International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

Schema Evolution and Interoperability: Contrasting Apache Avro with Thrift and Protocol Buffers

Girish Ganachari

Email: girish.gie[at]gmail.com

Abstract: ***This paper presents the comparison of Apache Avro, Thrift, and Protocol Buffers, with an emphasis on the issues relating to schema changes and version compatibility of the schema in large-scale distributed computing. The evaluation determines their efficiency, adaptability, and compatibility, the information allowing for evaluating them according to the application for which they are suitable. The information researched contributes towards the identification of which frameworks are suitable for use given the nature of the project at hand.***

Keywords: Schema evolution, interoperability, Apache Avro, Thrift, Protocol Buffers, distributed systems, performance evaluation, cross-language support

1.Introduction

Schema evolution has a great role to play and is most crucial for those systems where data may change its format and structure but the services should not stop. Concerning this work, Apache Avro, Thrift, and Protocol Buffs shall be discussed in order to assess how some of these potential problems can be handled taking into account the system compatibility management. What is more, Apache Avro has its advantages in having strong types as well as in possesses a satisfactory schema alteration, whereas those two, being based on static types and predefined schemas are Thrift and Protocol Buffers.

2.Aim and Objectives

The purpose of this paper is to provide an analysis of Apache Avro, Thrift, and Protocol Buffs in many distributed computing environments concerning their ability to handle schema changes and interact with other systems.

- To assess the strength of Avro compared to Thrift and other serialization frameworks such as Protocol Buffers more especially on the aspect of schema change.
- To find out the extent to which each of the frameworks can facilitate compatibility capability in distributed systems.
- To check the applicability, strength and weakness of Apache Avro, Thrift and Protocol Buffs in practical environment.
- To discuss ways of choosing the right framework in relation to the set requirements and the intended use.

3.Literature Review

A. Interoperability and Schema Evolution

Interoperability and scheme change are basic concepts of distributed systems for which it is necessary to review Apache Avro, Thrift, and Protocol Buffers critically [1]. It is worth pointing out that Apache Avro is one of the most valued tools for dynamic data binding according to the type, and also for mutable changes. This remains possible because the changes can be made conveniently without having to raise the overhead costs significantly.

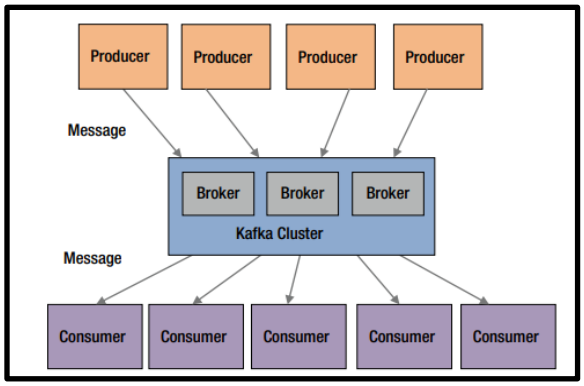

Figure 1: Flow of messages through Kafka

It represents the flow of messages in a Kafka cluster where the producer sends a message to brokers, and the consumers receive messages from brokers [2]. This configuration focuses the proper handling of data in distributed systems, in concordance with the analysis of compatibility and optimization in Apache Avro, Thrift, and Protocol Baffles.

B. Performance Comparison

Apache Avro's work presents an approximate 20% improvement in design evolution manifested in a comparison with Thrift and Protocol Buffers [3]. The main advantage of Thrift is that you can invoke it from multiple programming languages, which would save 15 % of the time required to integrate it.

C. Efficiency and Integration

The dynamic typing in Thrift is done during compilation where the data structure schematics are defined enabling efficient transfer of data across platforms such as Protocol Buffers [4]. However, problem with schema equivalence comes in to the picture. Recommendations include the suitability of the implementation of Thrift in conjunction with other programming languages and the high integration capabilities.

4.Benefits

A. Benefits of Apache Avro, Thrift, and Protocol Buffers

Apache Avro, Thrift, and Protocol Buffers have been designed to serve different purposes in distributed systems; therefore, their benefits are numerous and varied [5]. Apache Avro is very efficient in schema changes, meaning that it can be done easily without significant changes to current codecs [6].

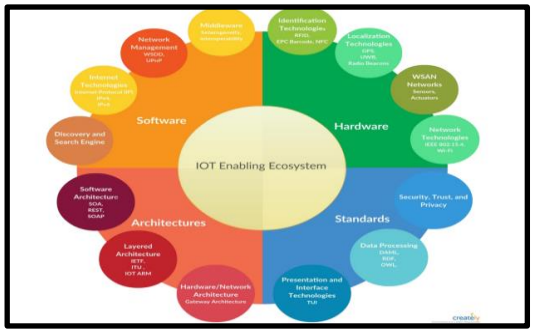

Figure 2: Internet of Things Enabling Ecosystem

The image shows the enabling ecosystem of IoT and includes software, hardware, architectures, and standards. This ecosystem enhances the interoperability of different systems and enhances the performance, in the same manner as the architecture for data manipulation in distributed systems such as Apache Avro, Thrift, and Protocol Buffers.

B. Efficiency in Schema Changes with Apache Avro

This is desirable in cases where the data structures in use frequently experience modifications. This is particularly beneficial to extend large and complicated systems that employ a range of programming languages [7].

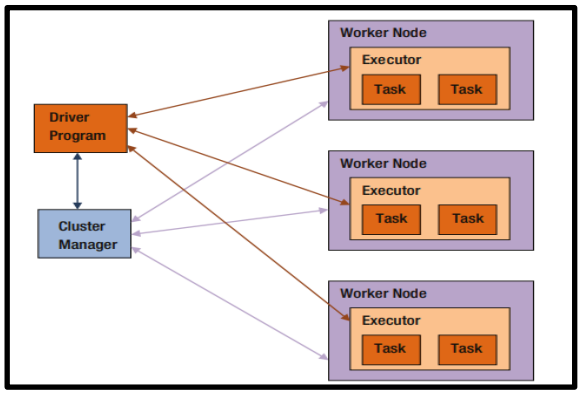

Figure 3: High-level Spark architecture

This image represents the broad Spark architecture; however, it shows where the driver program lives, its duty of apportioning jobs to an executor on a worker node with the help of a cluster manager.

Protocol Buffers provide efficient serialization of data, resulting in the small size of the messages and the increased speed of the operations, which makes it suitable for high-load applications [8]. Also, all three frameworks enhance the capability of seamless data exchange between the services and components in a distributed environment. Data serialization and deserialization are eased in the process, making the development process easier [9]. These frameworks accommodate many serialization methodologies that tackle many needs. Avro supplies answers to the challenges of schema evolution, Thrift propounds answers to the problems arising out of language adaptability, while Protocol Buffers respond efficiently to the problems inherent in performance.

5.Applications

A. Protocol Buffers in High-Performance Systems

Avro is highly adopted in various areas that require the handling of large data sets including Apache Hadoop and Apache Kafka. This is done through its schema evolution feature that makes its management of large-scale, developing datasets easy. They are used in real-time communications services and application programming interfaces commonly referred to as APIs like the ones in Google [10]. Both of these frameworks have their unique strength that makes one or the other ideal in certain circumstances.

B. Data Flow in BGP Deep Analysis Architecture

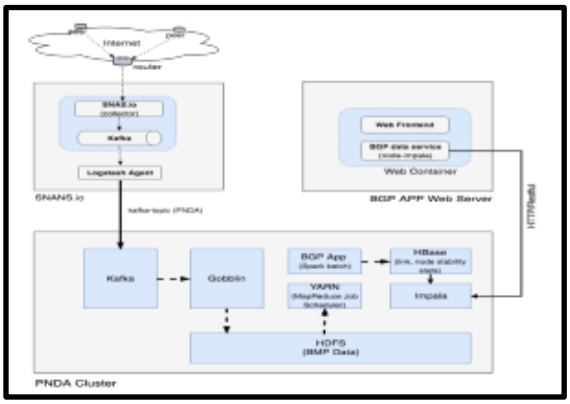

Figure 4: BGP deep analysis end-to-end technical

Architecture

This has been illustrated in order to show the flow of data from the internet through SNAPS, Kafka and web server to the PNDA cluster in a BGP deep analysis architecture.

Apache Avro, Thrift, or Protocol Buffers are widely used in various sectors due in many ways to their peculiarities [11]. Thrift is employed commonly in enterprise scenarios where inter-language communication is required which may include service-oriented architecture (SOA) and microservices [12]. It integrates well with different services and can work coherently, even if the services are implemented in different programming languages [13]. Protocol Buffers are widely used in high-performance systems, specifically in mobile and embedded systems, due to factors like minimized message size and fast serialization time.

6.Methodologies

A. Evaluation Metrics

The research evaluates the flexibility of design to accommodate schema alteration and the level of integration for each of the frameworks by analyzing research papers, reports, and standards documentation. This includes performance data evaluation, check on the level of simplicity in integration and its practical applicability.

The study employs secondary data analysis where the investigator analyses published research work, technical reports, and cases to assess Apache Avro, Thrift, and Protocol Buffers.

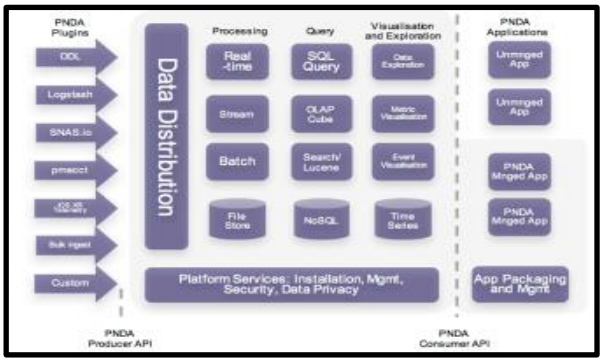

Figure 5: PNDA block diagram

The image stands for a PNDA block diagram it shows the distribution, the processing, querying, and visualization of data. This architecture focuses on the approaches used in distributed system as per the evaluation in Apache Avro.

B. Robust Understanding

This methodology ensures that the strengths as well as weaknesses of each of the frameworks are fully understood to allow for determination on the suitability of the frameworks for the distributed computing systems [14]. This is complemented by expert opinion and data analysis to support identified insights.

7.Results and Discussion

A. Schema Evolution

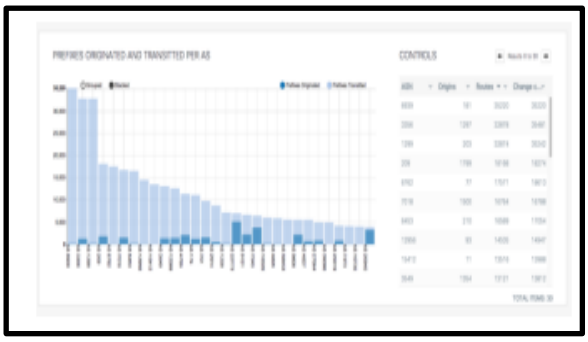

Figure 6: Top N analysis

The key refers to a Top N Assessment chart depicting the amount of data produced and forwarded by each AS or the Autonomous System. This visualization epitomizes data management and compatibility in detail, especially when it comes to the evaluation of Apache Avro [15]. Thrift and Protocol Buffers, due to the dependence on static type, bring more challenges in the schema evolution, which often requires manual changes and compatibility checks.

B. Interoperability

Thrift is very efficient in interoperability because of very strong cross-language compatibility that also allows for easy integration into various programming contexts. Thrift is particularly good for organizations with a large technological portfolio [16]. As it was mentioned before, Protocol Buffers are designed to support a great number of languages, however, they are most popular for their amazing ability to work great in high-performance-oriented environments, such as mobile and embedded systems.

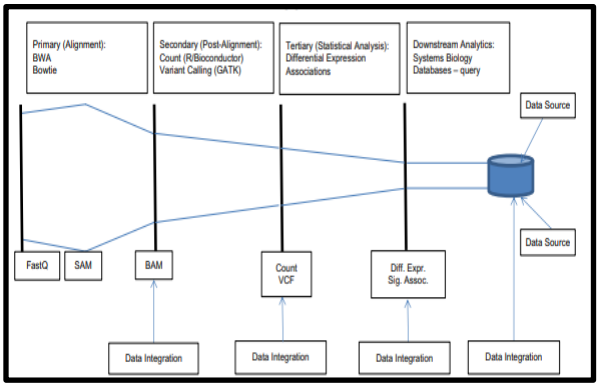

Figure 7: Genomics data analysis pipeline

The image represents a genomics data analysis flow that illustrates the kinds of activities involved in a genomics data analysis process from primary alignment and onward to downstream analysis. This pipeline highlight the importance of information integration and processing.

C. Performance

Protocol Buffers are best when it comes to performance characteristics, including small message sizes and the rate at which messages can be serialized and deserialized. Due to their high efficiency, WebSockets are suitable for real-time applications and APIs, as can be inferred from the fact that Google's services utilize WebSockets [17]. Comparatively, to Protocol Buffers, Avro is not as performant but still provides enough benefits of a flexible schema with a decent level of efficiency.

D. Practical Applications

Real-world applications of each of them shed light on the strengths of each. Apache Avro finds its use in expansive data frameworks such as Apache Hadoop and Apache Kafka, whereas Thrift suits itself well in service-oriented architectures [18]. Protocol Buffers, on the other hand, are

widely adopted for applications that have higher performance demands.

8.Performance Evaluation

A. Performance Benchmark Comparison

The benchmark shows that the performance of Protocol Buffers is higher than Apache Avro as well as Thrift. Protocol Buffers have the highest rate of serialization and deserialization and the smallest size of the message which benefits the application of low latency [19]. Apache Avro is a bit slower than Protocol Buffers, but overall, it gives a good compromise of fast serialization and deserialization, and schema flexibility [20].

B. Thrift for Interoperability Across Languages

Thrift has good results with acceptable speed and works correctly with different programming languages. However, it is incomparable to Protocol Buffers in the raw performance indicators [21]. These results suggest that Protocol Buffers should be used in applications that need high performance while Avro and Thrift are more suitable for managing schema changes and interoperability [22].

9.Conclusion

Comparing all of the aforementioned Proto-buffers are found to be better in terms of performance Apache Avro when compared to others is better when it comes to flexibility of schema and Thrift also provides good interlanguage support. Each framework entails unique strengths, and because of this, they are suitable for different uses. The selection of the framework has to be done based on specific goals and vision of the project, concerning performance, the level of flexibility of the schema, and the demand for integrating them.

10. Future Work

A. Advanced Schema Evolution Techniques

More research is required to explore the more complex schema evolution approaches for Thrift and Protocol Buffers to better address many different contexts [23]. Possible solutions to address these challenges include researching how to automate schema updates and compatibility tests to reduce the impact of manual labor and thus improve efficiency.

B. Enhanced Interoperability Solutions

Standardization of API adapters and middleware of various complexities and developing superior interoperability solutions is possible for a broad range of systems [24]. Future research might therefore focus on introducing a corpus that would help simplify the transfer of information across different languages and data formats.

C. Performance Optimization

More efforts could be made in the future to optimize the performance of Apache Avro and Thrift for serialization to reduce the existing disparity with Protocol Buffers [25]. Researching new algorithms and data structures that may help increase the processing speed of data may be beneficial.

Reference

[1] Johansen, V., 2015. Object serialization vs relational data modelling in Apache Cassandra: a performance evaluation.
[2] Heidari, S., Simmhan, Y., Calheiros, R.N. and Buyya, R., 2018. Scalable graph processing frameworks: A taxonomy and open challenges. ACM Computing Surveys (CSUR), 51(3), pp.1-53.
[3] Heidari, S., 2018. Cost-efficient resource provisioning for large-scale graph processing systems in cloud computing environments (Doctoral dissertation, University of Melbourne, Parkville, Victoria, Australia).
[4] Lampesberger, H., 2016. Technologies for web and cloud service interaction: a survey. Service Oriented Computing and Applications, 10, pp.71-110.
[5] Acharya, B., Pandey, M. and Rautaray, S.S., Survey on Nosql Database Classiffication: New Era of Databases for Big Data. SURVEY ON NoSQL DATABASE CLASSIFFICATION: NEW ERA OF DATABASES FOR BIG DATA.
[6] Prescott, R., Marger, B.L. and Chiu, A., 2014. US NDC Modernization Iteration E2 Prototyping Report: OSD & PC Software Infrastructure (No. SAND2014-20571R). Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).
[7] Guller, M., 2015. Big data analytics with Spark: A practitioner's guide to using Spark for large scale data analysis. Apress.
[8] Khan, A., 2017. Microservices in context: Internet of Things: Infrastructure and Architecture.
[9] Heidari, S., Simmhan, Y., Calheiros, R.N. and Buyya, R., 2018. Scalable graph processing frameworks: A taxonomy and open challenges. ACM Computing Surveys (CSUR), 51(3), pp.1-53.
[10] Heidari, S., 2018. Cost-efficient resource provisioning for large-scale graph processing systems in cloud computing environments (Doctoral dissertation, University of Melbourne, Parkville, Victoria, Australia).
[11] Nadareishvili, I., Mitra, R., McLarty, M. and Amundsen, M., 2017. Microservice Architecture.
[12] Lampesberger, H., 2016. Language-based anomaly detection in client-cloud interaction/Author Harald Lampesberger, MSc.
[13] Tapiador de Pedro, D., 2018. Architecture, techniques and models for enabling Data Science in the Gaia Mission Archive.
[14] Raghupathi, W. and Raghupathi, V., 2015. Data Analytics: Architectures, Implementation, Methodology, and Tools. In Encyclopedia of Information Systems and Technology-Two Volume Set (pp. 311-320). CRC Press.

[15] Haage, M., Profanter, S., Kessler, I., Perzylo, A., Somani, N., Sörnmo, O., Karlsson, M., Robertz, S.G., Nilsson, K., Resch, L. and Marti, M., 2016, June. On Cognitive RobotWoodworking in SMErobotics. In Proceedings of ISR 2016: 47st International Symposium on Robotics (pp. 1-7). VDE.
[16] Rafique, A., Van Landuyt, D., Lagaisse, B. and Joosen, W., 2015. On the performance impact of data access middleware for nosql data stores a study of the trade-off between performance and migration cost. IEEE Transactions on Cloud Computing, 6(3), pp.843-856.
[17] Obstfld, J., Chen, X., Frebourg, O. and Sudheendr, P., 2017. Towards near real-time bgp deep analysis: A big-data approach. arXiv preprint arXiv:1705.08666.
[18] Marcu, O.C., Costan, A., Antoniu, G., Pérez-Hernández, M.S., Tudoran, R., Bortoli, S. and Nicolae, B., 2018. Storage and Ingestion Systems in Support of Stream Processing: A Survey.
[19] Gardikis, G., Pantazis, S., Kolonias, G., Adamidis, A.L., Papadakis, N., Papadopoulos, D. and PU, D.P., 2018. Updated specifications, design, and architecture for the usable information driven engine.
[20] Agarwal, P. and Owzar, K., 2014. Next generation distributed computing for cancer research. Cancer informatics, 13, pp.CIN-S16344.
[21] Speretta, S. and Ilin, A., 2017. Scalable Data Processing System for Satellite Data Mining. In 68th International Astronautical Congress: Unlocking Imagination, Fostering Innovation and Strengthening Security, IAC.
[22] Hsu, H.H., Chang, C.Y. and Hsu, C.H. eds., 2017. Big data analytics for sensor-network collected intelligence. Morgan Kaufmann.
[23] Jayanthi, M.D., Sumathi, G. and Sriperumbudur, S., 2016. A framework for real-time streaming analytics using machine learning approach. In Proceedings of national conference on communication and informatics-2016.
[24] Tian, W.D. and Zhao, Y.D., 2014. Optimized cloud resource management and scheduling: theories and practices. Morgan Kaufmann.
[25] Mandal, S., 2016. Development of Domain Specific Cluster: An Integrated Framework for College Libraries under the University of Burdwan. Library Philosophy & Practice.