PARQUET & AVRO

Presenter Introduction

- Tim Spann, Senior Solutions Architect, airis.DATA
  - ex-Pivotal Senior Field Engineer
  - DZONE MVB and Zone Leader
  - ex-Startup Senior Engineer / Team Lead

- http://www.slideshare.net/bunkertor
- http://sparkdeveloper.com/
- http://www.twitter.com/PaasDev

Presenter Introduction

- Srinivas Daruna

  Data Engineer, airis.DATA

  Spark Certified Developer

airis.DATA

airis.DATA is a next generation system integrator that specializes in rapidly deployable machine learning and graph solutions.

Our core competencies involve providing modular, scalable Big Data products that can be tailored to fit use cases across industry verticals.

We offer predictive modeling and machine learning solutions at Petabyte scale utilizing the most advanced, best-in-class technologies and frameworks including **Spark**, **H20**, **Mahout**, and **Flink**.

Our data pipelining solutions can be deployed in batch, real-time or near-real-time settings to fit your specific business use-case.

Agenda

- ❑ Avro and Parquet - When and Why to use which format?
- ❑ Use cases for Schema Evolution & practical examples
- ❑ Data modeling - Avro and Parquet schema
- ❑ Workshop
  - Read Avro input from Kafka
  - Transform data in Spark
  - Write data frame to Parquet
  - Read back from Parquet
- ❑ Our experiences with Avro and Parquet
- ❑ Some helpful insights for projects design

AVRO - Introduction

- Doug Cutting created Avro, a data serialization and RPC library, to help improve data interchange, interoperability, and versioning in Hadoop Eco System.
- Serialization & RPC Library and also storage format.
- What led to a new serialization mechanism.?
- Thrift and PB are not splittable and codegen required for both of them. Dynamic reading is not possible
- Sequence files does not have schema evolution
- Evolved as in-house serialization and RPC library for hadoop. Good overall performance that can match up to Protocol Buffers in some aspects.

Some Important features of AVRO

- **Dynamic Access** – No need of Code generation for accessing the data.
- **UnTagged Data** – Which allows better compression
- **Platform in-dependent** – Has libraries in Java, Scala, Python, Ruby, C and C#.

Compressible and Splittable – Complements the parllel processing systems such as MR and Spark.

- **Schema Evolution:** "Data models evolve over time", and it's important that your data formats support your need to modify your data models. Schema evolution allows you to add, modify, and in some cases delete attributes, while at the same time providing backward and forward compatibility for readers and writers

Summary of AVRO Properties

- Row Based
- Direct mapping from/to JSON
- Interoperability: can serialize into Avro/Binary or Avro/Json
- Provides rich data structures
- Map keys can only be strings (could be seen as a limitation)
- Compact binary form
- Extensible schema language
- Untagged data
- Bindings for a wide variety of programming languages
- Dynamic typing
- Provides a remote procedure call
- Supports block compression
- Avro files are splittable
- Best compatibility for evolving data schemas

AVRO Schema Types

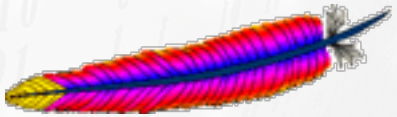
Primitive Types

`null`: no value
`boolean`: a binary 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

Primitive types have no specified attributes.
Primitive type names are also defined type names. Thus, for example, the schema "string" is equivalent to: {"type": "string"}

Complex Types

Records
Enums
Arrays
Maps
Unions
Fixed



https://avro.apache.org/docs/current/spec.html

Avro Schema

Understanding Avro schema is very important for Avro Data.

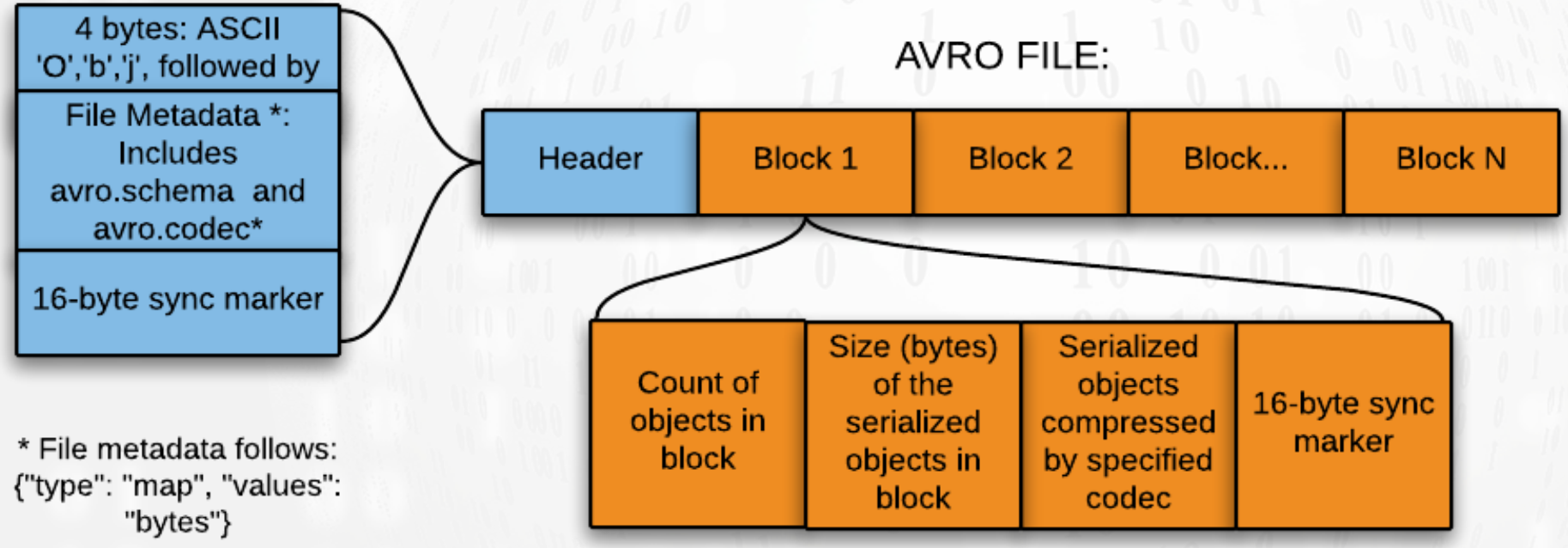
- JSON Format is used to define schema
- Simpler than IDL(Interface Definition Language) of Protocol Buffers and thrift
- very useful in RPC. Schemas will be exchanged to ensure the data correctness
- You can specify order (Ascending or Descending) for fields.

Sample Schema:

```
{
    "type": "record",
    "name": "Meetup",
    "fields": [ {
        "name": "name",
        "type": "string”,
        "order" : "descending"
    }, {
        "name": "value",
        "type": ["null", "string”]
    }
    …..
    ]
}
```

Union

File Structure - Avro



Workshop - Code Examples

- Java API to create Avro file - API Support
- Hive Query to create External table with Avro Storage Format – Schema Evolution
- Accessing avro file generated from Java in Python – Language Independence
- Spark-Avro data access

Few interesting things…

- Avro Cli – Avro Tools jar that can provide some command line help
- Thrift and Protocol Buffers
- Kryo
- Jackson-avro-databind java API
- Project Kiji (Schema management in Hbase)

Please drop mail for support if you have any issues or if you have suggestions on Avro

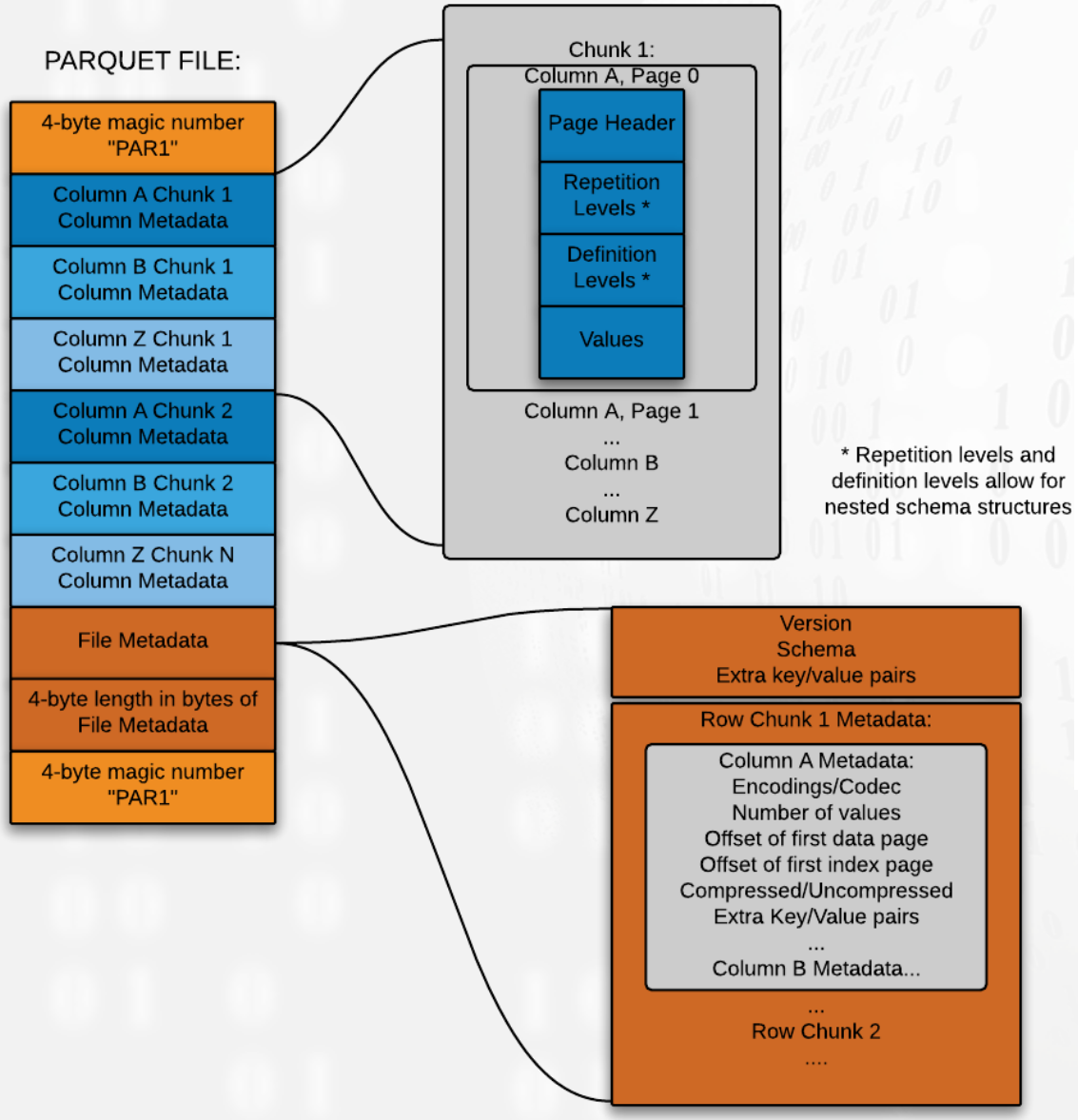
PARQUET - Introduction

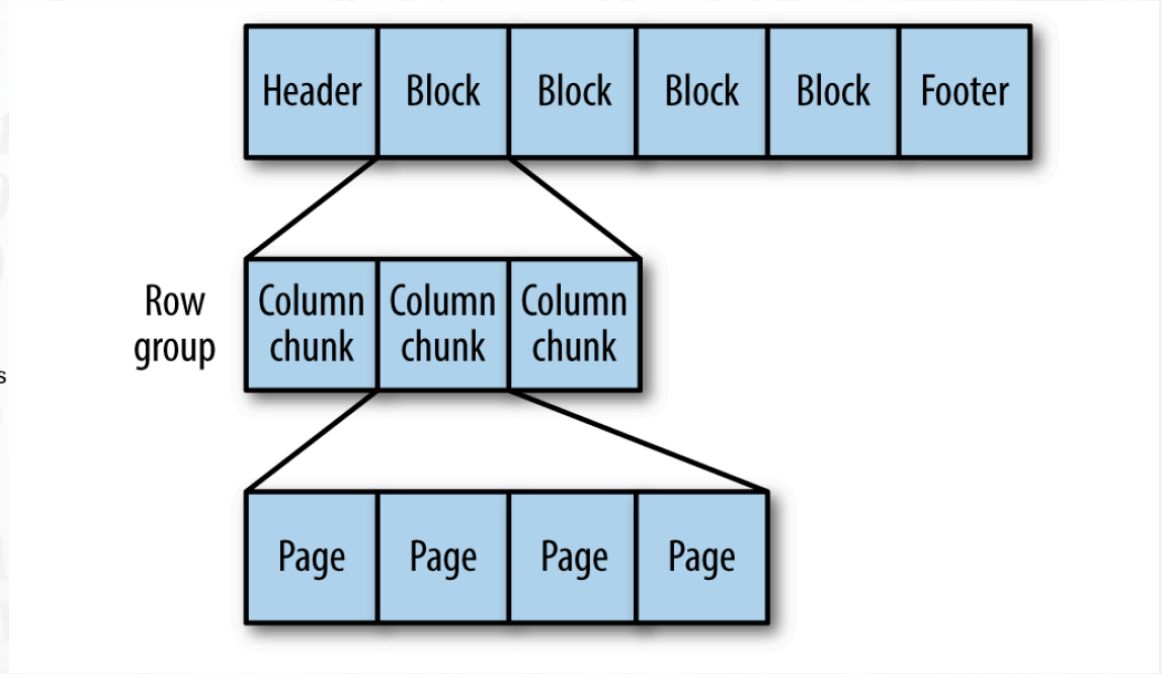
- Columnar storage format that come out of a collaboration between Twitter and Cloudera based on Dremel
- What is a storage format?
- Well-suited to OLAP workloads
- High level of integration with Hadoop and the ecosystem (Hive, Impala and Spark)
- Interoperable Avro, Thrift and Protocol Buffers

PARQUET cont..

- Allows compression. Currently supports Snappy and Gzip.
- Well supported over Hadoop eco system.
- Very well integrated with Spark SQL and DataFrames.
- Predicate pushdown: Projection and predicate pushdowns involve an execution engine pushing the projection and predicates down to the storage format to optimize the operations at the lowest level possible.
- I/O to a minimum by reading from a disk only the data required for the query.
- Schema Evolution to some extent. Allows adding new columns at the end.
- Language independent. Supports Scala, Java, C++, Python.

File Structures - Parquet





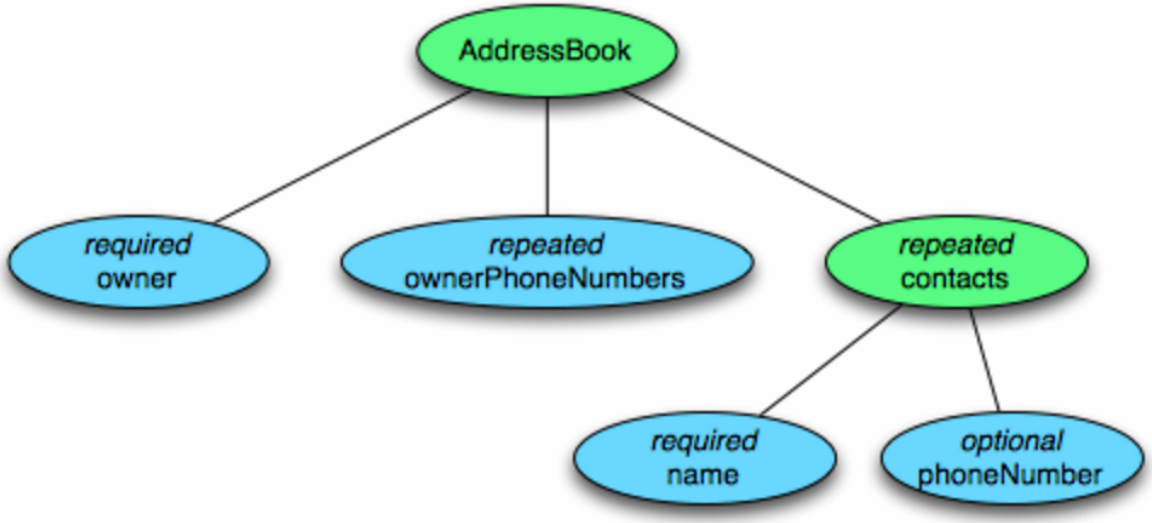
Parquet

Sample Schema

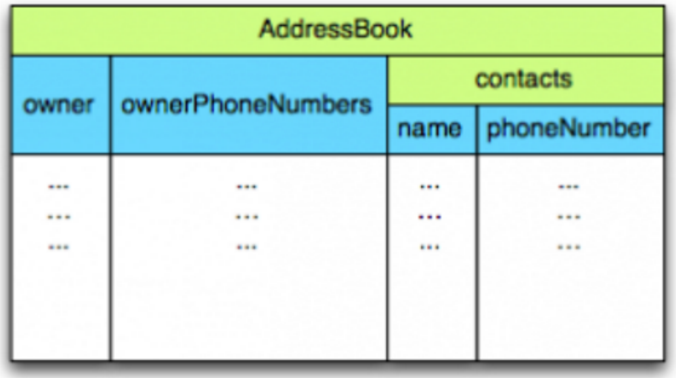
```
message Meetup {
  required binary name (UTF8);
  required binary meetup_date (UTF8);
  required int32 going;
  required binary organizer (UTF8);
  required group topics (LIST) {
    repeated binary array (UTF8);
  }
}
```

- Be careful with Parquet Data types
- Does not have a good stand alone API as Avro, have converters for Avro, PB and Thrift instead
- Flattens all nested data types in order to save them as columnar structures

Nested Schema resolution



Column	Type
owner	string
ownerPhoneNumbers	string
contacts.name	string
contacts.phoneNumber	string



Parquet

Parquet few important notes..

- Parquet requires a lot of memory when writing files because it buffers writes in memory to optimize the encoding and compressing of the data
- Using a heavily nested data structure with Parquet will likely limit some of the optimizations that Parquet makes for pushdowns. If possible, try to flatten your schema

Code examples

- Java API
- Spark Example
- Kafka Exmple

How to decide on storage format

- What kind of data you have?
- What is the processing framework? Future and Current
- Data processing and querying
- Do you have RPC/IPC
- How much schema evolution do you have?

Our experiences with Parquet and Avro

Name space re-definitions

```
<item>
   <chapters>
       <content>
    <name>content1</name>
    <pages>100</pages>
        </content>
    </chapters>
</item>
<otheritem>
   <chapters>
       <othercontent>
    <randomname>xyz</randomname>
    <someothername>abcd</someothername>
       </othercontent>
   </chapters>
</otheritem>
```

Is Agenda Accomplished..??

- ✓Avro and Parquet - When and Why to use which format?
- ✓Use cases for Schema Evolution & practical examples
- ✓Data modeling - Avro and Parquet schema
- ✓Workshop
  - Read Avro input from Kafka
  - Transform data in Spark
  - Write data frame to Parquet
  - Read back from Parquet
- ✓Our experiences with Avro and Parquet
- ✓Some helpful insights for projects design

Questions... ????????

Notes

- https://dzone.com/articles/where-should-i-store-hadoop-data
- https://developer.ibm.com/hadoop/blog/2016/01/14/5-reasons-to-choose-parquet-for-spark-sql/
- http://www.slideshare.net/StampedeCon/choosing-an-hdfs-data-storage-format-avro-vs-parquet-and-more-stampedecon-2015
- http://parquet.apache.org/
- https://github.com/cloudera/parquet-examples
- http://avro.apache.org/docs/current/spec.html#schema_primitive
- http://www.michael-noll.com/blog/2013/03/17/reading-and-writing-avro-files-from-the-command-line/
- https://cwiki.apache.org/confluence/display/AVRO/FAQ
- http://avro.apache.org/
- https://github.com/miguno/avro-cli-examples
- http://avro.apache.org/docs/current/spec.html#schema_primitive
- https://dzone.com/articles/getting-started-apache-avro
- https://github.com/databricks/spark-avro

Notes

- https://github.com/twitter/bijection
- https://github.com/mkuthan/example-spark
- http://blog.cloudera.com/blog/2015/09/making-apache-spark-testing-easy-with-spark-testing-base/
- https://github.com/databricks/spark-avro/blob/master/README.md
- http://www.bigdatatidbits.cc/2015/01/how-to-load-some-avro-data-into-spark.html
- http://engineering.intenthq.com/2015/08/pucket/
- https://dzone.com/articles/understanding-how-parquet
- http://blog.cloudera.com/blog/2015/03/converting-apache-avro-data-to-parquet-format-in-apache-hadoop/

Our Solutions Team

- **Prasad Sripathi, CEO, Experienced Big Data Architect, Head of NJ Data Science and Hadoop Meetups**
- **Sergey Fogelson, PhD, Director, Data Science**
- **Eric Marshall, Senior Systems Architect, Hortonworks Hadoop Certified Administrator**
- **Kristina Rogale Plazonic, Spark Certified Data Engineer**
- **Ravi Kora, Spark Certified Senior Data Scientist**
- **Srinivasarao Daruna**, Spark Certified **Data Engineer**
- **Srujana Kuntumalla**, Spark Certified **Data Engineer**
- **Tim Spann**, **Senior Solutions Architect, ex-Pivotal**
- **Rajiv Singla, Data Engineer**
- **Suresh Kempula, Data Engineer**

Technology Stack