Journal of Statistics & Management Systems
ISSN 0972-0510 (Print), ISSN 2169-0014 (Online)
Vol. 26 (2023), No. 1, pp. 169–175
DOI : 10.47974/JSMS-956

RDBMS to distributed in-memory data grid: A paradigm shift for improved performance

Biswaranjan Jena [§]
Debahuti Mishra [†]
Smita Prava Mishra *
Department of Computer Science and Engineering
Siksha O Anusandhan (Deemed to be Univesity)
Odisha
India

Abstract

As per available Cisco VNI report [1], global IP traffic will grow more than three times and global network device connections will be more than 28.5 billion. Availability of high-speed network and connectivity will lead to more online business opportunities to enterprises and in turn will generate large data. Sometimes these large datasets need to be processed in near real time to do some critical analysis. Traditional databases are not designed to do real time data analysis and have challenges in terms of performance and throughput. This paper describes various widely used data storage tools and frameworks currently available and how to achieve the best performance while processing large datasets using in memory data grid. It briefly covers the evolution of in-memory data grid from traditional relational database management systems. It also provides a high-level feature comparison across available in-memory database solutions and how they provide the best of solution for processing real time data beyond Hadoop referencing use cases like fraud detection and genetic data processing. It also emphasizes on leveraging power of GPU alongside CPU to achieve high performance and throughput while processing large data with multiple iterations.

Subject Classification: *68P20.*

Keywords: *Ignite, Hadoop, Hazlecast, IMDG, IMDB, Bigdata.*

1. Introduction

Data is everywhere and is the backbone of any organization or enterprise. With evolution of new technologies like IOT, sensors, RFID

[§]*E-mail:* biswarjena@gmail.com
[†] *E-mail:* debahutimishra@soa.ac.in
**E-mail:* smitamishra@soa.ac.in (Corresponding Author)

based devices and their integration to enterprise systems TBs of data is getting generated. Social integration is very common in enterprises where they connect to millions of users or subscribers either to understand the challenges or collect feedback or reaching out them to sell their products and solutions. With increase in data generation strategies and sources processing these large chunks of data has its own challenges. In this paper we will analyze various tools and frameworks to process and store large amount of data for better performances as per needs of applications, organizations or enterprises.

2. Data Stores

Data can be categorized as structured and unstructured and can be stored on different datastores based on its size and utility. Structured data have predefined structure which is easy to process and generated as part of transactions. Unstructured data like pdf documents, images, videos are large data objects which need lot of processing power. Based on processing and throughput requirement various datastores are used as described in this section.

2.1 *RDBMS as Data Store*

Relational Database Management System (RDBMS) follows a standard table like structure for data storage. It uses standard SQL based queries to perform Create, Read, Update, Delete (CRUD) operations. These data physically exist in the file system of server disks space where RDBMS setup is done. It can be deployed both in centralized and decentralized manner but traditionally enterprises prefer to setup in centralized mode to avoid additional complexity and overhead of distributed database management. Multiple database servers are setup in master slave mode where all transactional data is written to master node and then the data is replicated to slave node for ensuring no data loss. To fulfill high performance and throughput requirement databases are scaled vertically by assigning additional resources which is always limited to maximum capacity the server supports. Concurrent read write performance is limited by the lock acquired on the data to maintain data integrity and has its overhead of disk-based processing.

2.2 *Big-Data as Data Store*

Apache Hadoop [2] is one of the most widely used open source big-data database which has the capability to handle large datasets on top of distributed cluster technology using simple programming interfaces.

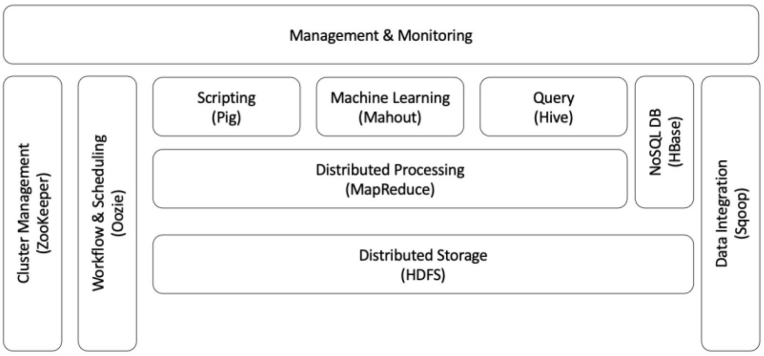

Figure 1

Hadoop Logical Architecture

Due to clustering, its simple to scale by adding additional cluster nodes. It supports MapReduce which is a standard framework to process large datasets in reliable and fault-tolerant way. The large data is divided into chunks which are processed using map tasks in complete parallel manner. It has its own file system called HDFS [3] which provides high throughput access to the streaming datasets and can store data in multiple distributed disks. Major components of Hadoop ecosystem are represented in Figure 1. Hadoop is specially designed to process large data leveraging its distributed multi node cluster, it has its own overhead of administration and setup cost. It's still inefficient compared to in-memory databases due to high disk-based lookup cost and less suitable for stream data processing and real time data analysis.

2.3 *In-Memory Distributed Database*

Disk based processing is slower as compared to in-memory processing because of the additional overhead of Input Output Operations (IOPs). Frameworks like Apache Spark [4], Apache Flink [5], Apache Storm [6] are most popular in terms of in-memory data processing and supporting machine-learning and stream processing using distributed in-memory architecture. Storm and Flink performs well for real-time data processing whereas Spark support few machine learning algorithms but not preferrable for real-time data processing and don't have its own filesystem. Irrespective of platform and limitations its proven that in-memory distributed processing technology gives superior performance as compared to disk-based processing. Also advance clustering technology provides better data availability via replication across multiple nodes,

additional processing capability by utilizing resources of cluster nodes and executing tasks parallelly. In-memory database can also be used to cache most frequently used data and make it available to applications. There are Multiplein-Memory Databases (IMDB) [7] are available in market. IMDBs are best suited where large data processing needs to be carried out in small interval with highest throughput but still lack the power of grid where processing load is shared across different nodes.

3. Comparative Analysis

In-Memory Data Grid (IMDG) supports a grid of distributed nodes to support parallel processing and high availability. IMDG also support in memory caching reducing disk-based access to databases for most frequently used data. Apache Ignite [8], Redis [9], Hazlecast [10] are few widely used IMDBs which support grid topology. In Grid topology set of nodes run as part of a cluster share the processing load as represented in Figure 2. Apache Ignite is distributed in memory grid which supports range of use cases like caching for fast data access, transactional and analytical processing, high volume data processing, lambda architecture, microservices support, first level and second level cache, big-data accelerators, machine learning, collocated joins etc.

Apache Ignite is easy to use, support clustering and sql based query and supports multiple caching patterns. Data replication across nodes with support to query on collocated data makes it much efficient and avoid network round trips. It provides various design like service grid, compute grid which can be specifically designed for handling services and computations. Ignite also provides APIs using which data can be easily fetched from key-value store. Majid *et. al.* [11] in their paper have gone a step ahead where they have leveraged the processing capability of GPUs

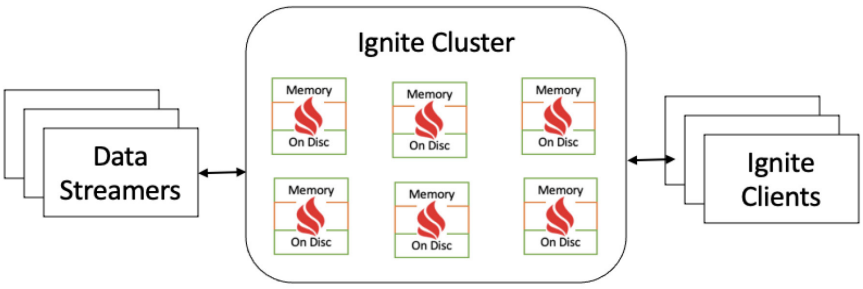

Figure 2

In-memory Ignite cluster

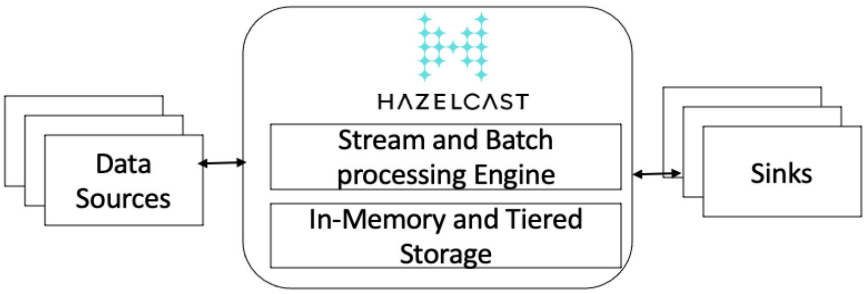

Figure 3

Hazlecast In-Memory Architecture

alongside CPU and shown the performance capabilities while processing genetic algorithm as various compute tasks in Ignite and getting it executed across various Ignite cluster nodes. Rishikesh Bansod *et. al.* [12] have demonstrated the high throughput while analyzing discrepancies and detecting fraud for large amount of transactional data generated in a trading system.

Redis is a widely used open source in-memory data structure which can be used as database, cache and broker which supports atomic transactions. As a message broker it supports publish-subscribe model where subscribers consume message from published channels. It supports setting timeout on key as it works as a key-value store. After expiration of timeout period, the key is automatically deleted. It supports least Recently Used (LRU) strategy for data eviction.

Hazlecast supports Directed Acyclic Graphs (DAGs) strategy to coordinate the processing, similar to the strategy Apache Spark uses but it has the advantage of in-memory speed to complete the task much faster. It provides common programming interface to process both batch and stream data.It can be embedded into data processing microservices to leverage cloud native capabilities too. As per the benchmark report available, it can process one billion events per second with the limited hardware as compared to other competing technologies. Hazlecast provides superior features compared to Redis, like automatic cluster handling, providing querying API support, full streaming stack instead of pub-sub messaging to streaming data with support to popular languages like Java, Python. The processed results can easily be integrated to analytics and enterprise systems as given in Figure 3.

Table 1

Standard features comparison

Features	Apache Ignite	Redis Enterprise	Hazlecast
Native ANSI SQL Support	Supported	Not Supported	Supported
Distributed ACID Support	Supported	Limited Support	Limited Support
Multiple Caching Pattern	Supported	Limited Support	Supported
Collocated processing	Supported	Not Supported	Supported
RDBMS-Hadoop integration	Supported	Supported	Supported
Full In-Mem computing Sol.	Supported	Limited Support	Not Supported
Built in Machine Learning	Supported	Supported	Supported
Open source support	Supported	Supported	Supported
Query API Support	Supported	Not Supported	Supported

Each of the tools mentioned in this paper are industry leaders but based on actual usage , ease of use and key features high level comparison is shown in Table 1.

4. Conclusion

This paper analyzes various tools and frameworks used for processing large data and describes the evolution of various data storage strategies starting from relational databases to bigdata to in-memory database. RDBMS systems are preferred as OLTP system can do limited data handling. Big-Data systems like Hadoop can process large datasets using its cluster architecture and provide tremendous performance improvement along with high availability via data replication and parallel computing strategies but still lacks the fast-processing capability compared to IMDB. Apache Ignite is a perfect blend of solution which adopts clustering strategy like Hadoop and high performance in-memory computing. This makes it suitable for processing large amount of data with highest throughput with high availability especially while doing real time data analysis. As per our preliminary study Apache Ignite provides superior functionality and services compared to other IMDG solutions. Subsequent study can be done by doing benchmarking on large datasets against these tools.

References

[1] CISCO Visual Networking Index, https://twiki.cern.ch/twiki/pub/HEPIX/TechwatchNetwork/HtwNetworkDocuments/white-paper-c11-741490.pdf.

[2] Vimala Ithayan, J., and C. Sundar, "A Secured Healthcare Management and Service Retrieval for Society Over Apache Spark Hadoop Environment", *IETE Journal of Research*, 2021, pp. 1-20.

[3] H. Fei., C. Yang., Y. Jiang., Y. Li., W. Song., D. Q. Duffy., J. L. Schnase, and T. Lee, "A hierarchical indexing strategy for optimizing Apache Spark with HDFS to efficiently query big geospatial raster data" *International Journal of Digital Earth*, Vol. 13, No. 3, 2020, pp. 410-428.

[4] E. Shaikh, I. Mohiuddin, Y. Alufaisan and I.Nahvi, "Apache Spark: A Big Data Processing Engine", *2019 2nd IEEE Middle East and North Africa Communications Conference*(MENACOMM). doi:10.1109/menacomm46666.2019.8988541.

[5] A. Katsifodimos and S. Schelter, "Apache Flink: Stream Analytics at Scale", *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, doi:10.1109/ic2ew.2016.56.

[6] R. Evans, "Apache Storm, a Hands on Tutorial",*2015 IEEE International Conference on Cloud Engineering*. doi:10.1109/ic2e.2015.67.

[7] List of In-Memory DBs, https://en.wikipedia.org/wiki/List_of_in-memory_databases.

[8] A. Tapekhin, I. Bogomolov, and O. Velikanov, "*Analysis of Consistency for In Memory Data Grid Apache Ignite*". 2019 Ivannikov Memorial Workshop (IVMEM). doi:10.1109/ivmem.2019.00013.

[9] T. Priovolos, S. Maroulis and V.Kalogeraki , "*A Framework for Managing an Elastic Redis Cache*", 2019 38th Symposium on Reliable Distributed Systems (SRDS). doi:10.1109/srds47363.2019.000.

[10] Hazlecast, https://hazelcast.com/.

[11] A. H. Sojoodi, M. Salimi Beni and F. Khunjush , "Ignite-GPU: a GPU-enabled in-memory computing architecture on clusters", *The Journal of Supercomputing*, 2020, pp. 1-28.

[12] R.Bansod, S. Kadarkar, R. Virk, M. Raval, R. Rashinkarand M. Nambiar, "High Performance Distributed In-Memory Architectures For Trade Surveillance System", *17th International Symposium on Parallel and Distributed Computing*, 2018, pp. 101-108.