西安建筑科技大学

本科毕业设计（论文）开题报告

学　　院　理学院

专业班级　数据科学与大数据 2101

题目名称　非结构化数据在实时通信系统中的存储设计

学生姓名　郭靖远

学　　号　2111030210

指导教师　鲁萍

年　月　日

一、课题目的和意义

1. 目的：

随着互联网的发展，实时通信系统（如即时通讯、在线会议、协同编辑）已成为重要的基础设施。这类系统需要处理大量非结构化数据，包括文本、图片、音频、视频等。相比于结构化数据，非结构化数据存储面临以下问题：

存储方式复杂：关系型数据库不适合存储大规模非结构化数据，传统方案往往需要引入额外的存储系统。

数据访问效率要求高：实时通信要求低延迟数据读写，存储系统需要高效支持查询和检索。

一致性与持久化问题：缓存与数据库结合使用时，如何保证数据一致性和高可用性?

本课题的目标是设计和实现适用于实时通信系统的非结构化数据存储方案，使其能够高效支持 WebSocket 通信，保证数据的持久化和快速检索，提高系统的实时性和稳定性

2. 意义：

本课题的研究不仅具有理论价值，也具备工程实践价值，具体体现在以下几个方面：

提升实时通信系统的数据管理能力：设计合适的存储方案，可以提升系统对非结构化数据的处理能力，优化聊天记录、语音、图片、视频等数据的存取效率。

提高存储系统的扩展性和性能：结合 MongoDB（NoSQL 数据库）与 Redis（缓存数据库），优化数据读写与查询，适应海量数据存储需求，降低服务器压力。

保障数据一致性与可靠性：采用事务管理、异步写入、分布式存储等方案，提高系统数据的可靠性，减少数据丢失风险。

为后续研究提供参考：本课题研究的存储方案可适用于其他场景，如社交媒体、音视频平台、协作软件等，为相关领域的研究和工程实践提供借鉴。

通过本研究，可以为实时通信系统提供一套高效、可扩展、低延迟的非结构化数据存储方案，提高系统的稳定性和用户体验。

二、课题关键问题及难点

非结构化数据的存储方式选择：是本课题的首要挑战。实时通信系统需要存储多种非结构化数据，如文本、图片、音频和视频等，传统关系型数据库难以满足高效存储需求。因此，需要在 MongoDB、GridFS、对象存储（如 MinIO） 等方案之间进行权衡，以确保数据存取的高效性、扩展性和低成本。如何选择合适的存储方式，使系统既能快速响应，又能节省存储资源，是一个关键问题。

数据存储结构设计：直接影响数据的查询效率和系统性能。MongoDB 采用文档存储模型，支持嵌套存储和索引优化，但面对海量聊天记录，如何合理建立索引、划分数据分片，以及管理历史数据的归档和清理，都是需要解决的问题。此外，数据的持久化策略也需要考虑，如何高效地从 Redis 持久化到 MongoDB，并在高并发环境下保障数据一致性，是本研究的重点。

数据一致性与持久化策略：是保障数据可靠性的关键。由于系统采用 MongoDB + Redis 组合架构，消息在缓存和数据库之间需要进行同步，而 Redis 主要用于加速查询，但其数据易失性使得持久化机制的选择（如 AOF 日志、定期快照）变得尤为重要。此外，在高并发场景下，MongoDB 如何保证事务一致性，以及如何在系统异常情况下恢复未同步数据，都是需要重点研究的问题。

WebSocket 消息存储优化：直接影响系统的实时性和用户体验。WebSocket 作为双向、长连接通信方式，要求系统能快速处理消息的存储和转发，同时保证消息的顺序性和可靠性。此外，用户离线时的未读消息存储策略也是一个关键点，如何在用户重新上线时高效推送未读消息，同时避免数据丢失或重复发送，需要合理的设计，如使用 Redis 作为短期存储，定期同步至 MongoDB，或结合消息队列 (Kafka/RabbitMQ) 优化消息处理。

高并发下的性能优化：是提升系统可用性的重要课题。在实时通信环境下，大量用户同时在线会带来高频数据读写，如果数据库查询或写入效率低下，可能导致系统延迟甚至崩溃。因此，需要采用 MongoDB 的分片 (Sharding) 技术提升数据查询性能，利用 Redis 进行热点数据缓存，以及引入消息队列（Kafka/RabbitMQ） 进行异步存储，以降低数据库压力，提高系统吞吐量。

数据安全与权限控制：关乎用户隐私和系统安全性。聊天记录、文件等非结构化数据可能包含敏感信息，需要严格的权限管理和数据加密机制。如何确保 MongoDB 和 Redis 存储的数据不被非法访问？如何在 WebSocket 传输过程中防止中间人攻击（MITM）？这些问题需要通过 RBAC（基于角色的访问控制）、数据加密存储（AES/RSA）、WebSocket TLS 传输加密 等安全措施来保障系统的数据安全性和用户隐私。

三、文献综述

[1] 张勇, 李娜, 刘杰. 基于 WebSocket 的分布式实时通信系统设计与实现[J]. 计算机科学, 2023, 42(3): 102-112.

[2] 李华, 王伟. 面向多终端的即时通信应用断线重连机制研究[J]. 软件工程, 2022, 35(1): 55-63.

[3] 王强, 赵鑫, 孙涛. 高并发环境下的即时通信系统性能优化研究[J]. 计算机工程与设计, 2021, 42(8): 1999-2005.

[4] 陈磊, 杨飞, 王佳. 基于微服务架构的即时通信平台设计与实现[J]. 通信技术, 2022, 55(4): 418-426.

[5] 赵鹏, 张琳. WebSocket 协议在移动即时通信系统中的应用与优化[J]. 现代计算机, 2021, 38(10): 65-69.

[6] 赵鹏, 张琳. WebSocket 协议在移动即时通信系统中的应用与优化[J]. 现代计算机, 2021, 38(10): 65-69.

[7] 杨帆, 李鹏. Go 语言在高并发实时通信系统中的应用与优化[J]. 软件技术, 2023, 41(5): 112-119.

[8] Smith, J., Brown, A. Real-time communication architecture with WebSocket and Node.js[J]. International Journal of Web Applications, 2021, 11(2): 187-195.

[9] Tan, C., Ren, Y., Wang, C. Adaptive signaling for real-time multimedia communication with WebRTC and WebSocket[J]. Applied Intelligence, 2023, 53(1): 804-812.

[10] Sinoara, R. A., Camacho-Collados, J., Rossi, R. G., et al. Cross-lingual and multimodal real-time communication framework[J]. Knowledge-based Systems, 2019, 163(Jan.1): 955-971.

[11] Carter, M., Thompson, G. High concurrency handling in WebSocket communication systems using Elixir[J]. IEEE Transactions on Network and Service Management, 2022, 19(2): 250-265.

[12] Chen, C., Peterson, J. WebSocket-based real-time messaging in containerized environments[J]. Journal of Cloud Computing, 2021, 10(3): 45-58.

四、主要研究内容、研究方法或设计方案

本研究的核心目标是设计和实现适用于实时通信系统的非结构化数据存储方案，确保系统能够高效存储、管理和检索非结构化数据，同时保障数据一致性、安全性，并优化系统性能。研究内容主要围绕存储架构设计、数据一致性、性能优化和安全策略展开，并结合 MongoDB、Redis、WebSocket 以及消息队列等技术进行实现。

1. 主要研究内容

非结构化数据的存储架构设计：研究 MongoDB、GridFS、对象存储（MinIO）等存储方案，分析它们的适用场景，并选择最佳存储方式。设计适用于实时通信的非结构化数据存储模型，包括聊天记录、图片、音频、视频等数据的存储方式。结合索引优化、数据分片（Sharding） 以及生命周期管理，提高查询性能并降低存储成本。

WebSocket 消息存储及数据一致性策略：研究 WebSocket 通信机制，优化消息的存储与转发，提高系统的实时性和稳定性。设计 Redis + MongoDB 组合存储策略，使用 Redis 进行短期缓存，定期持久化到 MongoDB，以提高消息存储效率。研究数据同步和一致性问题，确保 MongoDB 和 Redis 之间的数据不会丢失或冲突。

高并发环境下的数据存取优化：研究 MongoDB 在高并发环境下的读写性能，使用索引优化、分片技术 和查询优化提升数据访问效率。结合 Redis 缓存策略，优化聊天记录、未读消息等数据的存取效率。研究消息队列（Kafka/RabbitMQ），实现异步消息存储，降低数据库写入压力，提高系统吞吐量。

离线消息存储与推送：研究离线消息存储机制，确保用户在离线时的消息能够可靠存储，并在上线时高效推送。设计未读消息管理策略，保证用户可以精准接收未读消息，而不会产生消息丢失或重复推送的问题。

数据安全与权限管理：研究 RBAC（基于角色的访问控制） 或 ABAC（基于属性的访问控制），确保不同用户/群组的聊天记录和文件访问受控。采用 AES、RSA 加密技术，保障 MongoDB 和 Redis 存储的数据安全。研究 WebSocket TLS 加密通信，防止消息在传输过程中被窃取或篡改。

2. 研究方法

本研究采用理论研究 + 实验验证相结合的方法，通过文献分析、方案设计、系

统开发和实验测试等步骤，逐步优化非结构化数据在实时通信系统中的存储方案。首先，进行国内外文献调研，研究实时通信存储架构、MongoDB 存储优化、Redis 缓存策略、WebSocket 消息存储等方面的相关研究，总结现有方案的优缺点，并结合高并发环境的需求，确定最优的存储设计方案。

在方案设计方面，本研究采用 MongoDB + Redis + WebSocket 作为存储架构，确保数据的高效存储和快速访问。Redis 作为短期缓存，存储最近消息并加速查询，MongoDB 负责长期存储完整聊天记录，并定期从 Redis 持久化数据。对于文件、图片、音视频等非结构化数据，小文件存入 MongoDB GridFS，大文件存入 MinIO 对象存储，从而提高存储效率并减少数据库压力。此外，为确保数据一致性，研究 Redis + MongoDB 事件监听机制，保证消息先存入 Redis 后同步到 MongoDB，同时结合 AOF + RDB 持久化策略，防止 Redis 数据丢失，并利用 MongoDB 事务机制保障多文档更新的一致性。

在实验与测试方面，首先进行存储测试，模拟海量聊天数据存储，测试 MongoDB 和 Redis 的查询、写入性能；其次，开展高并发测试，使用 Apache JMeter 或 wrk 等工具，测试系统在高并发环境下的稳定性；此外，还进行消息一致性测试，模拟网络延迟、服务器宕机等情况，验证 Redis 和 MongoDB 之间的数据一致性；最后，进行安全性测试，包括权限控制和加密存储，以确保数据不会被未授权访问或篡改。

为了优化高并发环境下的存储效率，本研究采用 MongoDB Sharding（分片）机制，根据用户 ID 或时间维度划分数据，提高查询效率；结合 Redis 过期策略（LRU 淘汰），管理缓存数据，减少数据库负载；并引入 Kafka 消息队列，将高并发写入操作进行异步处理，降低 MongoDB 的写入压力。此外，在安全性方面，本研究采用 RBAC（基于角色的访问控制）机制，确保用户只能访问自己的聊天记录，并通过 AES 加密存储聊天记录、RSA 加密存储文件，保障数据隐私。同时，WebSocket 传输采用 TLS 加密，防止通信被监听或篡改，从而提升系统的安全性和可靠性。

五、工作的主要阶段、进度和完成时间

1. 主要阶段：

搭建基本的前后端框架、实现 WebSocket 通信模块、测试性能、性能优化

2. 进度：

第 4-6 周：查阅文献资料，完成系统需求分析和方案设计，搭建基本的前后端框架。

第 6-7 周：实现 WebSocket 通信模块，包括连接管理、双向通信及断线重连机制。

第 7-9 周：实现用户状态管理功能，并将状态数据存储在 Redis 中；完成消息持久化存储模块，实现数据同步。

第 10-12 周：优化前端页面加载速度和后端的高并发处理能力，确保系统在大规模用户并发下的稳定性。

第 13-14 周：进行系统的负载测试和高并发性能测试，收集各项性能数据，撰写论文初稿。

第 15-16 周：修订最终论文，制作答辩 ppt，准备答辩

3. 完成时间：

预计在 12 周内完成整个课题的研究工作，并提交最终的研究报告和论文。

六、已进行的前期准备工作

1. 技术调研与文献阅读

在课题开始前，我深入阅读了关于 WebSocket 协议、Go 语言高并发处理、Next.js 框架等相关文献和技术资料，了解了实时通信系统的基本原理和最新发展趋势，尤其是如何优化 WebSocket 连接、消息持久化与同步等问题。这为后续项目开发奠定了理论基础，并帮助明确了技术选型。

2. 技术栈选择与工具熟悉

根据课题需求，我选择了 WebSocket 作为前后端通信协议，Go 语言作为后端开发语言，Next.js 作为前端框架，并准备使用 Redis 进行状态管理。这些技术栈已在多个类似项目中得到验证，具备高效的并发处理能力和稳定的性能支持。我已进行了一些简单的实验，验证了这些技术栈在高并发环境下的可行性，并通过小范围的测试掌握了它们的基本用法。

3. 开发环境搭建

我已经在本地和云服务器上搭建了开发环境，安装了 Go 语言、Node.js、Next.js、Redis 等所需工具，并成功部署了一个简单的 WebSocket 服务端和前端页面，进行基础的通信测试。此外，我还配置了数据库（如 MongoDB 或 MySQL）用于消息存储，以便后续对消息持久化和查询性能进行优化。

4. 项目框架设计与需求分析

根据课题目标和技术选型，我已完成了系统需求分析，并根据功能需求初步设计了系统架构图，明确了前端、后端和数据库之间的交互方式。项目框架设计包括用户注册与登录、消息发送与接收、用户状态管理等功能模块，且考虑到系统的高可用性和可扩展性，已经设计了分布式架构和容错机制。

5. 性能测试与优化初步探索

在前期的技术调研过程中，我进行了一些性能测试，评估了 WebSocket 连接的稳定性和 Go 语言的并发处理能力。通过对不同负载场景的测试，初步验证了系统的可行性，并根据测试结果调整了系统设计，提出了初步的性能优化方案。

6. 相关文献及案例研究

我参考了大量国内外关于实时通信系统、WebSocket 协议优化及 Go 语言高并发处理的文献和案例，学习了其他开发团队在类似项目中的经验和解决方案，并对比分析了不同技术方案的优缺点。这些参考资料为我后续的设计与实现提供了重要的理论支持和实践指导。

七、指导教师审阅意见

指导教师签名：　　年　　月　　日

八、教研室（系）审阅意见

教研室（系）主任签名：　　年　　月　　日

九、学院审核意见

主管教学院长签名：　　年　　月　　日

说明：1.本报告须由承担毕业设计（论文）任务的学生按照《毕业设计（论文）任务书》相关要求，在做毕业设计（论文）开始后的前 3 周之内独立撰写完成，并交指导教师审阅。

2.每个毕业设计（论文）课题撰写本报告一份，作为指导教师审查学生能否承担该毕业设计(论文)课题任务的依据，并接受学校抽查。