Literature Seminar Report

1TD169 - Data Engineering I

Group 3

Team Members: Dafne Spccavento, Jonathan Queckfeldt, Shahab Sherveh, Adriana Purici, Amogh Sanjay

27-02-2025

Q1: Discuss and contrast the characteristics of batch vs. streaming applications, both from the problem point of view and from a technology point of view.

Batch processing is a method designed for handling large-scale datasets at regular time intervals, making it ideal for high-throughput applications. However, batch processing comes with high latency and can take hours or days to complete for large data volumes, making it less suitable for real-time processing (Shahrivari, 2014). In contrast, streaming applications are designed to process data directly as it arrives and handle data as a continuous stream. This makes streaming applications ideal for use cases such as fraud detection, stock market analysis and online recommendations, where you aim for real-time results on input data. However, stream processing requires more memory and introduces continuous system load (Benjelloun et al., 2020).

From a technological standpoint, batch processing often employs frameworks like Apache Hadoop, which are optimized for high-throughput and fault-tolerant processing of large datasets. These systems are efficient at performing complex computations over large datasets but are not designed for low-latency requirements. (Shahrivari, 2014).

Streaming applications use frameworks such as Apache Storm, Apache Flink, and Apache Spark Streaming to handle continuous streams of data. These are the three major open source alternatives (Lopez et al., 2016). Among these, Apache Storm plays a critical role at Twitter, enabling large-scale, real-time computation on user-generated data. Twitter's data consists of millions of real-time interactions, traditional batch processing would therefore be insufficient to meet their needs. Instead, a robust stream processing solution like Storm ensures that Twitter can analyze and act on user interactions instantly (Toshniwal et al., 2014).

Q2: Discuss in what way the role of data types and formats place on the technological solutions. Are there specialized tools that achieve high-performance and high usability for specific formats?

The data types and formats are building blocks of the data structure. It defines how each entry of our data is stored on the physical infrastructure and how it is accessed. Therefore, it plays a critical role since the whole point of working with data is to access specific entries. The choice of data types affects the size, accuracy and complexity. For instance, if we choose Float64 over Float32 the size of our data doubles therefore, when the data is already big enough, doubling has dire consequences.According to challenges related to big data mentioned in (Chan and Zhang 2014, 318-321) It could complicate and add to the size related challenges like Data capture and storage, Data transmission, Data curation, Data analysis and Data visualization.

On the other hand, by setting Float32 it might truncate the some decimal digits and if the data is huge it could propagate and lead to notable inaccurate results while doing Data analysis and Data visualization.

There are cases where the data types have non-atomic format, for example an array of float values or a matrix which is an array of arrays. Therefore the complexity increases and adds to Data curation, Data analysis and Data visualization challenges. For instance, it would be hard to query our data based on the rank of the matrices with classical DBMS. Specifically, scientific data often involves complex numerical data, such as multi-dimensional arrays and functions over such data, therefore SciSPARQL extension was introduced over SPARQL which is a schema free database with RDFs(Resource Description Framework) as entries (Andrejev & Risch, 2012, 1) (Andrejev et al., 2013). (Kersten et al., 2011) suggests 5 research areas that could improve querying scientific databases.

Q3: Try to, from a technological point of view, relate the following tools to each other, both historically and in the technological problems they have addressed. Is there some logical progression in a seemingly fragmented ecosystem? Apache Hadoop, HDFS, Hbase, BigTable, Cassandra, Twitter Storm, Twitter Heron, Apache Spark, Apache Kafka, Apache Flink

The need to process, store and handle large amounts of data was first recognized by Google. They created MapReduce and its implementation (Google MapReduce) along with Google File System (GFS) for data handling. Hadoop is an open-source implementation of MapReduce and the Hadoop Distributed File System (HDFS) is used to handle and process the data used by Hadoop. Hadoop and MapReduce are well suited for batch processing of large datasets using a distributed computer network (Shahrivari, 2014). However, Hadoop is poorly suited for interactive data processing (such as real time data analytics and processing) and iterative data processing (Zaharia et al. 2015). Its design also does not scale well with very large datasets and the processing time could take hours and days.

Traditional SQL databases have poor scalability and are not well suited for these massive datasets. NoSQL databases work very well with HDFS and similar file handling software for larger datasets. The introduction of Google's NoSQL software, Bigtable, led to the development of open-source NoSQL software such as Hbase and Cassandra. Hbase works on top of HDFS and can be integrated with the Hadoop ecosystem.

To overcome the limitations of Hadoop for real time data processing, Apache Spark was created. It enabled the processing of larger batches (GB vs MB) and reduced processing time. This enabled real time and interactive data processing. Apache Storm is a similar open-source framework which also scales well and is well suited for large datasets (A. Toshniwal et al., 2014). However debugging, scaling and making efficient use of Storm was not easy, so Apache Heron was developed as a replacement (Kulkarni et al., 2015). Apache Heron addresses these deficiencies and makes better use of cluster resources. Apache Kafka was developed to process data from multiple sources in real time, making better use of parallel resources. However, Spark is still used alongside Kafka as it still works well for batch processing and analytics. Apache Flink is a new software primarily designed for analytics of large datasets.

Q4: Why do you think NoSQL alternative such as MongoDB or Apache Cassandra has gained so much in popularity?

NoSQL databases have gained popularity due to their scalability, flexibility, and performance, making them well-suited for Big Data and distributed applications. Unlike traditional relational databases (RDBMS), which require fixed schemas and scale vertically by upgrading hardware, NoSQL solutions scale horizontally by distributing data across multiple servers, improving performance and fault tolerance (Chen & Zhang, 2014).

A major advantage of NoSQL is its schema flexibility, which allows for easier adaptation to evolving data models. MongoDB, for instance, is a document-based database that stores data in JSON/BSON format, making it highly suitable for web applications that require fast and dynamic data integration (Băzăran & Iosif, 2014). Similarly, Cassandra follows a columnar storage model, optimized for high-write workloads and ensuring high availability with no single point of failure (Araujo et al., 2021).

Another reason for NoSQL's adoption is its ability to efficiently handle semi-structured and unstructured data, which is increasingly common in modern applications. Many leading companies, including Netflix, Twitter, and LinkedIn, use NoSQL databases to manage large-scale, real-time analytics and distributed content delivery (Chen & Zhang, 2014). Additionally, NoSQL databases often follow the BASE (Basically Available, Soft-state, Eventually consistent) model, prioritizing availability and scalability over strict consistency. (Araujo et al., 2021).

Q5: Scalability is an important concept in distributed systems. Discuss this both from a theoretical basis and from a technological (i.e. what are current ways to achieve this). What is the difference between reactive and proactive autoscaling, and what are some of the current approaches/proposals to achieve it?

Scalability in distributed systems is a crucial challenge, particularly in data-intensive applications where large volumes of information must be processed efficiently. Traditional relational databases struggle with complex data types like multidimensional arrays. To address this, SciSPARQL extends SPARQL to enable efficient querying of large-scale scientific datasets, supporting both relational and distributed storage solutions like Chelonia. By representing large numeric arrays as proxy objects and retrieving data lazily, the system minimizes access and communication costs, allowing it to scale effectively as datasets grow (Andrejev et al., 2013).

Achieving scalability involves leveraging distributed storage, parallel computation, and optimized query processing techniques. SciSPARQL translates high-level queries into optimized retrieval operations, minimizing data movement. Chelonia enhances scalability by storing scientific data in a geographically distributed manner, reducing bottlenecks and enabling high-throughput access (Andrejev et al., 2013). Meanwhile, frameworks like Celery and MPI parallelize computational workloads across multiple nodes, ensuring efficient execution and load balancing (Lunacek, Braden, & Hauser, 2013). Similar principles apply to enterprise systems like Amazon's DynamoDB, which uses partitioning and replication for high availability and scalability (DeCandia et al., 2007).

Autoscaling, a key aspect of scalability, can be reactive or proactive. Reactive autoscaling, like the Horizontal Pod Autoscaler in Kubernetes, adjusts resources based on observed CPU utilization but often struggles with delayed responses (Kubernetes Documentation, 2025). In contrast, proactive auto scaling anticipates future demands, as seen in the Proactive Pod Autoscaler (PPA) for edge computing (Ju, Singh, & Toor, 2021). Using predictive models like ARIMA and LSTM, the PPA forecasts workload patterns, enabling timely scaling decisions and reducing latency. Beyond autoscaling, optimizing distributed protocols also enhances scalability. Rule-driven rewrites of protocols like Two-Phase Commit and Paxos significantly improve throughput, contributing to the overall efficiency of distributed systems (Chu et al., 2024)

References:

- A. Toshniwal et al., Storm@twitter, Proc. ACM SIGMOD Int. Conf. Manage. Data, pp. 147-156, 2014; https://doi.org/10.1145/2588555.2595641
- Andrejev, A., & Risch, T. (2012). Scientific SPARQL: Semantic Web Queries over Scientific Data. 2012 IEEE 28th International Conference on Data Engineering Workshops. DOI 10.1109/ICDEW.2012.67
- Andrejev, A., Toor, S., Hellander, A., Holmgren, S., & Risch, T. (2013, October). Scientific analysis by queries in extended SPARQL over a scalable e-Science data store. In *2013 IEEE 9th International Conference on e-Science* (pp. 98-106). IEEE.
- Araujo, J.M.A., Moura, A.C.E., da Silva, S.L.B., Holanda, M., Ribeiro, E.O., & da Silva, G.L. (2021). Comparative Performance Analysis of NoSQL Cassandra and MongoDB Databases. IEEE International Conference on Big Data.
- Benjelloun, S., et al. (2020). Big data processing: Batch-based processing and stream-based processing. 2020 Fourth International Conference on Intelligent Computing in Data Sciences (ICDS), 1–6
- Băzăran, C., & Iosif, C.S. (2014). The Transition from RDBMS to NoSQL: A Comparative Analysis of Three Popular Non-Relational Solutions: Cassandra, MongoDB, and Couchbase. Database Systems Journal, 5(2), 49-59.
- Chan, C.L. P., & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences, 275, 314-347.
- Chu, D. C., Panchapakesan, R., Laddad, S., Katahanas, L. E., Liu, C., Shivakumar, K., ... & Howard, H. (2024). Optimizing distributed protocols with query rewrites. *Proceedings of the ACM on Management of Data*, *2*(1), 1-25.
- Ju, L., Singh, P., & Toor, S. (2021, December). Proactive autoscaling for edge computing systems with kubernetes. In *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion* (pp. 1-8).
- Kersten, M. L., Idreos, S., Manegold, I., & Liarou, E. (2011). The Researcher's Guide to the Data Deluge: Querying a Scientific Database in Just a Few Seconds. Proceedings of the VLDB Endowment, 4(12).
- Kubernetes Documentation. (2025). *Kubernetes Documentation*. https://kubernetes.io/docs/
- Kulkarni, S et al. (2015). Twitter Heron: Stream Processing at Scale. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15). Association for Computing Machinery, New York, NY, USA, 239–250. https://doi.org/10.1145/2723372.274278
- Lopez, M. A., Lobato, A. G. P., & Duarte, O. C. M. B. (2016). A performance comparison of open-source stream processing platforms. 2016 IEEE Global Communications Conference (GLOBECOM), 1-6.

- Lunacek, M., Braden, J., & Hauser, T. (2013, September). The scaling of many-task computing approaches in python on cluster supercomputers. In *2013 IEEE international conference on Cluster computing (cluster)* (pp. 1-8). IEEE.
- Shahrivari, S (2014). Beyond Batch Processing: Towards Real-Time and Streaming Big Data, Computers 2014, 3(4), 117-129; https://doi.org/10.3390/computers3040117
- Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., Bhagat, N., Mittal, S., & Ryaboy, D. (2014). Storm@twitter. ACM SIGMOD International Conference on Management of Data, 147–156.
- Zaharia, M et al. (2012). Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI'12). USENIX Association, USA, 2. https://www.usenix.org/conference/nsdi12/technicalsessions/presentation/zaharia