Advances in Computer, Signals and Systems (2024)
Clausius Scientific Press, Canada
DOI: 10.23977/acss.2024.080514
ISSN 2371-8838 Vol. 8 Num. 5

Challenges and Solutions of Distributed Transactions in Medical Software under Microservices Architecture

Jiangke Wu[*]

Hangzhou Truth AI Technology Co., Ltd, Hangzhou, Zhejiang, China
xingfudeshi@gmail.com
[*]*Corresponding author*

Keywords: Microservices Architecture, Medical Software, Distributed Transaction Solutions, Transaction Consistency, Medical Data Security

Abstract: This paper explores the application of microservices architecture in the development of medical software, particularly the importance of distributed transaction management in ensuring system consistency and data security. By analyzing the advantages of microservices architecture, including decentralized construction, technological flexibility, scalability, fault isolation, and continuous integration and delivery, the article delves into the challenges of distributed transactions in handling medical data, such as data consistency, transaction coordination and management, and performance and scalability issues. Subsequently, this paper introduces several common distributed transaction solutions, such as two-phase commit, three-phase commit, compensatory transactions, and the Saga transaction model, and demonstrates the application effects of these solutions in different medical scenarios through the analysis of actual cases in medical systems. This paper aims to provide a reference for developers of medical software, helping them effectively address the challenges of transaction management when implementing complex distributed systems, and ensuring system stability and data security.

1. Introduction

In today's digital age, the healthcare industry is at the forefront of a technological revolution. The acceleration of digitalization has led to an increasing complexity in medical software systems. These systems are tasked with the critical responsibility of handling vast amounts of patient data, detailed medical records, and extensive treatment information, while ensuring the privacy protection, accuracy, and real-time updates of this information. Microservices architecture, as an innovative software development paradigm, offers new perspectives and a variety of strategies to address these challenges. However, in the process of service decoupling and the adoption of distributed architectures, the traditional centralized architecture's transaction management methods are no longer fully effective. The complexity of distributed transaction management has gradually emerged as a key obstacle to optimizing system performance and enhancing stability. This challenge necessitates the exploration of more advanced methods to ensure the efficient operation and reliability of the system, in order to meet the growing technological demands of the healthcare industry. This paper will delve into the practical application of microservices architecture in the field of medical software, as well as the challenges and potential solutions of distributed transaction management, aiming to provide valuable references and suggestions for the healthcare industry's informatization process.

2. Advantages of Microservices Architecture in the Field of Medical Software Development

Microservices architecture is a design approach that breaks down a software application into a series of small, independent service units. These services operate in their own processes, typically built according to specific business needs. They exchange data and functionality through simple communication protocols, such as RESTful APIs. This design approach endows medical software with greater flexibility, enabling it to quickly adapt to market changes and achieve rapid iteration and updates. At the same time, it also supports a variety of technology stacks to meet the specific needs of different medical business scenarios. A general microservices architecture is shown in Figure 1.[1]

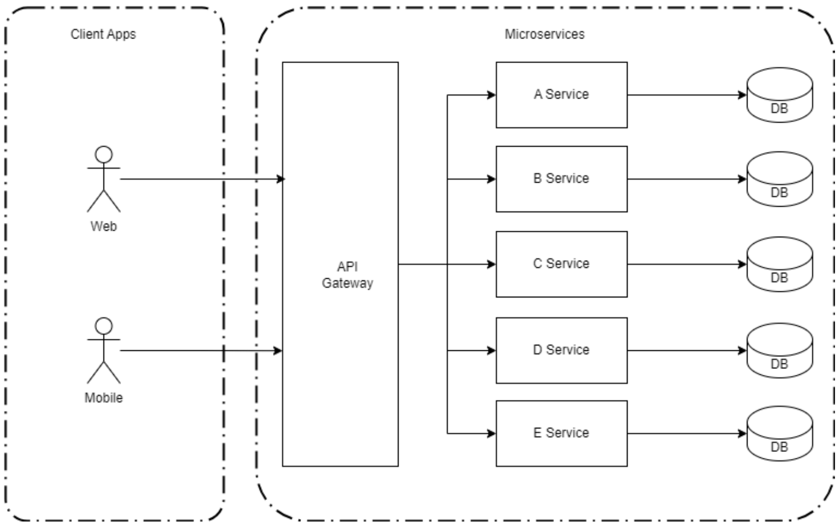

Figure 1: Microservices Architecture.

The advantages of microservices architecture are mainly manifested in the following key points:

2.1. Decentralized Construction

Medical software using microservices architecture can achieve independent development and deployment of various functional components, which not only enhances the efficiency of development work but also improves the convenience of system maintenance.[2]

2.2. Technological Flexibility

Microservices architecture allows each service to choose the most appropriate technology stack based on its specific business needs, providing a broad range of technological options for medical software development teams.

2.3. Scalability

Faced with the growing data scale and service demands in the medical field, microservices architecture possesses the capability for horizontal scaling, which can adapt to the increase in user base and data volume, ensuring the system's continuous development and expansion.

2.4. Fault Isolation

Thanks to the independent operation of services, a fault in a single service will not affect the entire application system, thereby enhancing the stability and reliability of the overall system.

2.5. Continuous Integration and Continuous Delivery (CI/CD)

The microservices architecture promotes the implementation of CI/CD practices, allowing medical software to quickly adapt to policy changes and market dynamics, achieving agile iterative updates.

3. Challenges of Distributed Transactions in Microservices Architecture

While microservices architecture brings many benefits, it also introduces new challenges, especially in the management of distributed transactions. Traditional transaction management mechanisms face many limitations in the context of microservices architecture, such as how to ensure transaction consistency across services and how to handle data synchronization between services, which have become difficult issues in the development of medical software.

3.1. Data Consistency Issues

Medical software has extremely high requirements for data consistency because the accuracy of the data is directly related to the quality of diagnosis and treatment and patient safety. Under the microservices architecture, we need to pay particular attention to solutions for data consistency to ensure the continuity and reliability of medical services. This may include adopting more advanced distributed database technologies, implementing more refined synchronization mechanisms between services, or designing customized transaction management strategies to meet the medical industry's special demands for data precision and real-time performance. At the same time, considering the sensitivity of medical data, any consistency solution must strictly comply with data protection and privacy regulations to ensure the security of patient information.[3]

3.2. Transaction Coordination and Management

Transaction coordination and management face significant challenges in microservices architecture. Traditional transaction management relies on a centralized coordinator, such as the two-phase commit protocol, but in microservices architecture, the independence and distributed nature of services make this centralized coordination less flexible and scalable. Moreover, microservices architecture advocates a decentralized management philosophy, which means that each service may need to handle transactions independently, requiring smarter transaction coordination strategies, such as event-based coordination or the Saga pattern. Transaction management and coordination strategies must adapt to this decentralized architecture to ensure effective operation without sacrificing system stability and efficiency.[4]

3.3. Performance and Scalability

In the medical software field, due to the involvement of a large amount of critical diagnostic and treatment data and the high demand for real-time business processes, the performance and scalability of distributed transactions are particularly important. This requires us to consider not only the consistency and reliability of transactions when designing microservices architecture but also to fully assess the impact of network communication on performance and take corresponding optimization measures. For example, communication latency can be reduced through service discovery and load balancing mechanisms, or the system's throughput can be improved by adopting asynchronous processing and message queueing techniques. At the same time, the system's monitorability and fault recovery capability must also be considered to ensure that problems can be quickly located and resolved in a distributed environment, safeguarding the continuity and stability of medical services. Moreover, as the number of services increases, the complexity of maintaining transaction logs, coordinating transaction states, and handling fault recovery also increases. This requires the design of efficient transaction mechanisms that reduce reliance on network communication, optimize resource usage, and ensure that system scalability is not limited by transaction management.[5]

4. Distributed Transaction Solutions

4.1. Two-Phase Commit (2PC)

Two-Phase Commit (2PC) is a traditional mechanism for handling distributed transactions, designed to ensure that all participating nodes either commit the transaction in full or roll it back entirely. It consists of two phases: in the preparation phase, the coordinator asks each participant if they are ready to commit the transaction; if all participants agree, the commit phase begins, and the coordinator notifies them to formally commit the transaction. Although 2PC can guarantee the atomicity and consistency of transactions, it also faces challenges such as single-point failures and performance bottlenecks.[6]

4.2. Three-Phase Commit (3PC)

Three-Phase Commit (3PC) is an improvement on the Two-Phase Commit (2PC) protocol, introducing a timeout mechanism and an additional preparation phase to address the blocking issues in 2PC. 3PC reduces the blocking time for participants through three steps: pre-prepare, prepare, and commit, thereby enhancing the system's responsiveness. However, 3PC adds an extra round of network communication, which may impact performance.[7]

4.3. Try-Confirm-Cancel Transactions (TCC)

The TCC transaction model maintains transaction consistency through three steps. Initially, in the Try phase, the system performs preliminary checks and reserves necessary resources; then, in the Confirm phase, the system confirms and completes the actual operation of the transaction; if problems arise during the process, in the Cancel phase, the system rolls back the previous operations. The TCC model requires business logic to support a compensation mechanism, making it more suitable for scenarios with complex business processes that require strict consistency. For example, Apache Seata has implemented the TCC transaction model.[8]

4.4. Local Message Table

The local message table is a scheme that uses a message queue to achieve eventual consistency. Each service maintains a local message table, and transaction operations are first completed in the local database, then a message is generated and stored in the local message table. Subsequently, the message is sent to the message queue asynchronously, and message consumers are responsible for taking the message from the queue and processing it to ensure transaction consistency. This method can effectively avoid the complexity in distributed transactions while ensuring system consistency.

4.5. Message Transactions

Message transactions are a solution for achieving distributed transaction consistency through reliable message delivery. It is typically used in conjunction with message middleware (such as Kafka, RabbitMQ), ensuring transaction consistency through the mechanisms of message sending and acknowledgment. Unlike the local message table approach, message transactions do not rely on each service maintaining a local message table but instead directly utilize the message middleware to manage the reliable delivery of transactions. This approach can reduce direct database operations, thereby improving the system's throughput and scalability.

4.6. Best-Effort Notification

Best-Effort Notification is a flexible transaction processing approach that ultimately achieves transaction consistency by retrying notifications to related services multiple times. This method is suitable for scenarios where the requirement for data consistency is not very high. It enhances the system's availability and response speed by relaxing the strong consistency requirements of transactions. For example, Alibaba's RocketMQ provides support for transactional messages, handling the commit and rollback of transactions through best-effort notifications.

4.7. Saga Transaction Model

The Saga transaction model is a solution for handling long-running transactions. It breaks down a long transaction into multiple local transactions, each with a corresponding compensatory action. If a step in the transaction chain fails, the Saga model will roll back the previously successful operations by performing compensatory actions to undo these changes. The Saga model is particularly suitable for scenarios with long business processes that require a high degree of flexibility. Currently, Apache Seata has implemented the Saga transaction model.[9]

5. Case Analysis within Medical Systems

5.1. Electronic Medical Record System (EMR)

In a well-known large general hospital, its Electronic Medical Record System (EMR) plays a central role and is closely integrated with several key systems such as the Laboratory Information System (LIS), Picture Archiving and Communication System (PACS), and Hospital Information System (HIS). The system uses the Two-Phase Commit (2PC) protocol to ensure the synchronized updates and consistency of patient information across platforms, thereby ensuring the continuity and accuracy of medical services.

During the patient's visit, once the doctor updates the medical record information in the EMR, the system automatically triggers interactions with LIS and PACS to synchronize examination and imaging data. The 2PC protocol ensures the atomic submission of data in the background, avoiding medical risks caused by inconsistencies between systems.

5.2. Telemedicine Service Platform

A telemedicine service platform serving multiple regions connects experts with patients through digital means, providing real-time consultations, diagnoses, and treatment recommendations. The platform adopts the Three-Phase Commit (3PC) protocol to handle the complex transactions that may arise during the consultation process.

After the expert completes the remote consultation, the 3PC protocol ensures the immediate update of the patient's medical records, the accurate preservation of consultation records, and the synchronized distribution of remote diagnostic recommendations, thereby enhancing the quality and efficiency of medical services.

5.3. Hospital Pharmaceutical Supply Chain Management System

The hospital pharmaceutical supply chain management system is responsible for the procurement, inventory management, and sales tracking of drugs, with transaction management in this process requiring a very high degree of accuracy. The system adopts a local message table and message transaction mechanism to ensure the consistency and real-time nature of drug data.

When drugs are sold, the system records transaction information in the local message table and asynchronously updates inventory and triggers supply chain responses through a message queue. This mechanism enhances the flexibility of data processing and the system's response speed.[10]

5.4. Medical Equipment Monitoring System

In the Intensive Care Unit (ICU), the medical equipment monitoring system is responsible for real-time monitoring of patients' vital signs, with extremely high requirements for the timeliness and accuracy of data. The system adopts a best-effort notification mechanism to ensure that critical data can be transmitted to medical staff in a timely manner.

When there are changes in the patient's vital sign data, the system uses the best-effort notification mechanism to attempt multiple data transmissions to the monitoring platform under different network conditions, ensuring that medical staff can obtain the most up-to-date patient information and respond quickly.

5.5. Integrated Diagnostic and Treatment Services

A multidisciplinary integrated diagnostic and treatment service case involves experts from various departments such as internal medicine, surgery, and radiology, who work together to formulate a comprehensive treatment plan for patients. This service adopts the Saga transaction model to coordinate business processes between different departments.

The Saga transaction model ensures the consistency and traceability of the diagnostic and treatment process by defining a series of local transactions and corresponding compensatory actions. Even if some operations fail at certain stages, Saga can still guarantee the continuity of the entire diagnostic and treatment service and the integrity of the data.

5.6. Medical Consultation Appointment System

The medical consultation appointment system aims to provide a precise and reliable appointment management platform for doctors and patients. The system must ensure the accurate transmission of appointment information to avoid conflicts or delays due to notification failures.

After a successful appointment, the system utilizes a best-effort notification mechanism to attempt multiple notifications to doctors and patients through various communication channels. Additionally, the system's asynchronous processing mechanism ensures stability and availability under high concurrency conditions.

6. Conclusion

This paper has systematically analyzed the application of microservices architecture in the development of medical software, as well as the important role of distributed transaction management in addressing the complexity and consistency challenges of medical systems. In the process of discussing different distributed transaction solutions, the advantages and disadvantages of each solution and their applicable scenarios have been elaborated in detail with practical cases. Although microservices architecture brings flexibility and scalability, the management of distributed transactions remains a complex issue that requires careful consideration in system design. In the future, with the increasing demand for digitalization in the medical industry and the further development of distributed system technology, new transaction management schemes may be proposed to better meet the special needs of the medical industry. It is hoped that this paper can provide valuable insights and guidance for medical software developers in designing and implementing distributed systems, thereby promoting the continuous improvement of the reliability and security of medical software systems.

References

[1] Li, L., & Shu, C. (2024). Software development practice with microservices architecture and containerization technology. Internet of Things Technology, 14(05), 64-67. https://doi.org/10.16667/j.issn.2095-1302.2024.05.017

[2] Sun, L., Yan, C. M., Ma, Y. H., et al. (2023). Research on Internet hospital architecture based on distributed microservices business middle platform. Journal of Medical Informatics, 44(10), 74-80.

[3] Chen, X. L. (2024). Data consistency assurance mechanisms in distributed database systems in the cloud computing environment. Information and Computer (Theory Edition), 36(08), 137-139.

[4] Yang, H. B. (2024). Research on data consistency and transaction management strategies in large-scale distributed systems. Information and Computer (Theory Edition), 36(09), 68-71.

[5] Huang, X. Y., Zhou, X. P., & Wang, Z. M. (2023). Design and application of smart hospital infrastructure based on microservices. Journal of Medical Informatics, 44(01), 59-64.

[6] Fan, P., Liu, J., Yin, W., et al. (2020). 2PC: A distributed transaction concurrency control protocol of multi-microservice based on cloud computing platform. Journal of Cloud Computing: Advances, Systems and Applications, 9(1), 96-100.*

[7] Jiang, M. P., Li, R. K., Wang, J. Q., et al. (2023). Data cross-network exchange technology based on distributed transactions. Command Information System and Technology, 14(02), 91-94+100. https://doi.org/10.15908/j. cnki.cist. 2023.02.015

[8] Liu, J. (2021). Research and implementation of distributed transactions in microservices architecture. https://doi.org/10.27429/d.cnki.gxjdu.2021.000424

[9] Munonye, K., & Martinek, P. (2020). Enhancing performance of distributed transactions in microservices via buffered serialization. Journal of Web Engineering, 19(5), 647-684.

[10] Wang, Y. P., Cai, L., Jiang, Y., et al. (2022). Application of competence-oriented evaluation system in multidisciplinary discussion and treatment of difficult cases and case presentations. Chinese Medical Science, 12(22), 63-67.