Ashar Kamal Raja

SIT 323

Task 7.3P

Introduction:

Cloud-native applications necessitate powerful data management systems capable of handling large amounts of data, scaling horizontally, and ensuring high availability and dependability. Event sourcing and traditional database techniques are two common strategies for data management in cloud-native applications. This report analyzes and compares various approaches for data management in cloud-native applications that employ MongoDB in terms of performance, scalability, reliability, and maintainability.

Database Management Approaches in Cloud Native Application Development

There are several database management approaches that can be employed in the context of cloud-native apps, including standard database approaches, MongoDB, and event sourcing.

1. **Traditional database approaches:**
   Traditional database techniques immediately maintain the present state of an application in a database. These approaches, particularly for compose queries, can be more rapid and relatively easy than event sourcing (Trigoso, 2018). Traditional database systems, on the other hand, can be more difficult to scale horizontally since sharing and replication are complex and need large infrastructure. Additionally, traditional database methods may not be as auditable and maintainable as event sourcing.
1. **MongoDB:**
   A NoSQL document-oriented database, MongoDB stores data as adaptable, semi-structured documents that resemble JSON (Prakash, 2021). It enables quick performance, high availability, and simple scaling. Applications with fragmented data and schemas that are adaptable are ideally suited for MongoDB. It is frequently utilized in IoT, handling content, and web-based applications.
2. **Event Sourcing:**
   The method known as "event sourcing" keeps track of all application state modifications as a collection of events. The application's state can be determined by replaying those instances. The transparency, expansion, and reliability of event sourcing are only a few benefits. Event sourcing can, however, potentially be slower and more complicated than traditional database methods, particularly for read queries. The requirement for an independent event collection in event sourcing can make the application architecture more difficult (Trigoso, 2018).

Comparison of Database Management Approaches:

There are a number of things to take into account when contrasting event sourcing versus conventional database techniques. When it comes to performance, traditional database techniques can handle write-intensive workloads better than event sourcing can handle read-intensive workloads. When it comes to scalability, event sourcing may be more capable of handling massive volumes of data while traditional database systems may be more difficult to scale horizontally (Saraswathi, 2020). Both systems can be made highly reliable from a reliability standpoint using strategies like replication and container orchestration, but event sourcing adds an extra level of dependability due to its auditability. Last but not least, traditional database systems may be easier to maintain, although event sourcing may in some circumstances offer superior maintainability, particularly when it comes to changing the state of the app without compromising past data (Singh, 2023).

Database Management Techniques:

When it comes to data management strategies, data caching can enhance performance by lowering the number of necessary queries to the database. Cache data must be updated, nevertheless, in order to

prevent outdated information (Saraswathi, 2020). By distributing data across several nodes, data partitioning can boost speed. However, monitoring and maintaining partitions might become more difficult. By duplicating data across several nodes, data replication can increase availability and read performance while also adding to the complexity of managing and maintaining replicas.

Recommendations:

Based on the results of this study, it is advised that cloud-native databases, such MongoDB, be used when developing applications in order to increase performance and dependability. Implementing data partitioning will spread data across more nodes and enhance performance. Utilizing data caching while keeping cached data current will further improve performance. Additionally, data replication should be put in place to increase accessibility and examine how the application will perform, with an emphasis on facilitating a consistency between replicas. Regular monitoring and optimization of queries, indexes, and other configurations should be carried out to track and enhance the efficiency of the database. To prevent data loss, backup and recovery processes should also be used. By putting security measures like encryption, access controls, and inspection in place, the developers should also take data security and compliance into account. Last but not least, the company should think about utilizing managed database services like MongoDB Atlas to streamline database maintenance and offer extra capabilities like automated backups, scaling, and security (Singh, 2023).

Conclusion:

For data management in cloud-native apps that employ MongoDB, both event sourcing and conventional database techniques have advantages and limitations. The application's needs, including data volume, structure, and access patterns, will determine which solution is best. To guarantee excellent speed, scalability, stability, and maintainability, it's crucial to apply the proper data management strategies, database technology, and container orchestration platforms. These recommended practices can help developers create cloud-native applications that can expand horizontally, manage large amounts of data, and guarantee high availability and reliability.

References:

Maison, M. (2019, March 19). An introduction to event sourcing. IBM developer. Retrieved from https://developer.ibm.com/articles/event-sourcing-introduction/

Prakash, V. (2021) Top 6 cloud-native application development trends that help businesses thrive. Available at: https://www.kellton.com/kellton-tech-blog/top-6-cloud-native-application-development-trends-that-help-businesses.

Singh, G. (2023) Role of cloud-native in managing Big Data Applications: Quick Guide, Continuous Intelligence with AI. Xenonstack Inc. Available at: https://www.xenonstack.com/blog/cloud-native-in-big-data.

Saraswathi, R. (2020) Four architecture choices for application development in the Digital age, IBM. Available at: https://www.ibm.com/cloud/blog/four-architecture-choices-for-application-development.

Trigoso, F. (2018) Event sourcing versus traditional architectures: Dropsource - Full Stack Mobile Development Agency, Dropsource. Available at: https://www.dropsource.com/blog/event-sourcing-versus-traditional-architectures/.