Name: Deep Pawar (A20545137)
Professor: Joseph Rosen
Institute: Illinois Institute of Technology

CSP 554: Big Data Technologies

Fall 2024 - Research Paper Draft

COMPARATIVE ANALYSIS OF BIG DATA TECHNOLOGIES FOR DATA ACQUISITION STREAM-PROCESSING TOOLS AND ANALYSIS

Abstract: In today's digital landscape, data has become a critical asset, driving innovation and insights across diverse domains. However, organizations face significant challenges in managing the exponential growth of data in terms of volume, variety, and velocity. This paper comprehensively analyzes cutting-edge technologies across the data value chain, focusing on their roles and effectiveness in data acquisition and analysis. Technologies such as Spark, Storm, Kafka, Flink, and Samza will be examined for high-speed data ingestion. This research highlights key criteria for evaluating these technologies, including real-time analytics, distributed storage capabilities, and scalability. Additionally, case studies from domains such as healthcare, finance, and IoT will illustrate how these tools address unique challenges and bridge the gap between academic advancements and industry implementations. By offering a structured comparison and practical insights, this study aims to provide researchers, data engineers, and practitioners with a valuable resource to navigate the complexities of the big data value chain and select appropriate tools for specific data management and analysis needs.

Keywords: Big Data, Data Acquisition, Spark, Storm, Kafka, Flink, Samza, volume, variety, velocity, healthcare, finance, IoT.

INTRODUCTION

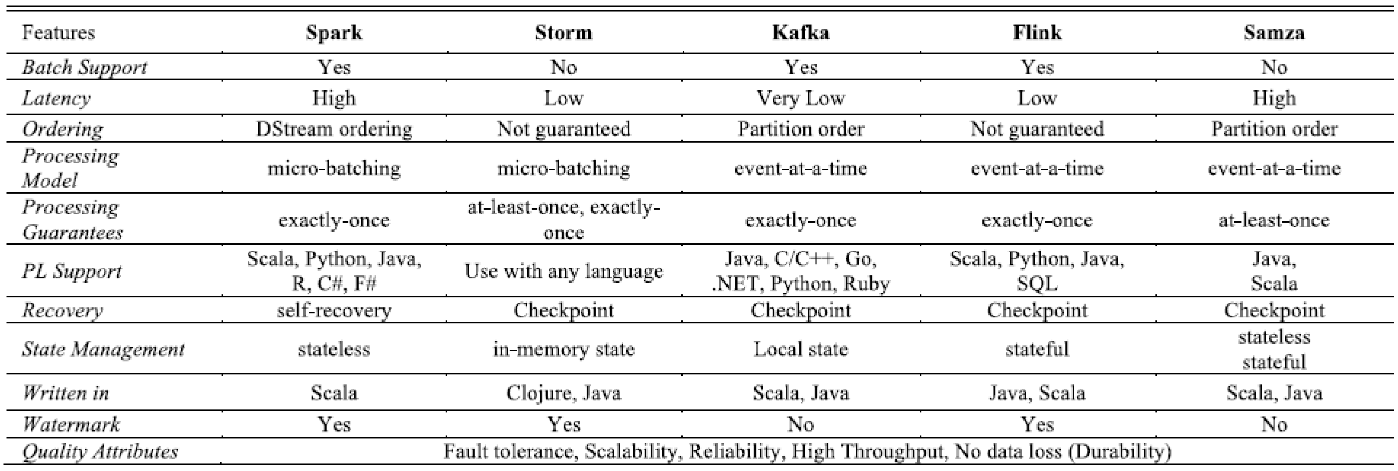
The era of big data has revolutionized the way organizations operate, enabling data-driven decision-making and fostering innovation across industries. The ability to process and analyze massive volumes of data in real-time is no longer a competitive advantage but a necessity for businesses striving to remain relevant in an increasingly digitalized world. However, the exponential growth of data presents unique challenges related to acquisition, storage, and analysis. To harness the full potential of big data, it is imperative to adopt efficient, scalable, and robust technologies tailored to specific stages of the data value chain. Data acquisition involves capturing data from diverse sources, often in real-time, and managing its variety, velocity, and volume. On the other hand, data analysis focuses on extracting patterns, trends, and insights, which are crucial for strategic planning and operational efficiency.

A plethora of technologies have emerged to address the challenges of data acquisition and analysis, each offering unique features and capabilities. Distributed stream-processing frameworks such as Apache Spark, Apache Storm, Apache Flink, Apache Kafka, and Apache Samza have gained prominence for their ability to handle high-speed data ingestion, perform complex computations, and provide real-time analytics. These tools have been adopted in various sectors, from monitoring patient vitals in healthcare and detecting fraudulent transactions in finance to optimizing sensor data in IoT environments.

This paper seeks to provide a comprehensive comparative analysis of these technologies, exploring their strengths, limitations, and suitability for different applications. By examining key evaluation criteria such as scalability, fault tolerance, and performance, this study will shed light on the factors that organizations must consider when selecting technologies for their big data initiatives. Furthermore, case studies from real-world applications will illustrate how these tools bridge the gap between theoretical advancements and practical implementations.

I. BIG DATA ACQUISITION TECHNOLOGIES

The primary focus of the acquisition stage is gathering, preprocessing, and transmitting data generated from diverse sources. As a result, acquisition technologies are often regarded as stream-processing tools. This section highlights open-source acquisition technologies commonly used in both industry and academic research. These technologies handle small, continuously incoming data streams at unpredictable rates, primarily relying on the limited memory of the underlying system. They leverage parallel computations and perform real-time operations to deliver analytics results within seconds or milliseconds. Such rapid processing is essential in applications like finance, banking, network and traffic monitoring, fraud detection, and emergency management, where timely actions are critical. A comparative analysis of five prominent acquisition technologies is presented in Table 1.



Features	**Spark**	**Storm**	**Kafka**	**Flink**	**Samza**
Batch Support	Yes	No	Yes	Yes	No
Latency	High	Low	Very Low	Low	High
Ordering	DStream ordering	Not guaranteed	Partition order	Not guaranteed	Partition order
Processing Model	micro-batching	micro-batching	event-at-a-time	event-at-a-time	event-at-a-time
Processing Guarantees	exactly-once	at-least-once, exactly-once	exactly-once	exactly-once	at-least-once
PL Support	Scala, Python, Java, R, C#, F#	Use with any language	Java, C/C++, Go, .NET, Python, Ruby	Scala, Python, Java, SQL	Java, Scala
Recovery	self-recovery	Checkpoint	Checkpoint	Checkpoint	Checkpoint
State Management	stateless	in-memory state	Local state	stateful	stateless stateful
Written in	Scala	Clojure, Java	Scala, Java	Java, Scala	Scala, Java
Watermark	Yes	Yes	No	Yes	No
Quality Attributes	Fault tolerance, Scalability, Reliability, High Throughput, No data loss (Durability)				

Table 1: Data Acquisition Technologies

Apache Spark is one of the most widely used open-source data processing engines, initially created by Matei Zaharia in 2009 at UC Berkeley's AMPLab and later contributed to the Apache Software Foundation in 2013. It supports both streaming and batch data processing, delivering high-performance analytics. At its core, Apache Spark uses the Resilient Distributed Dataset (RDD) abstraction to handle and process data from various formats and sources. Its ability to process data quickly is attributed to its efficient use of distributed in-memory data structures (RDDs) and its reduced cost of data shuffling. Since its inception, Apache Spark's popularity has grown significantly, and its adoption is expected to continue rising due to its robust community support, powerful capabilities, and versatile features. Additionally, Spark provides libraries for machine learning, fast SQL queries, and advanced data analytics.

Apache Storm is a real-time data processing framework originally developed by Nathan Marz at BackType in 2011 and later open-sourced by Twitter. It is a low-latency distributed system for real-time stream processing, enabling data analytics before the data is stored. Apache Storm reliably transforms input streams into new output streams through its core components: spouts (which emit tuple streams) and bolts (which process tuples and generate new streams). A Storm cluster can execute multiple worker processes by leveraging spouts and bolts. Capable of handling tens of thousands of messages per second, Storm integrates seamlessly with existing queuing and bandwidth systems. It is widely used for real-time analytics, continuous computation, and online machine learning.

Apache Kafka is a distributed messaging system developed by LinkedIn in 2010 to manage streaming activities capable of processing millions of messages per second, later open-sourced by the Apache Software Foundation. Kafka ensures zero message loss, exactly-once processing, and guaranteed message ordering, making it ideal for mission-critical applications, streaming analytics, and high-performance data pipelines. Its core components include topics (streams of messages), producers (publish messages to specific topics), brokers (store the published messages), and consumers (retrieve messages from brokers). Designed as an explicitly distributed system, Kafka supports multiple producers, brokers, and consumers based on the workload. All data in Kafka is persistently written to a log file on the filesystem, with messages referenced by their logical offsets in the log rather than explicit message IDs.

Apache Flink is a unified framework for stateful stream and batch data processing, developed in 2009 and later incubated as an Apache Project in 2014. Its main components are streams and transformations. Flink adopts a stream-first approach and employs Kappa architecture (using true streams) to enable automatic partitioning and caching. It is widely used for event-driven applications like fraud detection, anomaly detection, rule-based alerting, and business process monitoring, as well as for continuous ETL data pipelining and advanced data analytics.

Apache Samza is a scalable engine for real-time data processing that supports both streaming and batch processing. It was developed by LinkedIn and open-sourced by the Apache Software Foundation in 2013. Samza manages data streams using streams and partitions, which consist of ordered messages in key-value pairs. It offers flexible deployment options, allowing it to run in various environments, and supports processing and transforming data from any source. Samza can also be integrated as a client library in Java or Scala applications. Built to leverage Apache Kafka, Samza is often used for developing stateful applications that perform real-time data processing and continuous computation.

Conclusion:

The comparative analysis of Apache Spark, Apache Storm, Apache Kafka, Apache Flink, and Apache Samza highlights the diverse capabilities of these technologies in handling the complexities of big data acquisition. Each tool demonstrates unique strengths in scalability, fault tolerance, and real-time processing, making them suitable for specific use cases across industries. For example, Spark's support for micro-batching and high-performance analytics makes it an ideal choice for large-scale data processing, while Kafka's distributed messaging system ensures seamless data pipelines for mission-critical applications.

The structured examination reveals that no single technology can address all the challenges of big data. Organizations must carefully evaluate factors such as latency, processing guarantees, and compatibility with existing systems to select the most appropriate tool for their needs. Moreover, the integration of these technologies with advanced analytics and machine learning frameworks can further enhance their utility in addressing domain-specific challenges. By bridging the gap between academic research and industry requirements, this study aims to empower data engineers and practitioners with actionable insights into the strengths and trade-offs of various technologies.

Remaining Tasks:

1. **Case Studies from Target Domains:**

- Healthcare: Analyze how these technologies are utilized in monitoring patient vitals, predicting diseases, or streamlining hospital operations. Discuss practical implementations like real-time alerts for critical conditions using Apache Flink.
- Finance: Explore their role in detecting fraudulent transactions, managing high-frequency trading, or processing large-scale financial data streams with tools like Apache Storm or Kafka.
- IoT: Highlight applications such as optimizing sensor data, real-time traffic monitoring, or smart city solutions using Spark or Samza.

2. **Structured Comparison and Practical Insights**

- Creating a structured comparison of the technologies based on key real-world parameters like cost, ease of integration, and scalability.
- Providing practical guidelines or decision-making frameworks for choosing the right technology for specific challenges.

3. **Recommendations for Industry Practitioners:**

- Suggesting best practices for integrating these tools within existing big data infrastructures.
- Discuss future trends and enhancements expected in these technologies to address evolving data challenges.

References:

[1] S. Sakr, "Big Data Processing Stacks," in IT Professional, vol. 19, no. 1, pp. 34-41, Jan.-Feb. 2017, doi: 10.1109/MITP.2017.6.

[2] Ahmed Oussous, Fatima-Zahra Benjelloun, Ayoub Ait Lahcen, Samir Belfkih, Big Data technologies: A survey, Journal of King Saud University - Computer and Information Sciences, Volume 30, Issue 4, 2018, https://doi.org/10.1016/j.jksuci.2017.06.001.

[3] L. Rodríguez-Mazahua, C.-A. Rodríguez-Enríquez, J. L. Sánchez-Cervantes, J. Cervantes, J. L. García-Alcaraz, and G. Alor-Hernández, ''A general perspective of big data: Applications, tools, challenges and trends,'' *J. Supercomput.*, vol. 72, no. 8, pp. 3073–3113, Aug. 2016, doi: 10.1007/s11227-015-1501-1.

[4] K. Venkatram and M. A. Geetha, ''Review on big data & analytics Concepts, philosophy, process and applications,'' Cybern. Inf. Technol., vol. 17, no. 2, pp. 3–27, Jun. 2017, doi: 10.1515/cait-2017-0013.

[5] A. C. Ikegwu, H. F. Nweke, C. V. Anikwe, U. R. Alo, and O. R. Okonkwo, ''Big data analytics for data-driven industry: A review of data sources, tools, challenges, solutions, and research directions,'' *Cluster Comput.*, vol. 25, no. 5, pp. 3343–3387, 2022, doi: 10.1007/s10586-022-03568-5.

[6] Rao, T Ramalingeswara & Mitra, Pabitra & Bhatt, Ravindara & Goswami, Adrijit. (2018). The big data system, components, tools and technologies: a survey. Knowledge and Information Systems. 10.1007/s10115-018-1248-0-.

[7] F. Bajaber, S. Sakr, O. Batarfi, A. Altalhi, and A. Barnawi, ''Benchmarking big data systems: A survey,'' Comput. Commun., vol. 149, pp. 241–251, Jan. 2020, doi: 10.1016/j.comcom.2019.10.002.