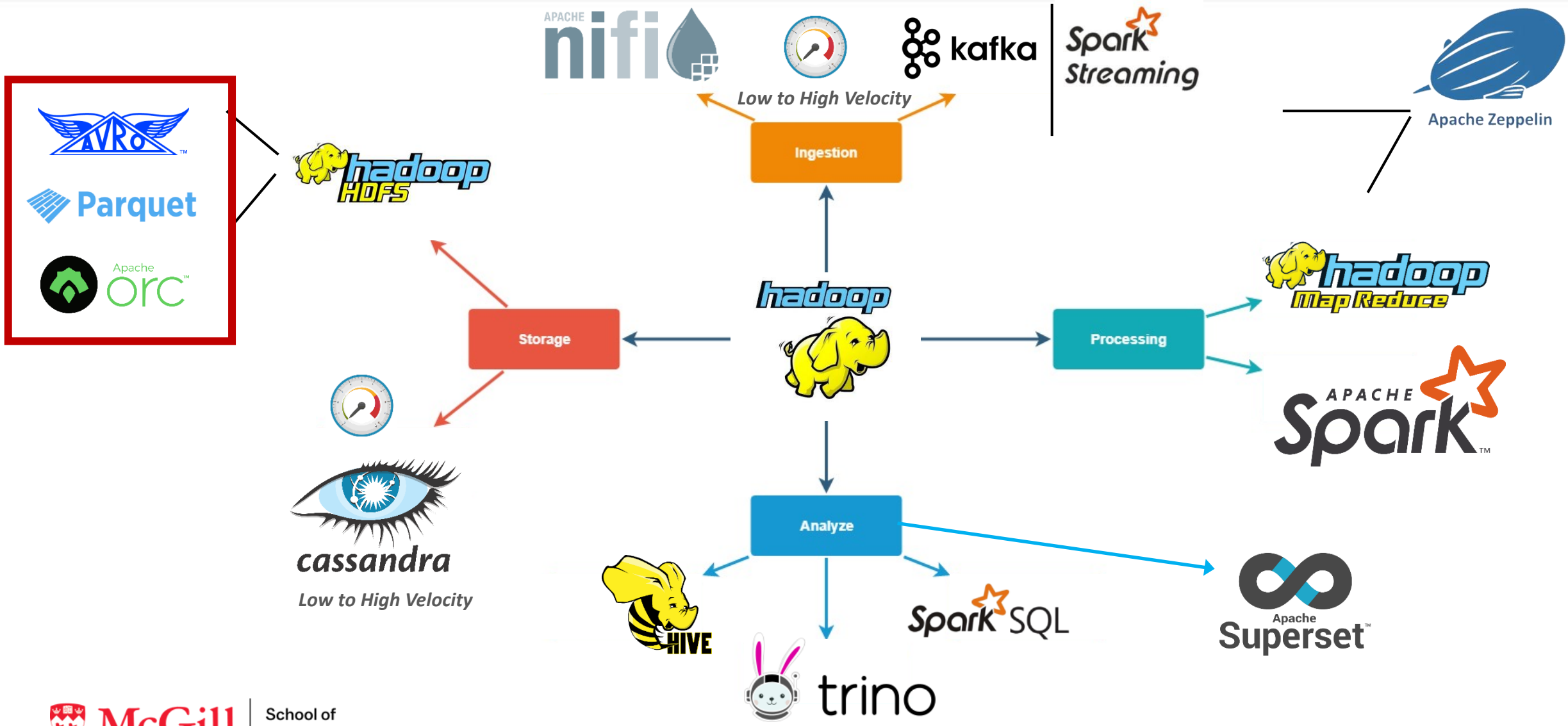
- *Learn to ingest on-demand structured data*
- *Lear to ingest real time structured and unstructured data*
- *Learn to collect events and capture data change (CDC)*

Volume

Veracity
Variety
Visualization

Velocity

Data at Scale Popular Technologies



Data at Scale Course

Part 1

Part 2

Volume

hadoop HDFS

Low Velocity — High Velocity

Distributed Storage

Volume

hadoop Map Reduce

APACHE Spark

On Disk — In-Memory

Distributed Processing

Variety

APACHE Spark

Spark SQL

HIVE

trino

AVRO

Apache orc

Parquet

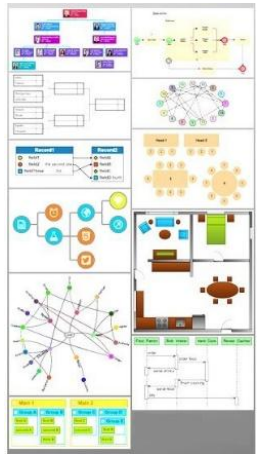
Unstructured — Structured

Visualization

Apache Superset

JAVA

Scala



PySpark

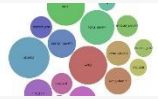
Spark MACHINE LEARNING

Theme of this Course

Data at Scale file Formats

- *What is a "good" file format*
- *Compatible Hadoop File Formats*
  - *Avro*
  - *Parquet*
  - *ORC*
- *Creating/Reading Avro, Parquet and ORC files*

Analyzing Data at Scale



- ***How to optimize files / Datasets size ?***

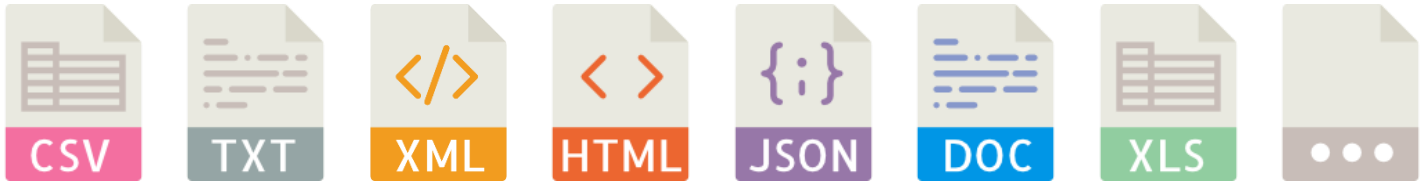
  How to reduce/optimize the storage space for a file / dataset?

- ***How to optimize querying time for a large dataset?***

  How to query a large dataset and get the result as fast as possible?

Which File Format to Choose

- CSV
- Text
- JSON
- Protobuff
- Thrift
- Sequence File
- Avro, Parquet, ORC …

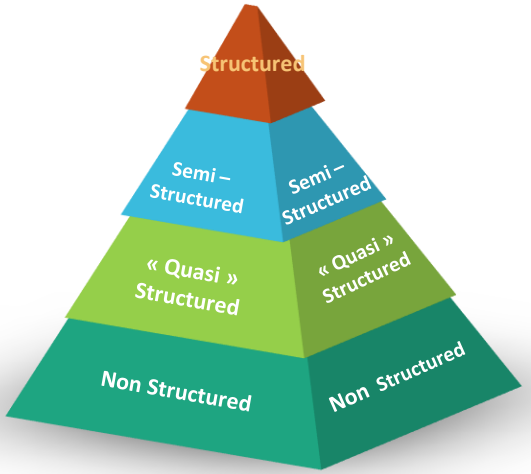


- Structured, semi/non-structured
- General usage
- Classified regarding their performances

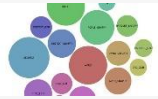
Parquet

AVRO™

Apache orc™



What is a 'Good' File Format?



Well-Defined

Optimized Binary Encoding

Expressive

Compressed (Natively)

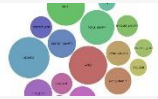
Simple

Splittable

(compatible with HDFS)

We should consider also file processing time, not only storage volume

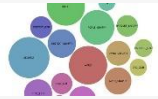
Well-Defined File Format



- Reading or analyzing a file should not be interrupted by a missing or miscoded value

```
movieId,title
55269,Darjeeling Limited, The
```

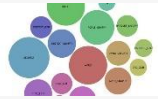
Expressive File Format



- **Reading data should be as easy as possible**

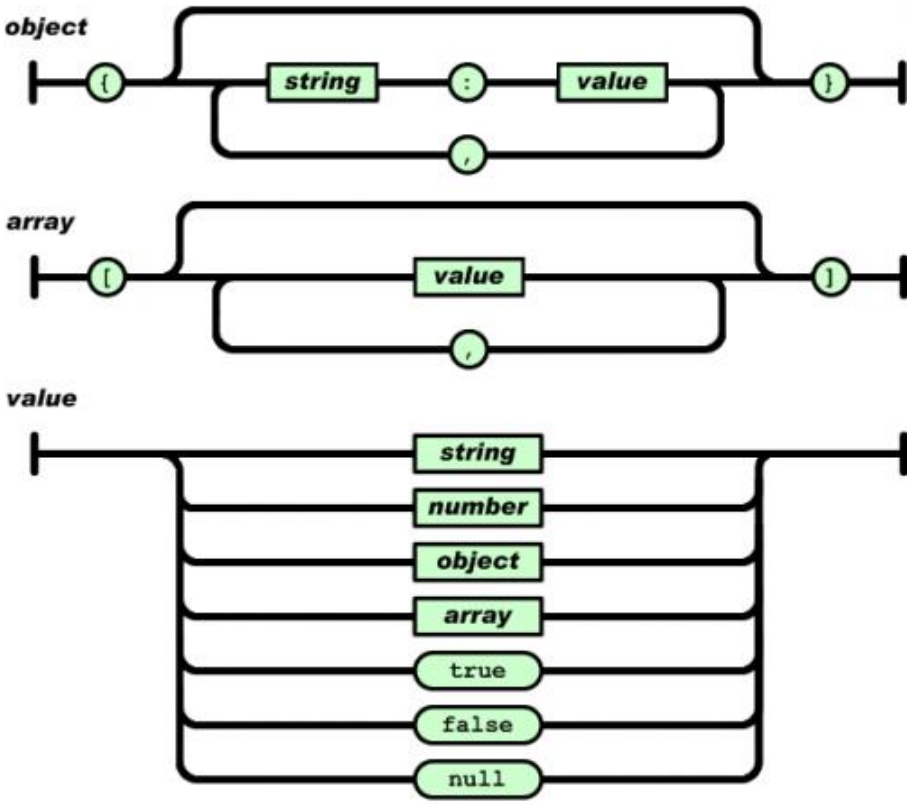
```
movieId,title,genres
94959,Moonrise Kingdom,Comedy|Drama|Romance

{"movieId": 94959, "title": "Moonrise Kingdom",
 "genres": ["Comedy", "Drama", "Romance"]}
```

Simple File Format

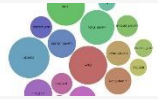


- **XML**: Well-Formed but **not** simple (definitions, namespaces, tags, …)
- **JSON**: is a **Well-Formed** format and **Simple**

JSON Definition



Optimized encoding File Format



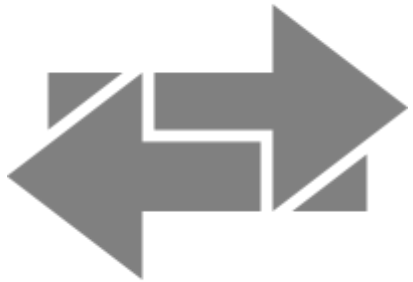
- **Everything should be encoded and written in byte to reduce storage size. (readable by machine not human)**

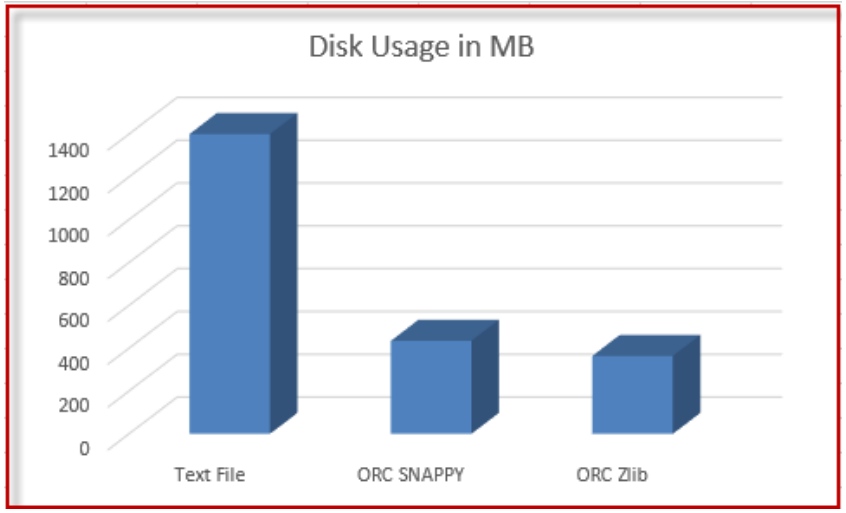
Encoding Example :

11034	ASCII - string size – 5 bytes
1a 2b 00 00	Big-endian Int encoding - 4 bytes
9a 56	zig-zag encoding - 2 bytes

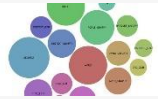
Compressed File Format

- Reduce the storage size
- Might increase processing time
- Should be also ‘Splitable’ to be Hadoop HDFS compliant

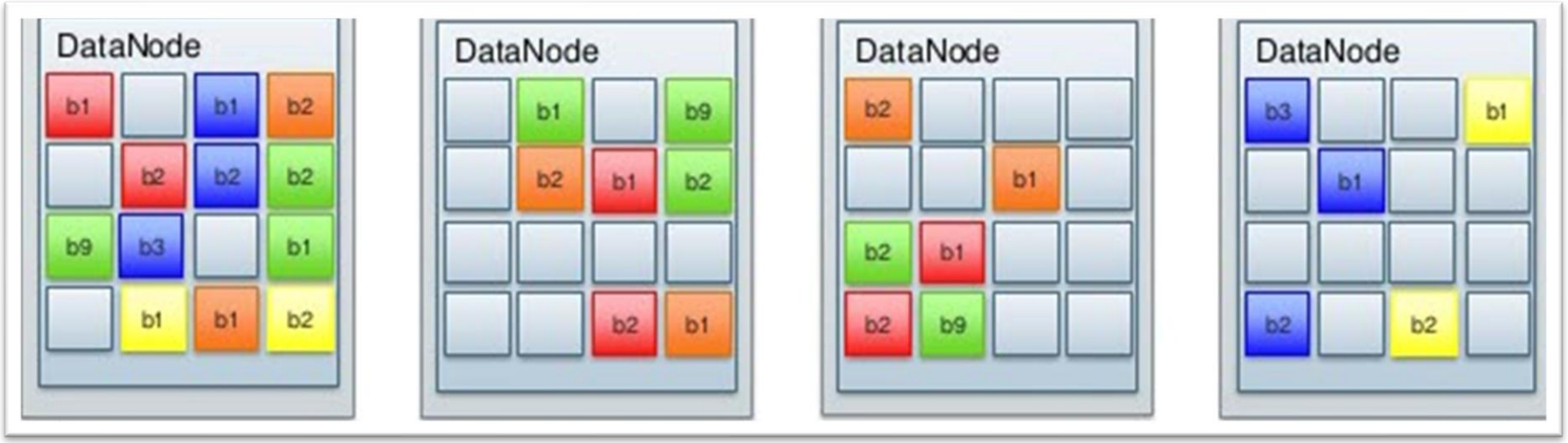




Splittable File Format



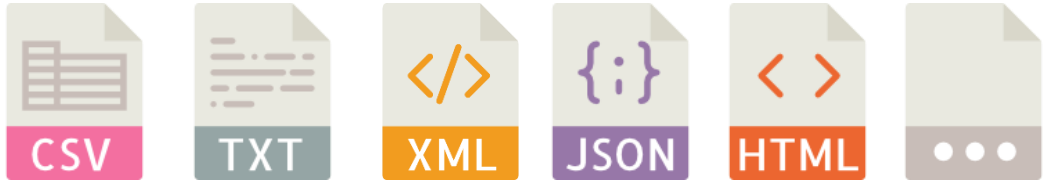
- Capability to process **independently** different chunks of a file
  - A chunk (split) should fit into a HDFS bloc size (usually 128MB)
- Allows processing **parallelization**



A Best File Format

- **Best file format for Data at Scale should be:**

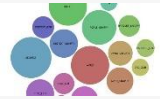
1. **Well-Defined**
2. **Expressive**
3. **Simple**
4. **Optimized Binary enconding**
5. **Compressed (natively)**
6. **Splittable**



Parquet

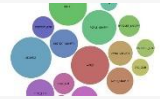
ORC

Files Formats Summary



	Well-Defined	Expressive	Simple	Binary	Compressed	Splittable
CSV	✓	✗	✓	✗	✗	✗
TEXT	✓−	✗	✓	✗	✗	✓
JSON	✓	✓	✓	✗	✗	✗
XML	✓	✓−	✗	✗	✗	✗
THRIFT	✓	✓−	✓	✓	✓	✗
SEQ FILES	✗	−	✗	✓	✓	✓
AVRO	✓	✓	✓	✓	✓	✓
Parquet/ORC	✓	✓	✓−	✓+	✓	✓

AVRO Format



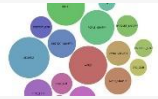
	Well-Defined	Expressive	Simple	Binary	Compressed	Splittable
AVRO	✓	✓	✓	✓	✓	✓

- A good file format
  - Avro is a data serialization system
  - Row Oriented
  - Need a Schema
  - Flexible object model
  - Implemented in C, C++, C#, Java, Perl, PHP, Python, and Ruby

Avro schema example

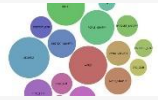
```
{
 "namespace": "customer.avro",
 "type": "record",
 "name": "customer",
 "fields": [
    {"name": "name",    "type": "string"},
    {"name": "age",     "type": "int"},
    {"name": "address", "type": "string"},
    {"name": "phone",   "type": "string"},
    {"name": "email",   "type": "string"},
 ]
}
```

Avro Provides



- High compatibility with Hadoop
- Rich data structures
- Compact, fast, binary data format
- A container file to store persistent data
- Remote Procedure Call (RPC)
- Simple integration with dynamic languages

AVRO Benefits

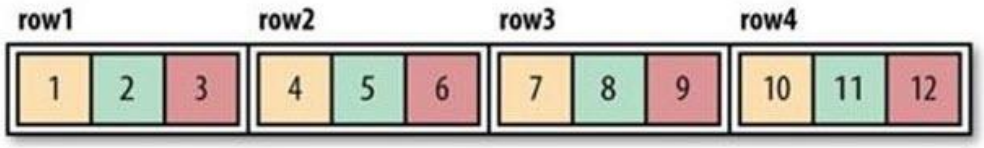


- Write and forget
- Low memory usage
- Ensure that the data is written (especially in case of failure)

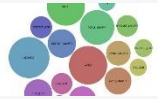
Logical table

	col1	col2	col3
row1	1	2	3
row2	4	5	6
row3	7	8	9
row4	10	11	12

Row-oriented layout (SequenceFile)



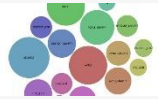
Parquet Format



	Well-Defined	Expressive	Simple	Binary	Compressed	Splittable
Parquet	✓	✓	✓−	✓+	✓	✓

- A Very good File Format
  - Column Oriented
  - Encode a group of values rather than one value (simplicity ☹)
  - Unlike ORC, Parquet is commonly used outside Hive
  - Need a Schema
  - Based off the Google Dremel paper

Parquet Benefits



- Very fast reading
  - Read selective columns
- Small encoding data size (reduce storage volume)
- Supports very efficient compression and encoding schemes

Parquet

Table

ID	Name	Age	Department
10	Bean	27	Production
11	Willis	61	Finance
12	Ferguson	58	Production

Column-Oriented layout

10	11	12	Bean	Willis	Ferguson	27	61	58	Production	Finance	Production

Column Oriented Format

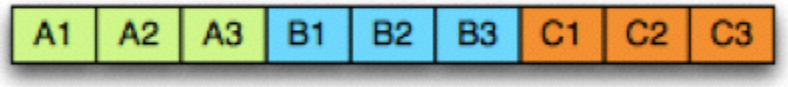
- Need to ‘wait’ all the rows to finalize a column
- Memorize rows into groups
- High memory usage
- In case of failure data is not guarantied to be written

Parquet

Table

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

column-oriented layout



ORC Format

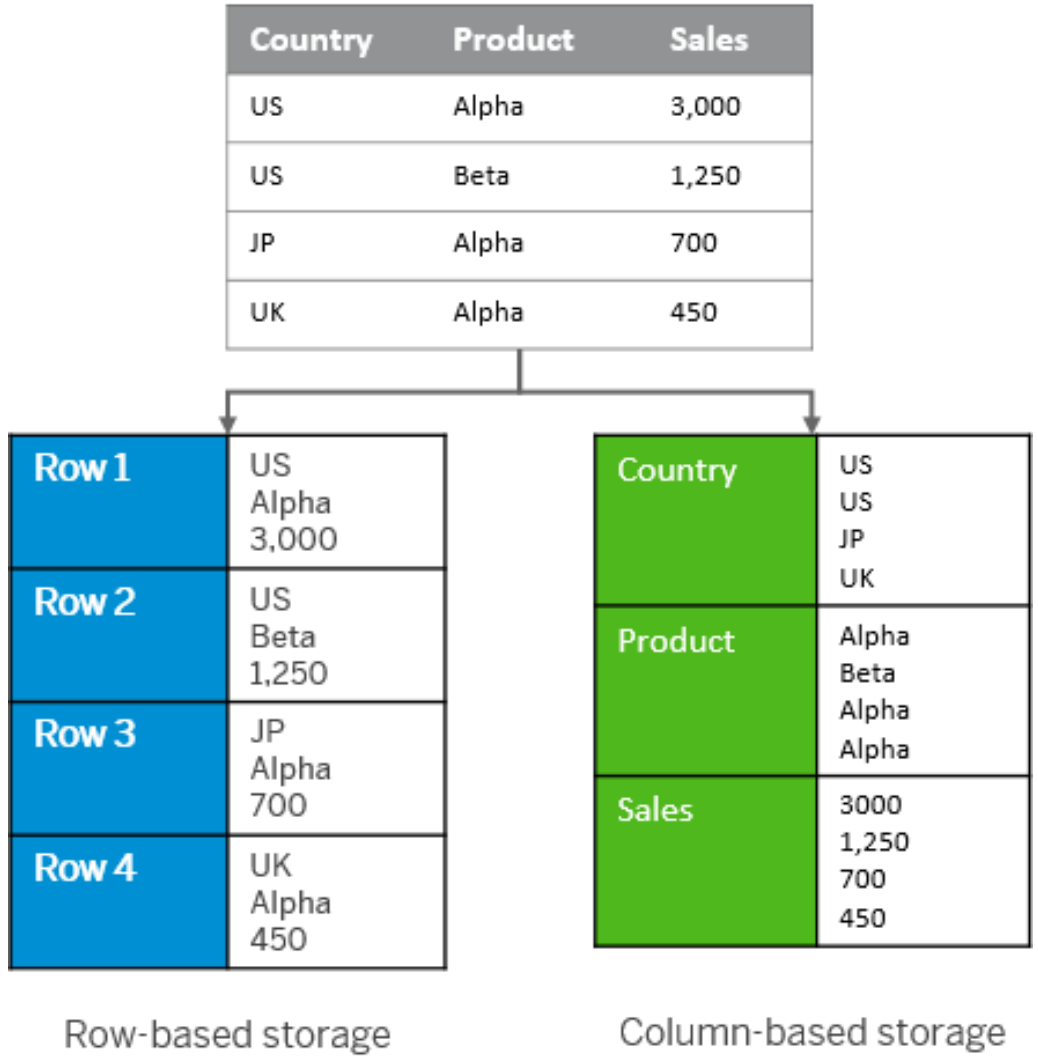
	Well-Defined	Expressive	Simple	Binary	Compressed	Splittable
ORC	✓	✓	✓−	✓+	✓	✓

- A Very good File Format Similar to Parquet

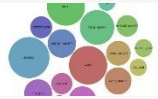
  - Column Oriented
  - Encode a group of values rather than one value (simplicity ☹)
  - ORC is used internally by Hive
  - Need a Schema

ORC Benefits

- Suitable for read-heavy workloads
- Read selective columns
- ORC has best compression ratio
- Natively supported by Hive



Creating Avro / Parquet / ORC Files



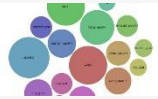
- **Step 1 – Create schema. You need to design JSON schema according to your data**
  *(Same schema is used for both Avro/Parquet. e.g: **schema.avsc**)*

- **Step 2 – Create an empty container based on your schema**
  (A physical file - can be used to populate Hive tables)

- **Step 3 – Serialize (import) the data into your container**
  (You can use Hive / Impala or external tools, API to serialize your data)

- **Step 4 – Read the data using your deserialization API provided for Avro/Parquet**
  (Hive, Impala, Spark, Pig, Sqoop, Nifi, Scala, Java, Python…)

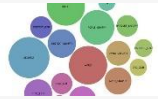
AVRO / PARQUET Schema



- **Schema is in JSON format and support different types:**
  - **Primitives**
    (*boolean, int, long double, string, fixed, binary*)
  - **Unions**
    (with ***null*** for optional data)
  - **Records**
    (for complexes objects)
  - **Maps and Lists**

Data type	Description
null	Null is a type having no value.
int	32-bit signed integer.
long	64-bit signed integer.
float	single precision (32-bit) IEEE 754 floating-point number.
double	double precision (64-bit) IEEE 754 floating-point number.
bytes	sequence of 8-bit unsigned bytes.
string	Unicode character sequence.

Creating Avro / Parquet Schemas

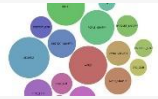


- **Needed to decode data**
- **Stored in the file header**

```
{
  "type" : "record",
  "namespace" : “YCBS257",
  "name" : "Employee",
  "fields" : [
    { "name" : "Name" , "type" : "string" },
    { "name" : "Age" , "type" : "int" }
  ]
}
```

- **type** – shows the type of the document, generally a record because there are multiple fields.
- **namespace** – This field describes the name of the namespace in which the object resides.
- **name** – describes the schema name. This schema name together with the namespace, uniquely identifies the schema within the store.
- **Fields** – describes names and types of the fields.

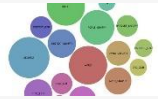
Creating Avro / Parquet / ORC Tables



- **Hive support creating tables stored in different formats:**
  - **Create Table ….. Stored as Avro**
  - **Create Table ….. Stored as Parquet**
  - **Create Table ….. Stored as ORC**

- **Compression : (most common)**
  - **SNAPPY, GZib, Zlib, Deflate**

Creating Hive Tables

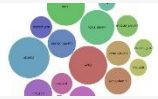


- Create table ….. Stored AS Avro / Parquet / ORC
- TBLPROPERTIES = > Compression Algorithm to use
- Populate the table with data using Load or Insert Into statements

Example:

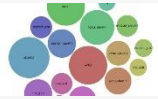
```
CREATE TABLE data_avro (start_date string, pk_start int, end_date string, pk_end
int, duration_sec int, is_member tinyint)
STORED AS AVRO TBLPROPERTIES ('AVRO.COMPRESS'='SNAPPY');


INSERT INTO TABLE data_avro SELECT * FROM data;
```

File Formats Comparison



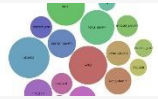
	Avro	Parquet	ORC
Schema Evolution	Best	Good	Better
Compression	Good	Better	Best
Splitability	Good	Good	Best
Row or Column	Row	Column	Column
Read or Write	Write	Read	Read

Compression Codecs Comparison



Format	Algorithm	Strategy	Emphasis	Comments
zlib	Uses DEFLATE (LZ77 and Huffman coding)	Dictionary-based, API	Compression ratio	Default codec
gzip	Wrapper around zlib	Dictionary-based, standard compression utility	Same as zlib, codec operates on and produces standard gzip files	For data interchange on and off Hadoop
bzip2	Burrows-Wheeler transform	Transform-based, block-oriented	Higher compression ratios than zlib	Common for Pig
LZO	Variant of LZ77	Dictionary-based, block-oriented, API	High compression speeds	Common for intermediate compression, HBase tables
LZ4	Simplified variant of LZ77	Fast scan, API	Very high compression speeds	Available in newer Hadoop distributions
Snappy	LZ77	Block-oriented, API	Very high compression speeds	Came out of Google, previously known as Zippy

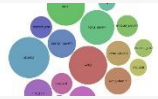
Avro / Parquet Common Usage



Data At Rest

Data In Motion

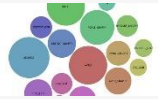
AVRO Toolbox



- **avro-tools (Apache)**

```
[cloudera@quickstart ml-20m]$ avro-tools
Version 1.7.6-cdh5.8.0 of Apache Avro
Copyright 2010 The Apache Software Foundation

C JSON parsing provided by Jansson and
written by Petri Lehtinen. The original software is
available from http://www.digip.org/jansson/.
----------------
Available tools:
          cat  extracts samples from files
      compile  Generates Java code for the given schema.
       concat  Concatenates avro files without re-compressing.
    fragtojson  Renders a binary-encoded Avro datum as JSON.
      fromjson  Reads JSON records and writes an Avro data file.
      fromtext  Imports a text file into an avro data file.
       getmeta  Prints out the metadata of an Avro data file.
    getschema  Prints out schema of an Avro data file.
           idl  Generates a JSON schema from an Avro IDL file
 idl2schemata  Extract JSON schemata of the types from an Avro IDL file
       induce  Induce schema/protocol from Java class/interface via reflection.
    jsontofrag  Renders a JSON-encoded Avro datum as binary.
       random  Creates a file with randomly generated instances of a schema.
      recodec  Alters the codec of a data file.
       repair  Recovers data from a corrupt Avro Data file
  rpcprotocol  Output the protocol of a RPC service
   rpcreceive  Opens an RPC Server and listens for one message.
      rpcsend  Sends a single RPC message.
       tether  Run a tethered mapreduce job.
       tojson  Dumps an Avro data file as JSON, record per line or pretty.
       totext  Converts an Avro data file to a text file.
     totrevni  Converts an Avro data file to a Trevni file.
  trevni_meta  Dumps a Trevni file's metadata as JSON.
trevni_random  Create a Trevni file filled with random instances of a schema.
trevni_tojson  Dumps a Trevni file as JSON.
[cloudera@quickstart ml-20m]$
```

Parquet Toolbox

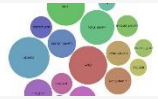


Parquet

- parquet-tools (Apache)

```
[cloudera@quickstart ~]$ parquet-tools -h
usage: parquet-tools cat [option...] <input>
where option is one of:
       --debug     Enable debug output
    -h,--help      Show this help string
    -j,--json      Show records in JSON format.
       --no-color  Disable color output even if supported
where <input> is the parquet file to print to stdout

usage: parquet-tools head [option...] <input>
where option is one of:
       --debug          Enable debug output
    -h,--help           Show this help string
    -n,--records <arg>  The number of records to show (default: 5)
       --no-color       Disable color output even if supported
where <input> is the parquet file to print to stdout

usage: parquet-tools schema [option...] <input>
where option is one of:
    -d,--detailed  Show detailed information about the schema.
       --debug     Enable debug output
    -h,--help      Show this help string
       --no-color  Disable color output even if supported
where <input> is the parquet file containing the schema to show

usage: parquet-tools meta [option...] <input>
where option is one of:
       --debug     Enable debug output
    -h,--help      Show this help string
       --no-color  Disable color output even if supported
where <input> is the parquet file to print to stdout

usage: parquet-tools dump [option...] <input>
where option is one of:
    -c,--column <arg>  Dump only the given column, can be specified more than
                       once
    -d,--disable-data  Do not dump column data
       --debug         Enable debug output
    -h,--help          Show this help string
    -m,--disable-meta  Do not dump row group and page metadata
       --no-color      Disable color output even if supported
where <input> is the parquet file to print to stdout
[cloudera@quickstart ~]$
```

ORC Toolbox



- **orc-tools (Apache)**

The subcommands for the tools are:

- convert (since ORC 1.4) - convert JSON/CSV files to ORC
- count (since ORC 1.6) - recursively find *.orc and print the number of rows
- data - print the data of an ORC file
- json-schema (since ORC 1.4) - determine the schema of JSON documents
- key (since ORC 1.5) - print information about the encryption keys
- meta - print the metadata of an ORC file
- scan (since ORC 1.3) - scan the data for benchmarking
- sizes (since ORC 1.7.2) - list size on disk of each column
- version (since ORC 1.6) - print the version of this ORC tool

Questions?

McGill | School of Continuing Studies

It’s time for a break

Grab some coffee, We’ll be back in 15min

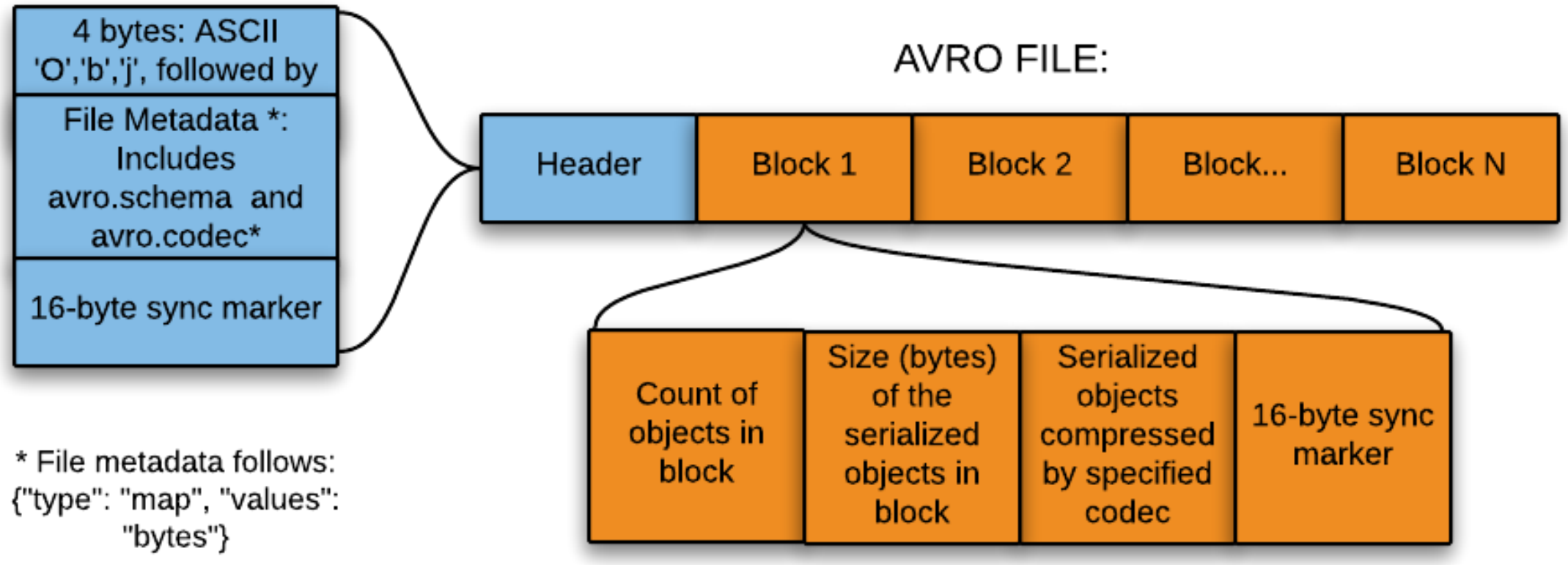
McGill | School of Continuing Studies

File Formats Workshop

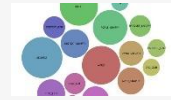
McGill | School of Continuing Studies

Appendix

AVRO file Structure

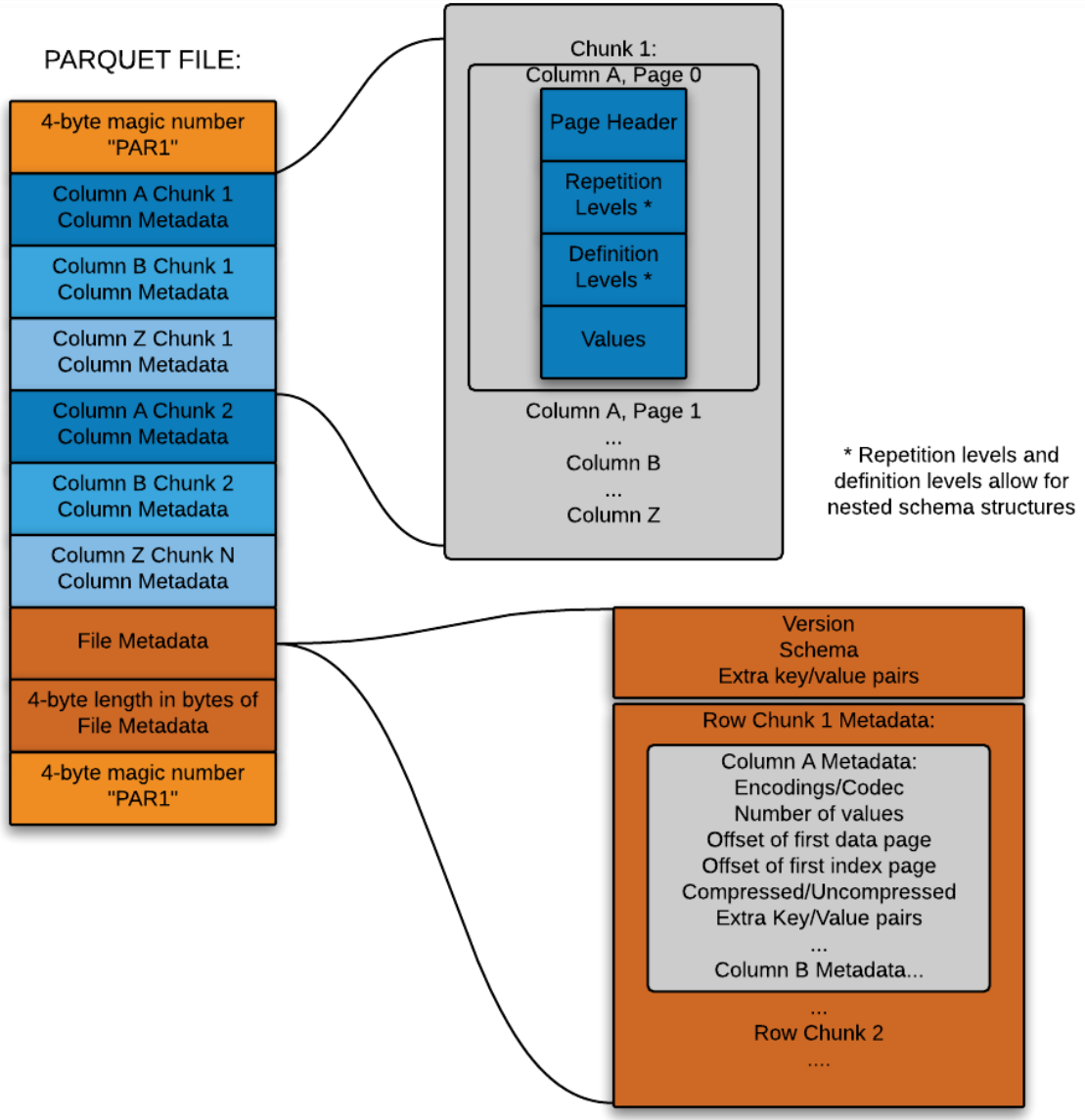


Parquet File Structure

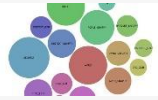


- Data is splitted into blocs (chunks)
- Bloc size is usually based on the HDFS bloc size
- Accumulates encoded data
- No file markers as in Avro

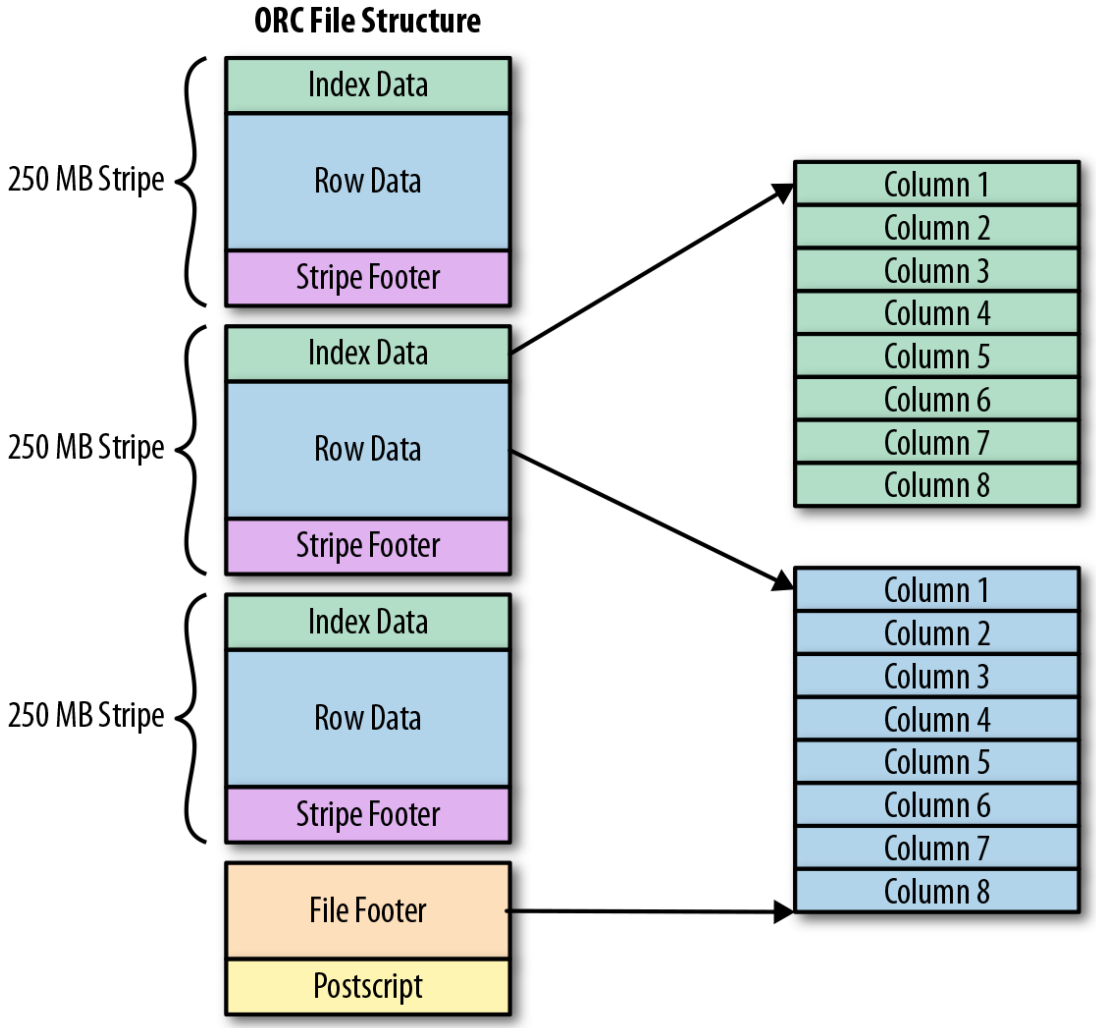
  No wasted space



ORC File Structure



- **Efficient compression**
- **Fast reads: ORC has a built-in index, min/max values, and other aggregates that cause entire stripes to be skipped during reads.**
- **Proven in large-scale deployments**



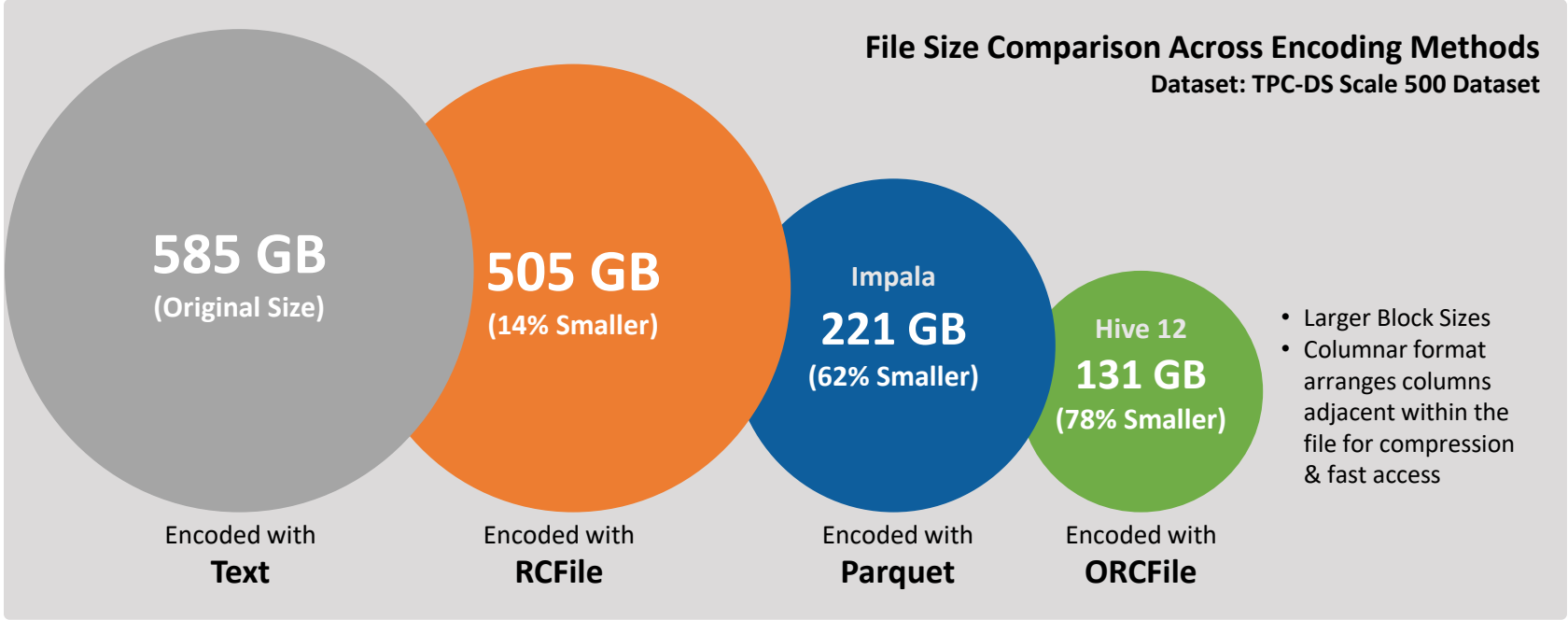
Interactive Query at Scale

Sustained Query Times

Apache Hive provides sustained acceptable query times even at petabyte scale

Smaller Footprint

Better encoding with ORC in Apache Hive reduces resource requirements for your cluster



Merci!

Thank You!

McGill | School of Continuing Studies