An Overview of Apache Avro

ANVESH NAYAN LINGAMPALLI[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: anveling@umail.iu.edu*

April 12, 2017

Apache Avro is a data serialization system, which uses JSON based schemas and RPC calls to send the data. Hadoop based tools natively support Avro for serialization and deserialization of the data.

Keywords: apache, avro, serialization

https://github.com/cloudmesh/sp17-i524/blob/master/paper2/S17-IR-2016/report.pdf

1. INTRODUCTION

Apache Avro [1] is a data serialization system. Data serialization is a mechanism to translate data in a computer environment, such as memory buffer, data structures, object state into binary or textual form. Java and Hadoop provides serialization APIs, which are java based. Apache Avro is a language independent system, that can be processed by multiple languages [2].

Avro is heavily dependent on schemas. These schemas are defined with JSON that simplifies its implementations with its libraries. Different schemas can be used for serialization and deserialization, and Avro will handle the missing fields or extra fields [3]. When Avro data is stored in a file, its schema is also stored with it. Avro stores the data definition in JSON format, which makes it easier to interpret.

Apache Avro provides rich data structures, compact binary data format, a container file, facility for remote procedure calls (RPC) and integration with dynamic languages [1]. In Remote procedure calls, the client and server exchange schemas during the connection. This exchange helps when there are missing fields or extra fields. Avro works well with Hadoop MapReduce, which provides the large scale processing across many processors to do the calculations simultaneously and efficiently.

2. COMPONENTS OF AVRO

Avro has two main components, data serialization and remote procedure call (RPC) support.[4]

2.1. Data Serialization

Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object. To serialize data using Avro, 1) define an Avro Schema. 2) compile the schema using Avro utility. 3) Java code corresponding to that schema is obtained. 4) populate the schema with data. 5) serialize the data using Avro library[**?**].

2.2. Remote Procedure Call

In a Remote Procedure Call, the client and server exchange schemas in the connection handshake. (This can be optimized so that, for most calls, no schemas are actually transmitted.) Since both client and server both have the other's full schema, correspondence between same named fields, missing fields, extra fields, etc. can all be easily resolved[5].

3. SPECIFICATIONS

3.1. Encoding

Avro specifies two serialization encodings, binary and JSON. Binary encoding is smaller and faster, but not efficient in the case of debugging and web-based applications, where JSON encoding is appropriate[6].

3.2. Object Container Files

Avro includes an object container file format. Objects stored in blocks that may be compressed and synchronization markers are used to permit splitting of the files for MapReduce processing. File consists of, a header followed by one or more data blocks. Header consists of four bytes 'O', 'b', 'j', 1. It also consists of the file metadata, including the schema and a 16-byte, randomly generated sync marker[7]. Currently used file metadata properties are, avro.schema which contains the schema of the objects stored in file avro.codec contains the name of the compression codec used to compress the blocks.

A file data block consists of, count of the objects in the block, size of bytes of the serialized objects in current block, the serialized objects and a 16-byte sync marker[7].

Each block's binary data can be efficiently extracted without deserializing the contents. The combination of the block size, object counts, and sync markers enable detection of corrupt blocks.

4. KEY FEATURES OF APACHE AVRO

Apache Thrift and Google's Protocol buffers are the competent libraries with Apache Avro. But Avro is fundamentally different from these frameworks. The key features of the Apache Avro are, Dynamic typing, untagged data, no manually-assigned field IDs.

4.1. Dynamic Typing

Serialization and deserialization without the code generation is possible in Apache Avro. Data is always accompanied by a schema that permits full processing of the data. This feature of the Avro, makes it possible to construct data-processing systems and languages. Avro creates binary structure format that is both compressible and splittable [3].

4.2. Untagged Data

Binary data with a schema together, allows the data to be written without any overhead. This feature provides faster data processing and compact data encoding.

4.3. Schema Evolution

Avro cleanly handles schema changes such as missing fields, added fields and changed fields. By using this feature, new programs can read old data and old programs can read new data.

5. APPLICATION

Primary use of Apache Avro is in Apache Hadoop where it can provide both serialization format for persistent data, and a wire format for communication between Hadoop nodes.

Avro was designed for Hadoop for making it interoperable across different languages. With Avro serialization, Pig utilizes AvroStorage().

5.1. Using Avro with Eventlet

Eventlet[4] is a concurrent networking library for python that allows the changing of the code, to run the code, instead of writing it. RPC support from the Avro combined with Eventlet is used for building highly concurrent network-based services. Avro is present in the transport layer on top of HTTP for RPC calls. It POSTS binary data to the server and processes the response. Eventlet.wsgi (Web server Gateway Interface) is used to build RPC server.

6. DISADVANTAGES

Avro is not the fastest in serialization process, but it is one of the faster frameworks. The syntax of Avro can be error prone. And error handling is complex when compared with Protocol buffers and Thrift.

7. EDUCATIONAL MATERIAL

Tutorialspoint.com [3]for Apache Avro provides the resources to use Avro for deserialization and serialization of the data. Apache Avro's [1] website provides documentation for understanding, deploying and managing the framework.

8. CONCULSION

Apache Avro is a framework, which allows the serialization of data that has schema built in it. The serialization of data is results in compact binary format, which does require proxy objects. Instead of using generated proxy object libraries and strong typing, Avro heavily relies on the schema that are sent along with serialized data.

REFERENCES

[1] "Apache avro," Web Page, accessed: 2017-03-21. [Online]. Available: https://avro.apache.org/

[2] "Apache Avro documentation," Web Page, accessed: 2017-03-21. [Online]. Available: https://avro.apache.org/docs/1.7.7/index.html

[3] "Avro tutorial," Web Page, accessed: 2017-03-22. [Online]. Available: https://www.tutorialspoint.com/avro/avro_overview.html

[4] "Using Avro with Eventlet," Web Page, accessed: 2017-03-22. [Online]. Available: http://unethicalblogger.com/2010/05/07/how-to-using-avro-with-eventlet.html

[5] "Rpc by avro," Web Page, accessed: 2017-03-22. [Online]. Available: https://www.igvita.com/2010/02/16/data-serialization-rpc-with-avro-ruby/

[6] "Encoding in avro," Web Page, accessed: 2017-03-22. [Online]. Available: https://avro.apache.org/docs/1.8.1/spec.html

[7] "Apache avro wiki," Web Page, accessed: 2017-03-22. [Online]. Available: https://en.wikipedia.org/wiki/Apache_Avro