HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA ĐÀO TẠO SAU ĐẠI HỌC

BÁO CÁO

CÁC HỆ THỐNG PHÂN TÁN

ĐỀ TÀI

HỆ THỐNG LƯU TRỮ KEY-VALUE PHÂN TÁN

Giảng viên hướng dẫn: TS. Kim Ngọc Bách

Lớp: M24CQHT02-B

Nhóm: 14

Danh sách học viên:
B24CHHT076 - Nguyễn Thanh Hùng
B24CHHT078 - Quản Trường Huy
B24CHHT092 - Tô Thanh Thái

Hà Nội – 2025

Nội dung phân chia công việc

Mã học viên	Họ và tên	Nội dung công việc	Phần trăm đóng góp
B24CHHT076	Nguyễn Thanh Hùng	Tìm hiểu replication và quorum, tích hợp vào hệ thống. Thiết kế và chạy các kịch bản thử nghiệm.	40%
B24CHHT078	Quản Trường Huy	Tìm hiểu vòng băm nhất quán, tích hợp vào hệ thống.	30%
B24CHHT092	Tô Thanh Thái	Tìm hiểu gossip protocol, tích hợp vào hệ thống.	30%

Mục lục

Chương 1. Giới thiệu

1.1 Mục tiêu

Mục tiêu của bài tập lớn này là xây dựng một hệ thống lưu trữ dữ liệu dạng key-value phân tán, có khả năng mở rộng linh hoạt và đảm bảo tính sẵn sàng cao trong môi trường mạng phân tán. Hệ thống sẽ sử dụng kỹ thuật consistent hashing để phân phối dữ liệu đều giữa các node, đồng thời áp dụng cơ chế replication và quorum để tăng cường độ tin cậy và khả năng chịu lỗi.

Ngoài ra, hệ thống sẽ triển khai giao thức gossip để cập nhật thông tin thành viên trong cluster, giúp các node đồng bộ trạng thái một cách phi tập trung, từ đó hỗ trợ việc cân bằng dữ liệu khi có node mới tham gia hoặc rời khỏi cluster. Một phần quan trọng khác là cơ chế migrate dữ liệu, đảm bảo dữ liệu được di chuyển an toàn và hiệu quả giữa các node khi vòng băm thay đổi.

Thông qua bài tập lớn này, mục tiêu còn là thực hành và hiểu sâu các khái niệm cơ bản trong hệ thống phân tán, đồng thời rèn luyện kỹ năng lập trình thực tế để triển khai một dịch vụ backend có khả năng chịu lỗi và mở rộng cao.

1.2 Phạm vi và giới hạn

Bài tập này tập trung vào việc xây dựng một hệ thống key-value store phân tán với các tính năng cơ bản nhất để đảm bảo phân phối dữ liệu hiệu quả và khả năng chịu lỗi trong môi trường mạng phân tán. Hệ thống sẽ triển khai consistent hashing để phân phối dữ liệu giữa các node, cùng với cơ chế replication và quorum nhằm tăng tính sẵn sàng và độ tin cậy.

Tuy nhiên, bài tập không đi sâu vào các vấn đề phức tạp hơn như tối ưu hiệu năng nâng cao, cân bằng tải động dựa trên tải thực tế của node, hay các cơ chế bảo mật và mã hóa dữ liệu. Ngoài ra, các tính năng như phân mảnh dữ liệu (sharding) phức tạp, xử lý xung đột đa phiên bản (multi-version concurrency control) hay khả năng tự động mở rộng quy mô (auto-scaling) cũng chưa được triển khai trong phạm vi này.

Việc phát hiện và xử lý node chết dựa trên cơ chế timeout và gossip đơn giản, chưa áp dụng các giao thức phức tạp như SWIM hay các kỹ thuật nâng cao khác. Mục tiêu chính là xây dựng một hệ thống prototype để hiểu và thực hành các khái niệm cơ bản trong hệ thống phân tán, làm nền tảng cho các nghiên cứu hoặc phát triển tiếp theo.

Chương 2. Các lý thuyết và công nghệ áp dụng

2.1 Mô hình phân tán

Trong các hệ thống lưu trữ hiện đại, đặc biệt là những hệ thống xử lý lượng lớn dữ liệu với yêu cầu cao về hiệu năng và tính sẵn sàng, việc sử dụng mô hình phân tán là một xu hướng tất yếu. Mô hình phân tán cho phép dữ liệu và tải xử lý được phân phối trên nhiều máy chủ (node) độc lập, giúp tăng khả năng mở rộng, chịu lỗi và tránh điểm nghẽn tập trung.

Trong mô hình này, mỗi node trong mạng lưới lưu trữ một phần dữ liệu và tham gia vào việc xử lý các yêu cầu đọc, ghi. Việc phân chia dữ liệu và đồng bộ trạng thái giữa các node là những bài toán then chốt cần giải quyết để đảm bảo hệ thống hoạt động hiệu quả và nhất quán.

Các đặc điểm chính của mô hình phân tán bao gồm:

- Tính mở rộng (Scalability): Hệ thống có thể tăng thêm node để xử lý tải lớn hơn mà không làm giảm hiệu năng.
- Tính chịu lỗi (Fault tolerance): Khi một hoặc nhiều node bị lỗi, hệ thống vẫn đảm bảo hoạt động và dữ liệu không bị mất.
- Tính nhất quán (Consistency): Dữ liệu phải được đồng bộ và nhất quán trên các node theo các chính sách nhất định.
- Tính sẵn sàng cao (High availability): Hệ thống luôn sẵn sàng phục vụ yêu cầu người dùng dù có sự cố xảy ra.

2.2 Vòng băm nhất quán (Consistent Hashing)

Một trong những thách thức lớn nhất của hệ thống phân tán là phân phối dữ liệu một cách hiệu quả giữa các node, đồng thời giảm thiểu lượng dữ liệu phải di chuyển khi cấu hình hệ thống thay đổi (ví dụ thêm hoặc bớt node). Consistent hashing là một kỹ thuật được thiết kế để giải quyết bài toán này.

- Nguyên lý hoạt động
  - Consistent hashing biến không gian băm thành một vòng tròn ảo (hash ring). Các node trong hệ thống và các key đều được ánh xạ vào vị trí trên vòng tròn dựa trên giá trị băm của chúng. Cụ thể:
  - Mỗi node sẽ được gán một hoặc nhiều vị trí trên vòng băm (thường gọi là virtual nodes hoặc replicas), giúp cân bằng tải và tránh tình trạng tập trung dữ liệu không đồng đều.
  - Khi một key cần lưu trữ, hệ thống sẽ tính giá trị băm của key và xác định vị trí của nó trên vòng băm. Key này sẽ được lưu tại node đầu tiên có vị trí lớn hơn hoặc bằng vị trí của key (di chuyển theo chiều kim đồng hồ trên vòng tròn).

    - Nếu vị trí băm của key lớn hơn tất cả các node trên vòng, key sẽ được lưu tại node đầu tiên trên vòng (vòng tròn là không gian tuần hoàn).
- Ưu điểm của consistent hashing
    - Giảm thiểu dữ liệu di chuyển: Khi một node mới được thêm vào hoặc xóa khỏi hệ thống, chỉ một phần nhỏ các key (tương ứng với phạm vi của node đó trên vòng băm) cần được di chuyển đến node mới hoặc node kế cận, thay vì phải di chuyển toàn bộ dữ liệu. Điều này giảm thiểu thời gian downtime và tối ưu hóa hiệu suất hệ thống.
    - Cân bằng tải: Việc sử dụng virtual nodes cho phép phân phối dữ liệu đều hơn giữa các node, tránh tình trạng node quá tải hoặc quá nhàn rỗi.
    - Khả năng mở rộng linh hoạt: Hệ thống có thể thêm hoặc bớt node dễ dàng mà không gây ảnh hưởng lớn đến hoạt động bình thường.
- Một số hạn chế
    - Độ phức tạp tăng khi số node lớn: Việc quản lý nhiều virtual nodes và cập nhật vòng băm có thể trở nên phức tạp hơn khi hệ thống mở rộng rất lớn.
    - Không giải quyết vấn đề nhất quán dữ liệu hoàn toàn: Cần kết hợp với các cơ chế khác như replication và quorum để đảm bảo tính nhất quán và chịu lỗi.

2.3 Replication và cơ chế quorum

2.3.1 Replication (Nhân bản dữ liệu)

Trong hệ thống phân tán, việc lưu trữ dữ liệu chỉ trên một node duy nhất sẽ tạo ra điểm nghẽn và rủi ro mất dữ liệu khi node đó gặp sự cố. Do đó, cơ chế replication được áp dụng nhằm nhân bản dữ liệu trên nhiều node khác nhau trong cluster. Mỗi key-value sẽ được lưu trên một số node nhất định (gọi là mức độ nhân bản – replication factor), giúp tăng tính sẵn sàng và khả năng chịu lỗi của hệ thống.

Trong bài tập này, cơ chế replication được triển khai bằng cách lưu dữ liệu trên một số node được xác định dựa trên vòng băm nhất quán. Cụ thể, sau khi xác định node chính chịu trách nhiệm lưu trữ key (primary node), dữ liệu sẽ được sao chép tới một hoặc nhiều node kế tiếp trên vòng băm để làm replica.

Việc này giúp hệ thống có thể tiếp tục phục vụ các yêu cầu đọc, ghi ngay cả khi một số node gặp sự cố hoặc không khả dụng. Đồng thời, replication giúp tăng khả năng chịu lỗi và tránh mất mát dữ liệu.

2.3.2 Cơ chế quorum

Mặc dù replication giúp tăng độ tin cậy, việc đồng bộ dữ liệu giữa các bản sao và xử lý các tình huống xung đột dữ liệu trở thành thách thức lớn. Để giải quyết vấn đề này, hệ thống áp dụng cơ chế quorum trong các hoạt động đọc và ghi dữ liệu.

Cơ chế quorum quy định một ngưỡng tối thiểu các node cần tham gia để đọc hoặc ghi dữ liệu được xem là thành công. Các tham số chính thường bao gồm:

- N: Tổng số bản sao dữ liệu (replication factor).
- W: Số node phải ghi thành công để một thao tác ghi được chấp nhận.
- R: Số node phải phản hồi thành công để một thao tác đọc được chấp nhận.

Để đảm bảo tính nhất quán, hệ thống thường lựa chọn các giá trị sao cho R + W > N, nghĩa là tập các node tham gia đọc và ghi phải có sự giao thoa nhất định. Điều này giúp đảm bảo rằng dữ liệu đọc ra luôn bao gồm ít nhất một bản sao mới nhất.

Ví dụ: với N=3 (mỗi key lưu trên 3 node), có thể chọn W=2 và R=2. Khi ghi, ít nhất 2 node phải xác nhận thành công; khi đọc, hệ thống sẽ lấy dữ liệu từ ít nhất 2 node và so sánh để trả về bản cập nhật nhất.

Ưu điểm của cơ chế quorum

- Đảm bảo tính nhất quán "nhất định" (eventual consistency) với mức độ kiểm soát.
- Tăng tính sẵn sàng: Hệ thống vẫn có thể hoạt động khi một số node không khả dụng.
- Giảm thiểu xung đột dữ liệu thông qua việc kiểm tra phiên bản dữ liệu (versioning) và chọn bản cập nhật mới nhất.

Trong bài tập, hệ thống key-value store sử dụng replication với replication factor cố định (ví dụ 3). Các thao tác đọc và ghi được thực hiện theo cơ chế quorum đơn giản để cân bằng giữa độ trễ và độ nhất quán. Phiên bản dữ liệu (version) được gắn kèm với mỗi giá trị để hỗ trợ giải quyết xung đột khi có các phiên bản khác nhau của cùng một key.

Việc áp dụng replication và quorum giúp hệ thống có khả năng chịu lỗi cao hơn, dữ liệu ít bị mất khi node gặp sự cố, đồng thời duy trì hiệu suất truy xuất dữ liệu ổn định trong môi trường phân tán.

2.4 Gossip Protocol

Gossip Protocol, hay còn gọi là giao thức truyền tin kiểu “giao tiếp theo kiểu đàm thoại,” là một kỹ thuật phổ biến được sử dụng trong các hệ thống phân tán để đồng bộ trạng thái các node một cách hiệu quả và phi tập trung. Mục tiêu chính của gossip là giúp tất cả các node trong cluster có thông tin nhất quán về trạng thái hệ thống, như danh sách node hiện tại, trạng thái node (alive, suspect, dead), hoặc các thay đổi về vòng băm (hash ring).

Trong gossip, mỗi node định kỳ chọn một hoặc nhiều node khác ngẫu nhiên để trao đổi thông tin trạng thái. Qua nhiều vòng trao đổi như vậy, thông tin sẽ lan truyền nhanh chóng và đồng đều khắp cluster, tương tự như cách mà virus hoặc tin đồn lan truyền trong cộng đồng.

Cách thức hoạt động

- Chu kỳ gossip: Mỗi node có một khoảng thời gian cố định (gossip interval) để thực hiện một vòng trao đổi thông tin.
- Chọn peer: Trong mỗi chu kỳ, node sẽ chọn một hoặc nhiều node khác (peer) trong cluster để gửi thông tin gossip.
- Trao đổi thông tin: Node gửi cho peer các bản cập nhật trạng thái mà nó biết, thường là các thông tin mới hoặc được cập nhật gần đây nhất.
- Cập nhật trạng thái: Sau khi nhận được thông tin gossip từ peer, node sẽ so sánh và cập nhật trạng thái của mình nếu có thông tin mới hoặc khác biệt.
- Tiếp tục lan truyền: Các node nhận thông tin mới sẽ tiếp tục lan truyền nó tới các node khác trong các vòng gossip kế tiếp.

Ưu điểm của Gossip Protocol

- Phi tập trung: Không cần có node điều phối trung tâm, tránh điểm nghẽn và điểm lỗi duy nhất.
- Khả năng chịu lỗi: Nếu một số node mất kết nối, thông tin vẫn có thể lan truyền qua các node còn lại.
- Khả năng mở rộng: Hoạt động hiệu quả với cluster có quy mô lớn do mỗi node chỉ trao đổi với một số nhỏ các node khác.
- Độ tin cậy cao: Thông tin được truyền đi nhiều lần và từ nhiều nguồn, giảm khả năng mất mát thông tin.

Trong hệ thống key-value store phân tán của bài tập, gossip protocol được sử dụng để đồng bộ danh sách node và trạng thái cluster giữa các node. Khi có node mới tham gia hoặc node rời khỏi cluster, thông tin này sẽ được cập nhật và lan truyền qua các vòng gossip, giúp các node cập nhật hash ring tương ứng.

Gossip cũng hỗ trợ trong việc phát hiện node chết dựa trên cơ chế timeout và cập nhật trạng thái node trong peer list. Mỗi node sẽ tự đánh giá trạng thái của các peer thông qua việc theo dõi heartbeat hoặc không nhận được phản hồi trong khoảng thời gian quy định.

Một số lưu ý và hạn chế

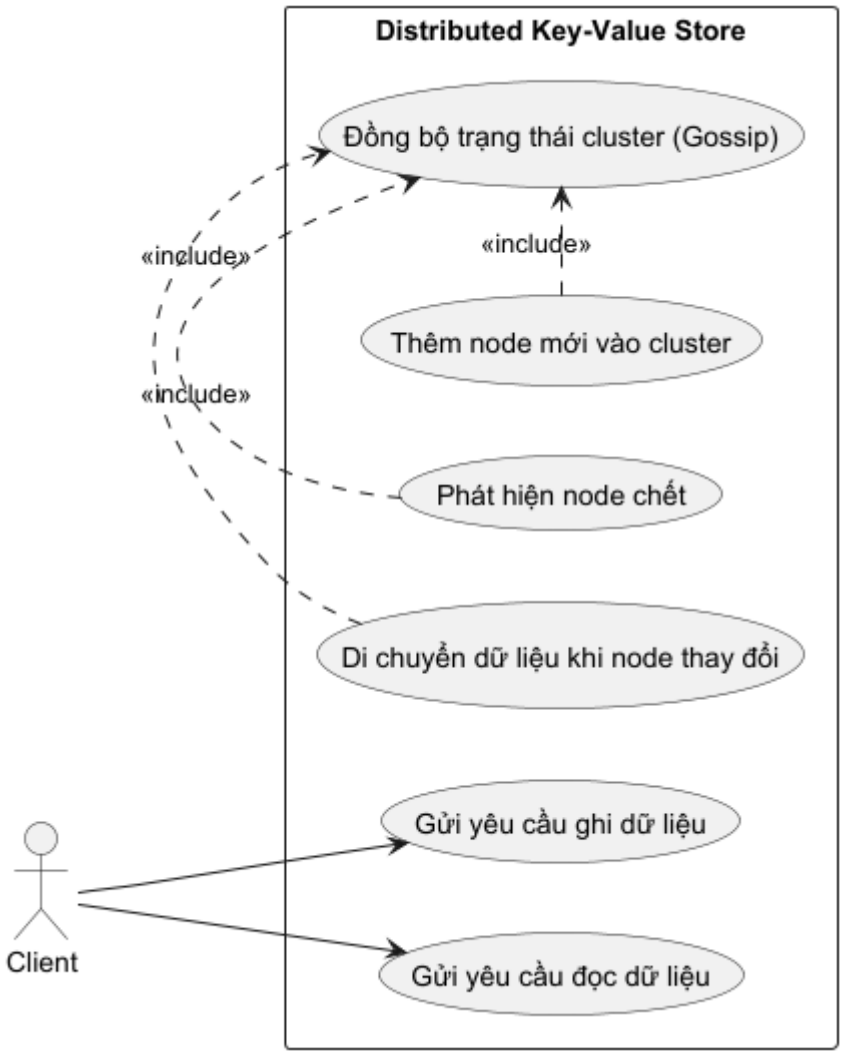
- Độ trễ thông tin: Do bản chất lan truyền từng bước, việc cập nhật trạng thái mới có thể mất một vài chu kỳ gossip, gây độ trễ trong đồng bộ.
- Sử dụng băng thông: Mặc dù gossip giảm tải so với broadcast toàn bộ, việc gửi thông tin thường xuyên cũng tiêu tốn băng thông, đặc biệt với cluster lớn.
- Xử lý trạng thái mâu thuẫn: Cần có cơ chế versioning hoặc incarnation number để phân biệt thông tin mới và cũ, tránh nhầm lẫn khi node chết và hồi phục.

2.5 Công nghệ sử dụng

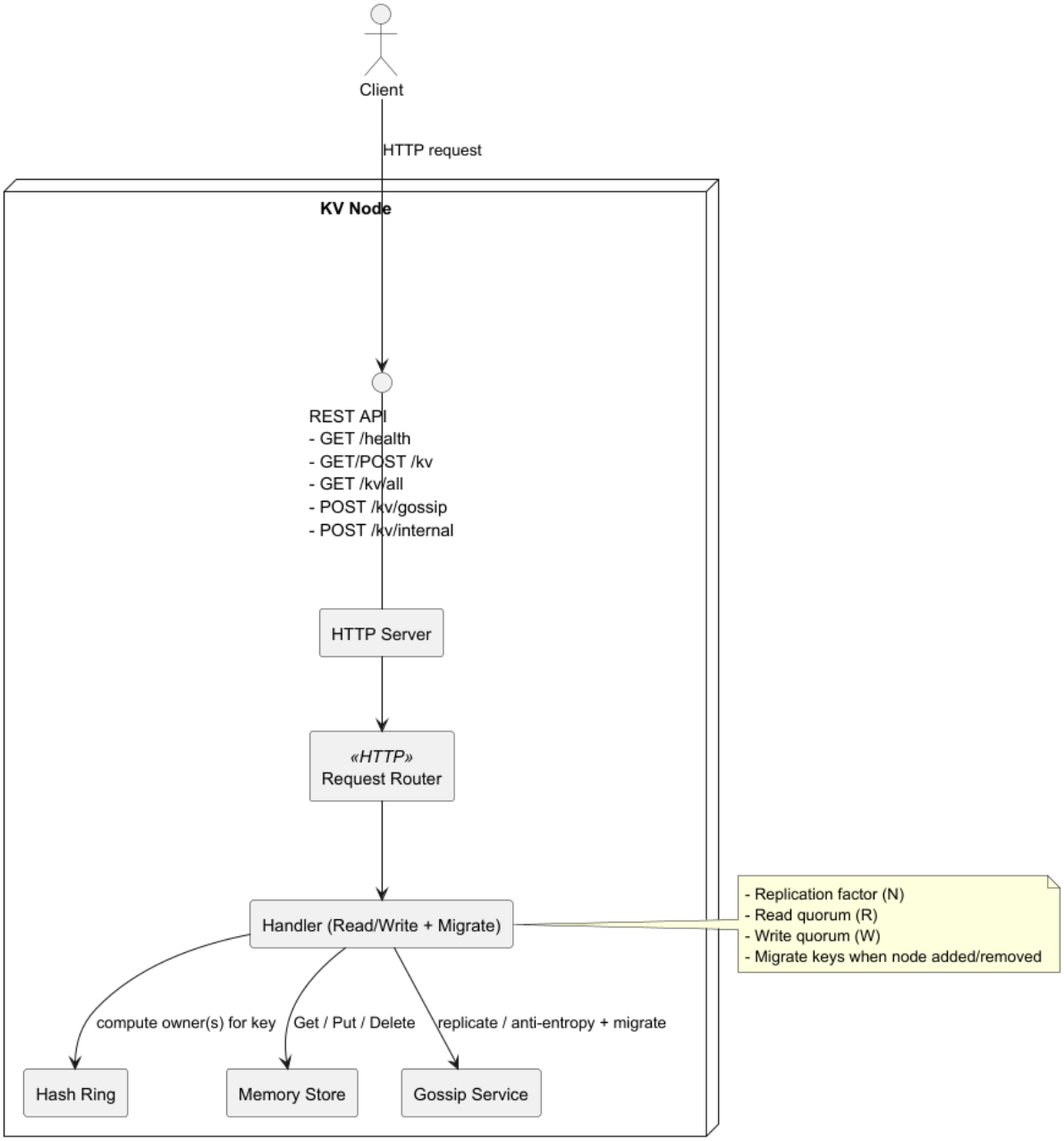
- Ngôn ngữ lập trình: Go (Golang)
  - Bài tập sử dụng ngôn ngữ Go nhờ vào ưu điểm về hiệu năng cao, hỗ trợ tốt cho lập trình song song (concurrency) qua goroutine và channel, cùng thư viện chuẩn phong phú hỗ trợ phát triển hệ thống mạng phân tán nhanh và hiệu quả.
- Giao thức mạng: HTTP
  - Giao tiếp giữa các node được thực hiện qua HTTP đơn giản và phổ biến. Mỗi node xây dựng các API RESTful để xử lý các yêu cầu đọc, ghi dữ liệu, cập nhật trạng thái gossip, và migrate dữ liệu. Việc sử dụng HTTP giúp dễ dàng mở rộng, debug, và tích hợp với các công cụ hiện có mà không cần triển khai các giao thức phức tạp như RPC.
- Hàm băm tự triển khai
  - Để tính toán vị trí của key trên vòng băm, hệ thống sử dụng hàm băm được tự viết dựa trên thuật toán SHA-1, nhưng chỉ lấy 4 bytes đầu tiên của giá trị băm để đơn giản hóa và tối ưu hiệu suất.
- Thư viện hỗ trợ
  - net/http: Cung cấp khả năng xây dựng HTTP server và client, phục vụ cho giao tiếp giữa các node.
  - crypto/sha1: Thư viện chuẩn để thực hiện hàm băm SHA-1.
  - encoding/json: Dùng để tuần tự hóa dữ liệu JSON gửi nhận qua HTTP.
  - sync: Đảm bảo an toàn truy cập dữ liệu trong môi trường đa luồng.

Chương 3. Thiết kế chi tiết

3.1 Biểu đồ use case

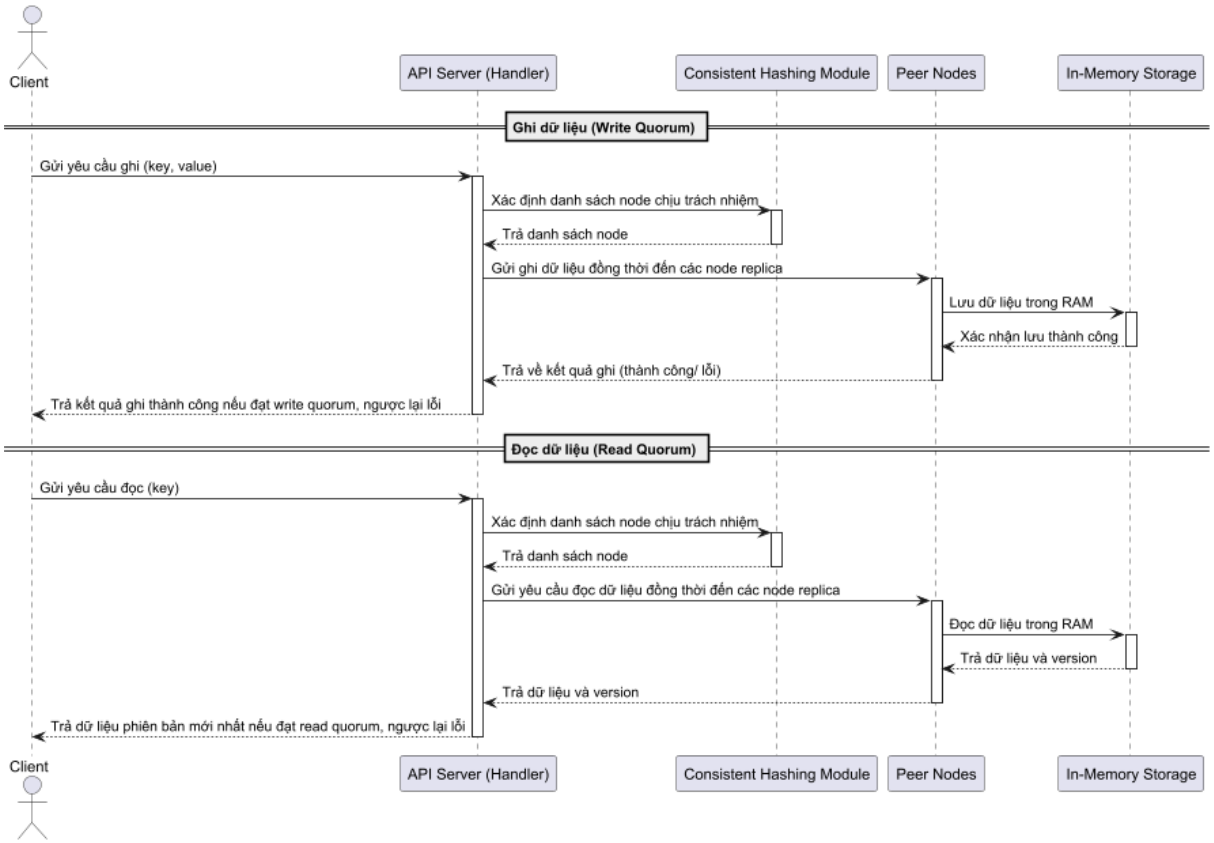


3.2 Biểu đồ các thành phần

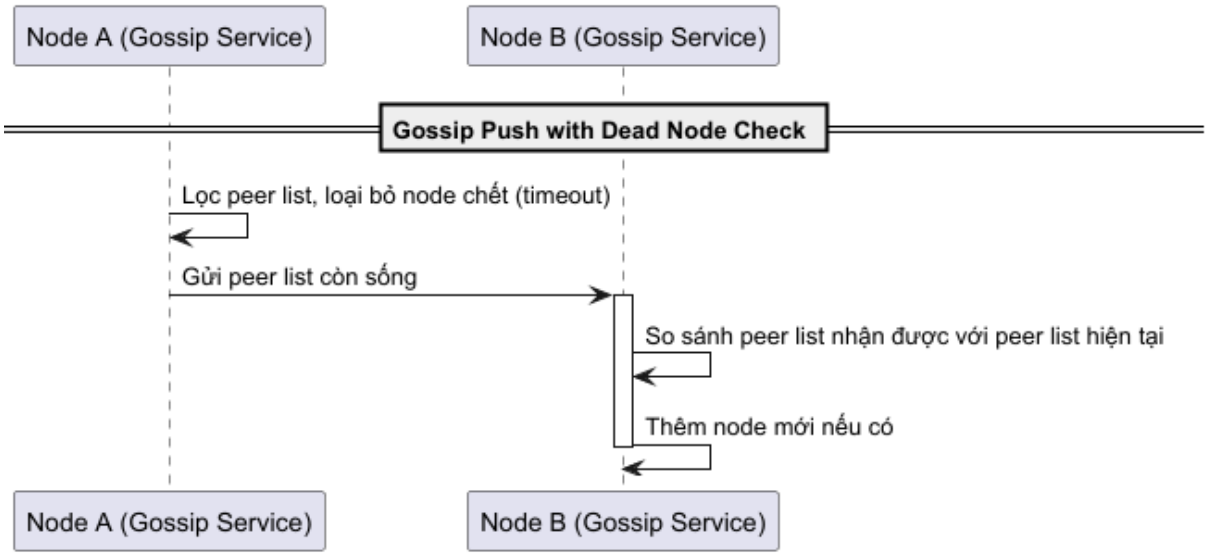


3.3 Biểu đồ tuần tự

3.3.1 Luồng đọc/ghi từ client



3.3.2 Luồng gossip



Chương 4. Kết quả thực nghiệm

4.1 Môi trường thử nghiệm

- 4 node chạy trên cùng máy local, mỗi node chạy trên một cổng khác nhau:
  - Node A: port 8001
  - Node B: port 8002
  - Node C: port 8003
  - Node D: port 8003
- Sử dụng cơ chế replication factor = 3.
- Quorum đọc/ghi: quorum size = 2 (yêu cầu ít nhất 2 node phản hồi để đọc/ghi thành công).
- Dữ liệu ban đầu được phân phối và replicate theo consistent hashing.

4.2 Kịch bản và kết quả

4.2.1 Hệ thống có đủ 4 node

Kịch bản	**Kết quả**
Client gửi yêu cầu ghi đến node A với Key = 12345, Value = 1	Hệ thống lưu key vào các node A, B, D, đạt write quorum 3 >= 2, ghi thành công
Client gửi yêu cầu đọc đến node A với Key = 12345	Hệ thống lấy key từ các node A, B, D, đạt read quorum 3 >= 2, đọc thành công

4.2.2 Số node hoạt động < quorum

Kịch bản	**Kết quả**
Tắt 3 node B, C, D Client gửi yêu cầu ghi đến node A với Key = 12345, Value = 1	Hệ thống báo lỗi do write quorum 1 < 2
Tắt 3 node B, C, D Client gửi yêu cầu đọc đến node A với Key = 12345	Hệ thống báo lỗi do read quorum 1 < 2

4.2.3 Xóa một node

Kịch bản	**Kết quả**
Thực hiện ghi như kịch bản 4.2.1 Sau đó tắt node D	Sau một khoảng thời gian, các node còn lại cập nhật xoá node D trong hash ring của mình và gửi key migrate đến node C

4.2.4 Thêm lại node

Kịch bản	Kết quả
Tiếp tục kịch bản 4.2.3 Bật lại node D	Các node nhận gossip từ node D thì cập nhật thêm node D vào hash ring của mình và gửi key migrate đến node D

Chương 5. Kết luận

Trong đề tài này, nhóm đã xây dựng một hệ thống lưu trữ khóa–giá trị phân tán sử dụng các cơ chế chính:

- Consistent Hashing giúp phân phối dữ liệu đồng đều giữa các nút, hạn chế việc di chuyển dữ liệu khi có thay đổi về số lượng nút.
- Read/Write Quorum đảm bảo tính nhất quán dữ liệu ở mức cấu hình được, cân bằng giữa hiệu năng và độ tin cậy.
- Gossip Protocol hỗ trợ trao đổi thông tin trạng thái giữa các nút, giúp hệ thống phát hiện và xử lý sự cố nhanh chóng.

Kết quả thử nghiệm cho thấy hệ thống vẫn hoạt động ổn định khi có nút bị lỗi hoặc khi mở rộng/thu hẹp cụm. Dữ liệu được phân phối và đồng bộ đúng như thiết kế, giảm thiểu mất mát và đảm bảo khả năng truy cập cao.

Tuy nhiên, hệ thống vẫn còn một số hạn chế:

- Chưa tối ưu cho khối lượng dữ liệu quá lớn hoặc tốc độ thay đổi cấu hình cao.
- Chưa triển khai cơ chế sao lưu dự phòng ngoài cụm.
- Chưa có cơ chế tự động điều chỉnh tham số quorum theo tình trạng mạng.

Trong tương lai, có thể mở rộng hệ thống theo các hướng:

- Tích hợp cơ chế nén và mã hóa dữ liệu.
- Hỗ trợ phân vùng dữ liệu theo nhiều tiêu chí khác ngoài khóa.
- Nghiên cứu áp dụng thuật toán phát hiện lỗi tiên tiến hơn để giảm thời gian gián đoạn.

Với kết quả đạt được, đề tài chứng minh rằng mô hình kết hợp Consistent Hashing, Quorum và Gossip là một giải pháp hiệu quả cho hệ thống lưu trữ phân tán yêu cầu tính sẵn sàng cao, khả năng mở rộng linh hoạt, và đảm bảo dữ liệu nhất quán trong môi trường nhiều thay đổi.

Chương 6. Tài liệu tham khảo

[1] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, New York, NY, USA, 1987.

[2] D. K. Gifford, "Weighted voting for replicated data," in *Proceedings of the Seventh ACM Symposium on Operating Systems Principles*, New York, NY, USA, 1979.

[3] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine and D. Lewin, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, New York, NY, USA, 1997.