State of the art in Distributed Computing Technology for Real-Time Analytics

Carl M Wilson

University of Huddersfield, u0370630@unimail.hud.ac.uk

1 INTRODUCTION

Big Data Analytics [BDA] can be defined as “high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.” (Gartner, 2021). This definition is widely considered to be *the* definition (Loshin, 2013), and while there are expansions on these “three V’s”, all later expansions always encompass “volume, velocity and variety” (Buyya et al., 2016).

The Large Hadron Collider [LHC] at CERN [Conseil Européen pour la Recherche Nucléaire or European Council for Nuclear Research] generates around 30 petabytes of information per year, and provides near real-time access to the LHC to a large community of physicists around the globe (CERN, 2021). This is a significant BDA challenge, both real-time and post-processing.

This paper will initially focus on the state of the art in distributed processing technologies employed to manage BDA problems, including the evolution from batch to stream processing as well as examining real-time analytics use case of condition monitoring at CERN.

2 TECHNOLOGY REVIEW

Apache Hadoop is an open-source batch processing distributed framework with one of its key components being MapReduce (Buyya et al., 2016). MapReduce is a distributed data processing model, and a lot of distribution engines are often analogous to the MapReduce paradigm – the idea of partitioning a dataset and processing on a distributed network.

An important aspect of real-time analytics are the three stream processing paradigms: *at-least-once*, *at-most-once* and *exactly-once*. *At-least-once* and *exactly-once* both save results prior to saving offset with the key difference between the *exactly-once* processing drops duplicate events making it more suitable for stateful applications where accuracy is key. *At-most-once* differs from the two previous paradigms in that is saves the offset prior to processing, making it slightly better for aggregation than *at-least-once* but at the risk of data loss in the event of failure. Both *at-least-once* and *at-most-once* are highly suited to applications, such as tickers, where accuracy of aggregation is not required (Alla, 2018).

2.1 Spark Streaming

Apache Spark is an open-source distributed computing framework also offering MapReduce type engine for parallel batch processing of data. It is naturally turned to as the successor for Hadoop, with its own data model known as Resilient Distributed Dataset [RDD]. RDD's are tabulations of data that can be transformed or filtered down to finer degrees of granularity and ultimately distributed for computation. Each RDD contains information on its lineage such that it can be traced back. Each RDD can then be stored on disk or held in memory for processing (Zaharia et al., 2012). This in memory processing provides a significant speed up over Hadoop (Buyya et al., 2016) with numbers quoted as high as two orders of magnitude faster computation that ordinary MapReduce (Ganelin, 2016).

Spark also differs from Hadoop in that it can parse data and pass it to the next step in a data pipeline without having to write to disk. Thus, between this and RDD Spark not only offers a significant speed-up over the Hadoop but makes it much more suitable than for stream-processing. Although it is claimed that Spark can process up to 100x faster than MapReduce, a study completed by (Ahmed et al., 2020) observed that Spark was two to fourteen times faster than Hadoop, depending on the type of application. It was noted that this was heavily depending on data size, parameter selection and tuning.

Spark Streaming is an extension of the Spark API, specifically designed to manage streaming datasets, but still uses the Spark Engine for processing. Spark Streaming uses the *exactly-once* processing and consumes and processes data in micro-batches of DStreams, these are discretised streams of data which represent a sequence of RDD's, see Figure 1.

Figure 1: internals of Spark Streaming and flow of data batches from Streaming to the Spark Engine. Note all data is in discrete batches into and out of Spark (Apache, 2021b)

Thus, it follows that the processing engine is limited to bound data – that is that each piece of distributed data for processing requires a fixed length. These micro-batches are a minimum of 500 ms (Alla, 2018). This additional process is one of the key constraints on why Spark Streaming cannot be as fast as Apache Flink for real-time analytics.

2.2 Flink

Apache Flink is an open-source stream processing framework that provides accurate results, is fault tolerant and can run thousands of nodes with high throughput and low latency (Alla, 2018). What initially sets Flink apart from Spark is its capability to managed unbound data, of no fixed length, see Figure 2.

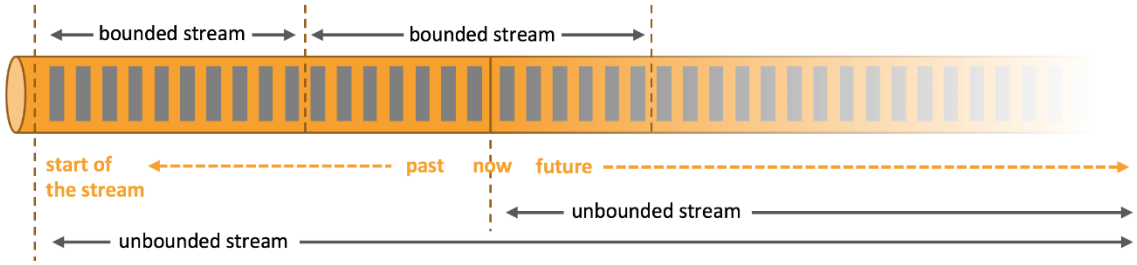

Figure 2: Unbounded and bounded data-streams in Apache Flink (Apache, 2021a).

Another strength of Flink is flexible windowing. Windows can be highly customised on sizing and triggers to support very complex streaming patterns. This allows Flink to capture the reality of the environment that data was created in – making it ideal for Internet-of-Things [IoT] applications as well as condition monitoring.

Flink can handle out-of-order data and has state management. Flink works on an *exactly-once* paradigm for stateful computations making it the most ideal candidate for real-time stateful analytics, where state management can be stored on disk, for a reduced overhead on in-system memory, or in-memory for higher velocity and more demanding applications, see Figure 3.

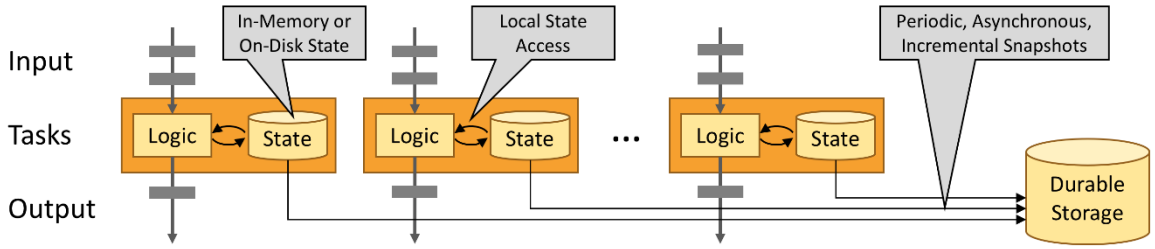

Figure 3: Stateful Flink Applications optimised for local state access (Apache, 2021a).

A benchmarking exercise comparing latency, throughput, singe and period burst workloads for Spark Streaming and Flink observed that Flink has the lowest latencies in four different situations (Van Dongen & Van den Poel, 2020).

2.3 Amazon Kinesis

Amazon Web Services [AWS] offer Kinesis as a fully managed cloud solution for collecting, processing, and analysing real-time data. Kinesis is a proprietary software, whereas the Apache offerings are open-source. Kinesis is capable of continuously capturing and storing terabytes of data per hour (Buyya et al., 2016).

Kinesis does differ slightly from Flink in that it can operate as the front-end for managing and storing streams of data for processing by a different application, such as Flink or it can manage and process entirely within Kinesis, see Figure 4.

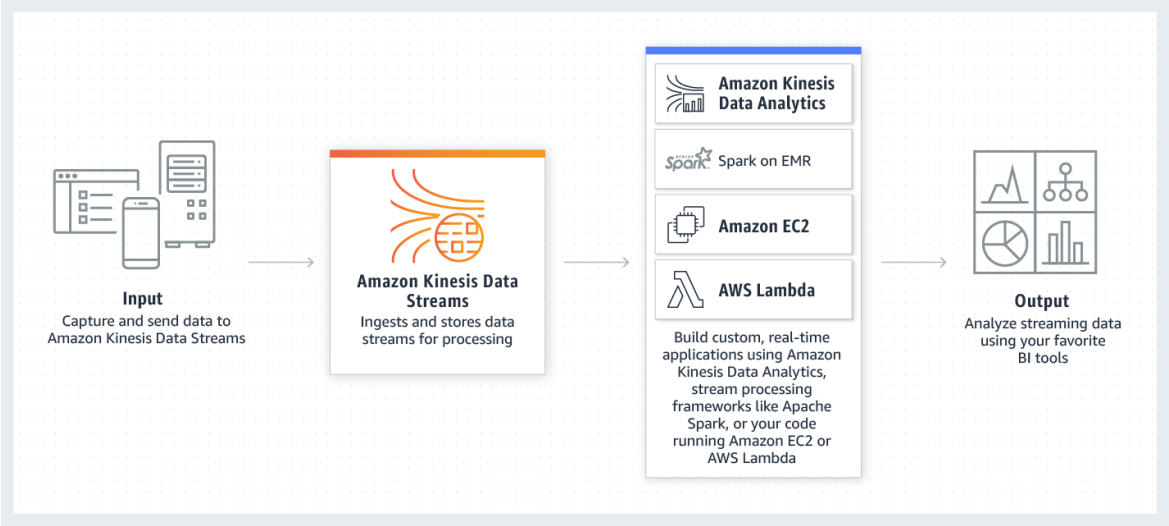

Figure 4: Integration of AWS Kinesis Data Streams with processing platforms (AWS, 2021)

Users create Kinesis Data Stream applications, which would take input data from a Kinesis data stream as a data record (Alla, 2018). The delay between data arriving in the stream and being usable by an application is less than 1 second, making it ideal for rea-time analytics (AWS, 2017). Kinesis Data Analytics also offers *exactly-once* processing.

Where Kinesis excels is its offer real-time scalability which can flex to the customers requirement. AWS offers a different approach to distributed computing, not requiring the user to put their own infrastructure in place but rather rent a fully managed service from AWS.

3 APPLICATION

The bespoke high performance computer [HPC] system at CERN, can capture data at a velocity of 300 gigabytes per second. (Marr, 2016). It is crucial that reliable data is captured during these experiments, and real-time monitoring of sensor performance is an excellent example of a real-time analytics solution.

In the case of (Pol et al., 2019) Artificial Neural Networks [ANN] were investigated to monitor the Compact Muon Solenoids [CMS] at the LHC.

The study monitored 250 chambers for 84 runs resulting in 21000 occupancy matrices which were 3D matrices of channel, layer and count data for sensors chambers. The study examined three approaches: local, considering only data in one layer, regional, considering only data in one chamber and global, considering all available data across all chambers.

Methods examined ranged from unsupervised, examining variance within a layer, to semi-supervised Support Vector Machine [SVM] and supervised learning techniques with fully connected ANN's and Convolutional Neural Networks [CNN]. CNNs were particularly interesting as the 3D nature of the occupancy matrices could be considered as semi-structured image data which CNNs excel at.

The results indicated that the CNN outperformed all other techniques and as a classifier could identify detector malfunctions that the existing production solution could, additionally more information could be extracted on potential malfunctions and data quality issues in an automated fashion. The solution was ultimately

run in parallel to the existing production solution for a period, before switching over after it was fine tuned for new production data.

4 CONCLUSIONS

It has been shown how the streaming platforms have evolved from MapReduce batch-processing in Hadoop, to higher speed micro-batch processing with streaming libraries, in Spark Streaming to high-speed streaming specific frameworks such as Flink. These evolutions have been driven by the increasing need for real-time processing of significant volumes of data at high velocity. It has also been shown that cloud solutions such as AWS Kinesis offer an alternate infrastructure for real-time analytics. While benchmarking studies did not confirm claimed processing power differences between different technologies, it was noted that setup and optimisation of each individual technology was crucial for performance.

It has been shown how machine learning can be used with BDA to live monitor the quality of data being collected at the LHC, using a CNN, which is now implemented in production.

Future applications will drive the need for lower latencies across larger distributed networks to ingest, process and make live decisions on significantly higher volumes of data, as velocity is defined as the rate of change – the greater the volume from higher fidelity data ingestion then the higher the velocity and the greater the distributed processing power required to handle it. A roadmap paper published examining the needs of the LHC for the next 10 years estimated that by 2025 an order of magnitude increase in CPU seconds will be required in comparison to 2020, with this again quadrupling at a peak in 2027. The solution to this problem will lie in development of innovative distribution models and software – as outlined in the paper (Amadio et al., 2019).

5 REFERENCES

Ahmed, N., Barczak Andre, L. C., Teo, S., & A, R. M. (2020). A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench. *Journal of big data, 7*(1), 1-18. https://doi.org/10.1186/s40537-020-00388-5

Alla, S. (2018). *Big Data Analytics with Hadoop 3* (1st ed.). Packt Publishing. https://doi.org/https://go.exlibris.link/S905vz5t

Amadio, G., Apostolakis, J., Banerjee, S., Bernius, C., Bhimji, W., Bloom, K., Britton, D., Calafiura, P., Canal, P., Carlino, G., Cattaneo, M., Chapman, J., Chen, G., Clemencic, M., Cogneras, E., Colling, D., Cosmo, G., Couturier, B., Cristella, L., Crooks, D., Crépé-Renaudin, S., Currie, R., De, K., De Roeck, A., Delgado Peris, A., Di Girolamo, A., Dimitrov, G., Dziedziniewicz-Wojcik, K., Egede, U., Forti, A., Frost, J., Gardner, R., Garonne, V., Genser, K., Geurts, F., Gheata, M., Giagu, S., Giffels, M., Girone, M., González Caballero, I., Grandi, C., Guan, W., Gyurjyan, V., Heiss, A., Hildreth, M., Hristov, P., Ivanchenko, V. N., Iven, J., Keeble, O., Klimentov, A., Komarov, I., Koppenburg, P., Kowalkowski, J., Kreczko, L., Lassnig, M., Leggett, C., Linacre, J., Linden, T., Livny, M., Love, P., Magini, N., Marshall, Z. L., Martelli, E., Mazumdar, K., Meinhard, H., Moneta, L., Nairz, A., Norman, A., Pansanel, J., Pascuzzi, V. R., Petric, M., Petzold, A., Piilonen, L., Piparo, D., Potamianos, K., Puig Navarro, A., Reuter, J., Rodrigues, E., Roy, G., Santana, R., Schovancová, J., Schulz, M., Sciabà, A., Sekmen, S., Seymour, M., Short, H., Sokoloff, M. D., Stewart, G., Sánchez, A., Templon, J., Tunnell, C., Van Gemmeren, P., Vuosalo, C., Vázquez Sierra, C., Wartel, R., Watts, G. T., Wenaus, T., Wenzel, S.,

Wissing, C., Wynne, B., & The, H. E. P. S. F. (2019). A Roadmap for HEP Software and Computing R&D for the 2020s. *Computing and software for big science, 3*(1), 1-49. https://doi.org/10.1007/s41781-018-0018-8

Apache. (2021a). *Flink In Memory Performance*. Retrieved 12/10/2021 from https://flink.apache.org/flink-architecture.html#leverage-in-memory-performance

Apache. (2021b). *Spark Streaming Internals*. https://spark.apache.org/docs/latest/streaming-programming-guide.html

AWS. (2017). *Amazon Kinesis Data Streams Developer Guide* https://docs.aws.amazon.com/streams/latest/dev/kinesis-dg.pdf#amazon-kinesis-streams

AWS. (2021). *Amazon Kinesis Data Streams*. Retrieved 15/10/2021 from https://aws.amazon.com/kinesis/data-streams/

Buyya, R., Calheiros, R. N., & Dastjerdi, A. V. (2016). *Big data: principles and paradigms* (1st ed.). Morgan Kaufmann. https://doi.org/10.1016/C2015-0-04136-3

CERN. (2021). *Computing*. Retrieved 10/10/2021 from https://home.cern/science/computing

Ganelin, I. (2016). *Spark: big data cluster computing in production* (1st ed.). Wiley. https://go.exlibris.link/QHvXVW7X

Gartner. (2021). *Gartner Glossary*. Gartner. Retrieved 10/10/2021 from https://www.gartner.com/en/information-technology/glossary/big-data

Loshin, D. (2013). *Big data analytics: from strategic planning to enterprise integration with tools, techniques, NoSQL, and graph* (1st ed.). Academic Press. https://ebookcentral.proquest.com/lib/hud/detail.action?docID=1361899

Marr, B. (2016). *Big data in practice: how 45 successful companies used big data analytics to deliver extraordinary results*. Wiley. https://go.exlibris.link/qhM2Y8kk

Pol, A. A., Cerminara, G., Germain, C., Pierini, M., & Seth, A. (2019). Detector Monitoring with Artificial Neural Networks at the CMS Experiment at the CERN Large Hadron Collider. *Computing and software for big science, 3*(1), 1-13. https://doi.org/10.1007/s41781-018-0020-1

Van Dongen, G., & Van den Poel, D. E. (2020). Evaluation of Stream Processing Frameworks. *IEEE transactions on parallel and distributed systems, 31*(8), 1-1. https://doi.org/10.1109/TPDS.2020.2978480

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., & Stoica, I. (2012). *Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing*

Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, San Jose, CA.