Evaluation of NoSQL Databases

Hia Wei Qi
School of Computer Sciences
Universiti Sains Malaysia
Simpang Ampat, Penang
hiaweiqi@student.usm.my

Wong Leh Ling
School of Computer Sciences
Universiti Sains Malaysia
Sibu, Sarawak
wonglehling0223@student.usm.my

Tan Yijoe
School of Computer Sciences
Universiti Sains Malaysia
Butterworth, Penang
yyijoetan@student.usm.my

***Abstract*—** NoSQL databases is able to handle unstructured and semi-structured data due to they are flexible and efficient. Due to this reason, this type of databases started to gain significant traction in various industries. The biggest difference among traditional relational database and NoSQL databases are NoSQL databases do not required a fixed schema, and thus they are allowed to scale horizontally and can handle vast amounts of data. This flexibility makes NoSQL databases suitable for applications like big data applications, real-time web applications and distributed systems.

This report will study four types of NoSQL databases, which are Amazon DynamoDB, MongoDB, Cassandra and Neo4j. Each database will be reviewed based on its history, its strengths and weaknesses, its key features. Furthermore, the report also provided a comprehensive guide for selecting the correct and appropriate NoSQL database based on specific needs.

Keywords—NoSQL, graph models, key-value pair, document-based, flexible database, no-schema

I. INTRODUCTION

NoSQL databases are designed to handle unstructured and semi-structured data. It provides flexible schema design, scalability, and high performance. Nowadays, the demand for more flexible, scalable, and efficient data storage solutions has been increased due to the explosion of big data. Unlike traditional relational database, NoSQL database provide a variety of data models, making them ideal for big data and real-time web applications. NoSQL databases are designed to work on distributed architecture, and also provide high availability and fault tolerance. The developers can scale out across multiple servers.

There are many primary types of NoSQL databases such as document stores, key-values stores, graph database, and column-family stores. Document stores allow developer to store data in JSON format which in more dynamic format. One of the most popular databases of document store is MongoDB. For key-value stores such as Amazon DynamoDB, it is used key-value pair mechanism to store the data which is simple and fast. Cassandra, one of the column-family stores are designed to store data with high availability and scalability. It is suitable for organization that need to distributed data across multiple data centers. Graph database such as Neo4j are built to manage and query high interconnect data. It provides a powerful query capabilities and data modeling to applications that need data analysis and connections.

II. REVIEWS OF 4 TYPES OF NO SQL DATABASES

This section reviews the Amazon DynamoDB, MongoDB, Cassandra and Neo4j.

A. Amazon DynamoDB

i. History and Evolution of Amazon DynamoDB

Amazon DynamoDB is a serverless, fully managed NoSQL database (Key-value store) service that provides desired levels of availability and performance. A key-value store indicates that the data is stored as a collection of key-value pairs. In simple words, users can access the data by specifying the key.

Amazon DynamoDB was launched by Amazon Web Services (AWS) in 2012. It was inspired by the seminal Dynamo white paper (DeCandia et al., 2007). The paper outlined a distributed key-value store designed for internal use to address the scalability and reliability issues Initially, DynamoDB was used internally by the Amazon and it was a completely different customer-oriented Database as a Service (DBaaS) that runs on Amazon Web Services (AWS) Elastic Compute Cloud (EC2) instances. DynamoDB builds on these concepts but making it a more integrated and providing a more user-friendly experience for the users.

DynamoDB has continued to evolute after the initial release in January 2012, introducing features such as global tables for multi-region replication, on-demand capacity mode, and support for transactions, extending its capabilities and performance. In 2013, Global Secondary Indexes (GSIs) were introduced. By 2014, support for JSON document storage and fine-grained access control were added. Between 2015 and 2017, DynamoDB introduced Streams for change data capture, cross-region replication, on-demand backups, auto-scaling, and global tables. From 2018 to 2020, significant updates included transactions for ACID operations, on-demand capacity mode, NoSQL Workbench, and DynamoDB Local. Recent updates between 2021 and 2023 include PartiQL for SQL-compatible querying, AWS Glue Elastic Views, and enhanced client SDKs and security features.

This is the document history that includes all the important changes in each release of the DynamoDB since 2012.

ii. Highlights of Amazon DynamoDB

Dynamo is highly scalable (DeCandia et al., 2007) as it allows users to scale up or down based on their request load without requiring user's capital investment in hardware. Dynamo allows users to customize their storage system to meet their desired performance, durability and consistency.

DynamoDB supports key-value NoSQL database, making it the preferred database for simple and fast data models, such as managing user profile data, web session

information and any other applications that need to quickly retrieve any amount of data at Internet scale.

Dynamo provides durability and crash recovery. It uses write-ahead logs, which helps to record the data writes before they occur. This act as a backup for any event of crash, DynamoDB can use this write-ahead logs to reconstruct the lost data writes.

iii. Evaluation of Amazon DynamoDB

DynamoDB is highly valued for its availability and scalability, making it ideal for high-traffic web applications, IoT, gaming, and mobile applications. DynamoDB regularly tests its resilience to node, rack and availability zone failures. DynamoDB performs power-off test to verify the validity of data stored in the database. DynamoDB promises a 99.999% availability for global tables and 99.99% availability for regional tables.

In the other hand, DynamoDB can become expensive at scale and may not offer as much flexibility as some developers need for complex queries and analytics.

B. MongoDB

i. History and Evolution of MongoDB

MongoDB is a an open-source document-oriented NoSQL database (Document-based store) server developed in C++. It is designed to handle document-oriented storage, providing high performance, availability, and scalability. A document-based store is a type of database that stores data in JSON-like documents. This indicates that users have more flexibility in storing complex data structure with using MongoDB.

It was developed in 2007 by a New-York based organization and first released in 2009. It was developed as a PaaS (Platform as a service) initially. MongoDB's schema-less design allows it to store complex data structures and high adaptability in changing application requirements without the need for extensive schema migrations.

MongoDB has now evolved, with the introduction of new features such as sharding for horizontal scalability, replica sets for high availability, and an aggregation framework for advanced querying capabilities. MongoDB Atlas, a cloud-based database as a service, was launched in 2016, offering a fully managed version of MongoDB. By now, the latest version of MongoDB is 7.0. It includes many improvements that enhance the performance, security, and usability for the users.

ii. Highlights of MongoDB

Replication

MongoDB achieves replication with replica sets. A replica set consists of groups of instances that maintain same data sets. Each replica-set consist of primary and multiple secondary nodes. All writes and reads operation are done on the primary node. Secondary nodes replicate a copy of the data of the primary node by using built-in replication. When a primary node fails, the secondary nodes will automatically determine which secondary should become the primary. This ensures high availability and redundancy.

Load Balancing

MongoDB allows scaling horizontally by supporting sharding. Sharding is a data distributing method that distributes data across multiple servers. This method offers very high scalability by partitioning the data across shards.

File Storage

MongoDB can be also used as a file system as it includes GridFS (Grid File System), a specification for storing and retrieving large files over multiple machines. GridFS divides a file into smaller parts and stores each parts separately within a collection. GridFS is beneficial for applications that require efficient storage and retrieval of large files, as it allows parallel processing of file chunks and simplifies file management within the database.

iii. Evaluation of MongoDB

MongoDB is highly flexible, ideal for applications requiring fast iterative development and dynamic schema changes. However, it may require more manual intervention for scaling and performance tuning compared to managed services like DynamoDB. It is true that MongoDB offers high level of flexibility when working with schema designs, but the choice of schema design employed by an application built on top of MongoDB will still ultimately determine whether the application is able to meet its performance objectives.

C. Cassandra

i. History and Evolution of Cassandra

Apache Cassandra is an open-source NoSQL distributed database that utilized by thousands of companies for scalability and high availability without sacrificing performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data. It is built to manage a vast volume of data over various cloud data centres.

In 2008, Cassandra was created for Facebook to handle massive amounts of data that generated by the Inbox Search feature. At that time, it is difficult for traditional relational databases to handle high write throughput and data volumes. Thus, Facebook engineers, Avinash Lakshman and Prashant Malik has got the idea from Google's Bigtable and Amazon's Dynamo papers to design Cassandra that can handle massive amounts of data across multiple cloud data centres. Cassandra has been released as an Apache project in July 2008.

In 2009, the project was moved to the Apache Software Foundation's Incubator. Cassandra was graduated from the Apache Incubator in February 2010 as an Apache Top-Level Project. It was successfully launched in October 2010 with the first stable release.

Cassandra has gained various adoption and community growth in 2011 to 2014. The Cassandra community was grew rapidly that had release Cassandra 1.0, 1.2, and 2.0. In these release, there are various features introduced such as virtual node support, improved query language (CQL), and lightweight transactions.

In 2015, Cassandra continued to release version 3.0 and version 4.0 in 2019 that has improved in storage engine, and better support for hybrid cloud deployments. There are many

organizations apply Cassandra as their database solution to handle large-scale data like Instagram, Apple, and Walmart.

Cassandra has introduced the latest major release of 4.0 in 2020. It also continues to developed and maintained actively by the Apache Software Foundation so that the performance, scalability, and enterprise-grade capabilities can be improved.

ii. Highlights of Cassandra

The primary architecture of Cassandra is made up of a cluster of nodes. It also structure as peer-to-peer system and closely similar to DynamoDB and Google Bigtable. In a cluster, each node plays a consistent work. Each node operates independently and consistently connected to other nodes. Each node is also the exact point where can store specific data. Each node in cluster will accept scan and write requests. If one of the node failure, other nodes in the network will handle the read/write requests. Besides that, it can expand to house more data easily. The quantity of data can be doubled by the system without overloading it by adding extra nodes. This capacity for dynamic scaling can goes both ways that is the developers can shrink the database system by lowering the number of nodes.

Additionally, data is stored and retrieved through a partitioning system in Cassandra. The location of a data set's primary copy is chosen by a partitioner. Every node is responsible for a set of tokens based on a partition key in which the partition key is the way to determine the place the data is stored. A hash function is added to the partition key as data enters a cluster. The coordinator node will send the data to the node with the same token under the partition. The data can replicate across nodes. The secondary nodes also called as replica node. The replication factor (RF) will determine how many replica nodes are needed for certain data collection.

Table below shows some of the key features of Cassandra.

Key Features	Description
Open-source availability	Cassandra is the open-source database hosted by Apache and it is free for anyone.
Distributed	It does not have master node, so it has no bottleneck slow down the process. Due to this feature, losing one node will not affect the system's performance significantly.
Scalability	The database system can be adjusted by adding or removing extra nodes to suit the dynamic needs.
Fault tolerance	The data replicative ability of Cassandra makes it tolerant of faults in the system.
Schema free	Cassandra is a schema-optional data model. It allows to create as many rows and columns as necessary.

iii. Evaluation of Cassandra

Strengths of Cassandra

There are many advantages of using Cassandra database. First, one of the main advantages of Cassandra is it can scale easily. The distributed architecture of Cassandra allows it to scale linearly by simply adding or removing nodes to the cluster. It helps organizations handle vast volumes of data and the database that can grow with their needs.

Besides that, the architecture of Cassandra helps to eliminate single points of failure. If one of the nodes fails, another node in the cluster can take over without interruption. Data replications feature also ensures the data is always available.

Weaknesses of Cassandra

Cassandra is a complex database system that is difficult to set up and maintain. The distributed architecture requires careful planning and configuration to make sure of high performance.

Additionally, since Cassandra uses query language, it is difficult for developers to develop applications on Cassandra.

Cassandra also does not support joins which will make it difficult to query data. Cassandra can use denormalization instead so that can optimize performance but it will make data modeling more complex.

D. Neo4j

i. History and Evolution of Neo4j

Neo Technology developed a type of graph database management system, called Neo4j in 2000. Neo4j is used to store and manage data, in a graph-based model. Where in this model, data are represented as nodes and relationships, which stands for entities and connections between entities respectively.

The origins of Neo4j is tightly related to a Swedish author and developer, Johan Svensson and his doctoral thesis work. This is because his research was mainly focused to explore new ways to store and also query the data more efficiently, when it comes to complex and interconnected data structures. And as result, it turns into development of the property graph model, which are the foundation of Neo4j.

In 2003, Neo Technology was founded by Svensson along with his colleagues. Then, they began working on the first version of Neo4j. After 4 years, which is in year 2007, the first release of Neo4j, version 0.9 was made available. After that, Neo4j started to gain traction from various industries over the next few years after it published. These fields included social networking, fraud detection and also recommendation engines. This is because Neo4j is able to manage and query highly connected data efficiently and this is crucial for these industries.

The company, Neo Technology continued to develop and enhance Neo4j as it gained the popularity, by developing and developing new features and improve its performance and expand its capabilities. Nowadays, Neo4j become very popular database in various industries, which include finance, healthcare, telecommunications, and e-commerce. Moreover, it also has active community of developers and contributors to help to make Neo4j better.

ii. Highlights of Neo4j

Neo4j is a powerful and versatile graph database management system that offers several notable features and benefits. Table below shows some of the key highlights of Neo4j:

Graph Data Model	It is built based on the property graph model and data is represented as nodes (entities), and

	relationships (entities connections). Since it mimics how human perceive and understand the connected data, it makes complex relationships modelling and querying easier.
Cypher Query Language	Neo4j used a declarative language called Cypher. It was inspired by SQL query language and was specially designed for graph data querying. Other than that, Cypher queries are also easy to read and expressed. This allows the developers traverse and manipulate graph data structures easily and efficiently.
ACID Compliance	Neo4j provides ACID compliance. It stands for Atomicity, Consistency, Isolation, and Durability. This compliance ensures the data integrity and also the transactions are processed reliably and consistently.
High Performance	Neo4j has high-performance in graph traversals and queries. It can store data efficiently and uses efficiently indexing techniques. As result, it allows fast retrieval and manipulation of connected data.
Scalability	With sharding and clustering capabilities, Neo4j supports horizontal scaling. It also enables Neo4j to handle large volumes of data and support high velocity workloads.
Native Graph Analytics	Neo4j provides built-in graph algorithms and also analytics capabilities. It allows users to perform complex graph data analysis, which includes path finding, centrality measures, community detection, etc.
Flexible Data Model	Other than graph data, Neo4j also able to store and query semi-structured data. This allows Neo4j flexible and versatile to handle various data formats.
Ecosystem and Integrations	Neo4j provided a lot of tools, drivers and libraries for various programming languages and frameworks. It also allows developer to use hybrid data architectures because it can integrates seamlessly with other data storage systems.
Visualization Tools	Neo4j provides useful and handy visualization tools to let users explore visually and analyse graph data, for example Neo4j Browser and Neo4j Bloom. It lets users to understand and communicate complex relationships much easier.
Enterprise-Ready Features	Neo4j provides some enterprise-grade features. These features include role-based access control, backup and recovery mechanisms, monitoring and alerting tools. Furthermore, it also support for various deployment options, including on-premises, cloud and hybrid.

iii. Evaluation of Neo4j

Strengths of Neo4j

Efficient Graph Traversals	Neo4j performs well at traversing and querying the graph data and highly connected data. This makes it a very good choice of database to use in applications that involves a lot of relationships or consists of complex relationships. These include social networks, fraud detection systems and recommendation engines.
Expressive Query Language	The query language of Neo4j, Cypher provides a powerful and easier way to query and manipulate graph data and this reduces the complexity of working with graph data that is in connected data structures.
Scalability and Performance	With sharding and clustering, Neo4j can scale up robustly and this makes it able to handle large volume of data and high velocity of workloads, but with maintained good performance.
Data Integrity	Since Neo4j have ACID compliance, data consistency and reliability was ensured. This make it suitable for mission-critical applications that require transactional integrity.
Ecosystem and Community	Neo4j built a community and provided wide variety of tools, integrations and libraries. As result, the development was accelerated and overall learning curve was reduced.

Constraints of using Neo4j

Limited for Purely Tabular Data	Neo4j is not the best choice for purely tabular or relational data because it cannot handle structured data as efficient as traditional relational databases, although it can handle semi-structured data.
Query Complexity	For complex queries and manipulation of graph data, Cypher can also be difficult to read and

	maintain, especially for developers that are new to graph databases.
Specialized Skillset	Since Neo4j is different compared to traditional relational databases, a different mindset is required to work with graph databases, and this might lead to a steeper learning curve.
Memory Footprint	Neo4j required relatively large memory footprint compared to some other database systems. So, careful resource planning and management for large-scale deployments are required.
Licensing and Cost	Although Neo4j provided a community edition for free, the enterprise edition that comes with advanced features and support can also be costly for large-scale deployments.

III. Discussion

As discussion, the use cases need to be considered seriously when choosing certain NoSQL database.

For applications that need high throughput and seamless integration with AWS services, DynamoDB is recommended. It may be a good choice in this use case because it uses key-value pair and high throughput. This made DynamoDB ideal for real-time applications for example bidding systems and gaming platforms.

On the other hand, for projects that need to have flexible data models or required rapid development cycles, MongoDB is more suitable for this use case. These projects include content management systems or mobile apps. It has a significant advantage due to its ability to handle dynamic schemas.

Next, for applications that need high availability and also scalability, Cassandra is best choice in such use case. These projects include IoT and recommendation systems. This is because Cassandra is particularly effective for write-heavy workloads.

Lastly, if the projects need to store or manipulate complex and interconnected data, Neo4j is the top choice in this use case. This is because Neo4j has powerful query capabilities, and it uses graph model. This use case includes systems like fraud detection, social networks and recommendation engines.

IV. conclusion

To manage unstructured or semi-structured data, NoSQL databases is the excellent choice because it provides flexible and efficient solutions. As conclusion, Amazon DynamoDB is ideal for projects that need high-throughput and low-latency, such as real-time bidding and also gaming platforms. MongoDB is ideal for dynamic applications like content management system and recommendation systems due to its flexibility and ease of use. Next, Cassandra is ideal for projects that need high availability and scalability and Neo4j is excellent in handling data in graph model due to its abilities to perform complex queries and data management easily and efficiently. Finally, to select the right NoSQL database, we need to consider the specific requirements and constraints of projects involved.

V. clear division of group members' roles

Hia Wei Qi (Team Leader)

- Leads the team, ensure that everyone stays on track with their tasks and deadlines.
- Compiling the final report by integrating contributions from all team members.
- Provides support and guidance to the team.
- Concludes the report.

Wong Leh Ling

- Conducts research to find relevant papers related to the topic.
- Ensure all the papers are appropriate and contribute to the understanding and analysis of the chosen NoSQL Databases.
- Ensure all the members understand the papers and cites all sources.
- Discuss with the team and gives recommendations on the reviews of 4 NoSQL Databases.

Tan Yijoe

- Analyse the advantages and disadvantages of NoSQL Databases.
- Highlight the limitations and solutions of NoSQL Databases.
- Compare among NoSQL Databases.
- Provides use cases for each of the NoSQL Databases.

VI. References

[1] "What is dynamodb?," DynamoDB, explained., https://www.dynamodbguide.com/what-is-dynamo-db/ (accessed Jun. 6, 2024).

[2] Cassandra vs mongodb - difference between NoSQL databases - AWS, https://aws.amazon.com/compare/the-difference-between-cassandra-and-mongodb/ (accessed Jun. 6, 2024).

[3] A. S. Gillis and B. Botelho, "What is mongodb? features and how it works – techtarget definition," Data Management, https://www.techtarget.com/searchdatamanagement/definition/MongoDB (accessed Jun. 6, 2024).

[4] S. Robinson et al., "What is neo4j (Graph Dabatase)? complete overview of neo4j," Graphable, https://www.graphable.ai/software/what-is-neo4j-graph-database/ (accessed Jun. 6, 2024).

[5] C. Kemper, “Getting to know neo4j,” *Beginning Neo4j*, pp. 13–23, 2015. doi:10.1007/978-1-4842-1227-1_2

[6] J. J, “Cassandra is a better option for handling big data in a no-SQL database,” *International Journal of Research Publication and Reviews*, pp. 880–883, Sep. 2022. doi:10.55248/gengpi.2022.3.9.27

[7] G. Harrison and M. Harrison, “MongoDB architecture and concepts,” *MongoDB Performance Tuning*, pp. 13–32, 2021. doi:10.1007/978-1-4842-6879-7_2

[8] G. DeCandia *et al.*, “Dynamo,” *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 205–220, Oct. 2007. doi:10.1145/1323293.1294281