ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN HỮU LOAN

GIẢI PHÁP BACKUP DỮ LIỆU, SỬ DỤNG CƠ CHẾ PHÂN CỤM ĐỘNG TRONG MẠNG NGANG HÀNG CÓ CẤU TRÚC

LUẬN VĂN THẠC SỸ: NGÀNH CÔNG NGHỆ THÔNG TIN

Hà Nội - Năm 2017

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN HỮU LOAN

GIẢI PHÁP BACKUP DỮ LIỆU, SỬ DỤNG CƠ CHẾ PHÂN CỤM ĐỘNG TRONG MẠNG NGANG HÀNG CÓ CẤU TRÚC

Ngành: Công nghệ thông tin
Chuyên ngành: Hệ thống thông tin
Mã số: 60.48.01.04

LUẬN VĂN THẠC SỸ: NGÀNH CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. Nguyễn Hoài Sơn

Hà Nội - Năm 2017

LỜI CẢM ƠN

Lời đầu tiên tôi xin gửi lời cảm ơn chân thành và sâu sắc đến thầy giáo TS. Nguyễn Hoài Sơn, một người thầy vô cùng tâm huyết đã hướng dẫn, giúp đỡ và động viên tôi trong suốt thời gian nghiên cứu và hoàn thiện luận văn.

Tôi xin chân thành cảm ơn các thầy, cô giáo Khoa Công nghệ Thông tin trường Đại học Công nghệ - Đại học Quốc gia Hà Nội đã truyền đạt kiến và tạo điều kiện tốt nhất trong suốt quá trình tôi học tập và nghiên cứu tại trường.

Tôi xin chân thành cảm ơn anh Nguyễn Đình Nghĩa, người đã giúp đỡ, hướng dẫn và hỗ trợ nhiệt tình tôi trong suốt quá trình nghiên cứu và xây dựng luận văn.

Tôi xin chân thành cảm ơn tất cả các bạn học viên cao học đã chia sẻ và giúp đỡ tôi rất nhiều trong quá trình hoàn thành các môn học tại trường. Nhân đây tôi cũng xin chân thành cảm ơn gia đình, bạn bè và các đồng nghiệp đã ủng hộ tinh thần, tạo điều kiện để tôi học tập và nghiên cứu chương trình thạc sỹ Đại học Công nghệ Đại học Quốc gia Hà Nội.

Hà Nội, ngày 28 tháng 3 năm 2017

Học Viên

Nguyễn Hữu Loan

LỜI CAM ĐOAN

Tôi xin cam đoan rằng luận văn thạc sỹ công nghệ thông tin "Giải pháp backup dữ liệu sử dụng cơ chế phân cụm động, trong mạng ngang hàng có cấu trúc" là công trình nghiên cứu của riêng tôi dưới sự hướng dẫn của TS. Nguyễn Hoài Sơn, không sao chép lại của người khác. Các tài liệu tham khảo được trích dẫn và chú thích đầy đủ.

Hà Nội, ngày 28 tháng 3 năm 2017

Học Viên

Nguyễn Hữu Loan

MỤC LỤC

DANH MỤC CÁC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Từ viết tắt	Giải nghĩa
Broadcast	Là cách thức truyền gói tin từ một điểm tới tất cả các điểm
Capacity	Khả năng lưu trữ của một node
Chord	Là một giao thức trong mạng ngang hàng biểu diễn mạng dưới dạng vòng tròn.
Node	Diễn tả một thực thể trong mạng như là peer hoặc máy tính kết nối mạng
DHT (Distributed Hash Table)	Bảng băm phân tán
Entry	Là một bước định tuyến trong bảng định tuyến
Mobile agent	Là chương trình có khả năng di chuyển một cách tự trị từ nút mạng này sang node mạng khác để hoàn tất tác vụ
ID (Identification number)	Số định danh
Peer	Một node trong mạng ngang hàng
P2P (Peer to peer)	Mạng ngang hàng
Random walk	Là cơ chế một node tìm kiếm và thu thập danh sách các Hub sau đó liên kết trực tiếp tới chúng
Server	Máy chủ
Supernode	Là một node tương tự như server, có khả năng chuyển tiếp thông tin và kết nối tới nhiều node khác trong hệ thống

DANH MỤC HÌNH VẼ

DANH MỤC CÁC BIỂU ĐỒ

DANH MỤC CÁC BẢNG

MỞ ĐẦU

Trong những năm gần đây, mạng ngang ngàng đã phát triển nhanh chóng, nhiều ứng dụng sử dụng mạng ngang hàng để hỗ trợ chia sẻ file, video, tin nhắn nhanh như Bittorrent, eDonkey, Fshare tool, Megadownloader. Các ứng dụng này sử dụng phương pháp phân mảnh để chia sẻ một file, các máy trong mạng liên kết với nhau để lấy các mảnh từ nhiều nguồn khác nhau để có được đầy đủ các mảnh và lắp ghép thành file. Phương pháp này vừa giảm tải cho các máy, vừa có thể lấy thông tin nhanh hơn và dễ dàng hơn, tận dụng được băng thông và không cần sử dụng các server trung tâm với cấu hình cao và có thể bị nghẽn cổ chai khi số lượng truy cập vào hệ thống lớn.

Trong thời kỳ đầu phát triển của mạng ngang hàng, việc tìm kiếm, chia sẻ thông tin thông qua hình thức sử dụng cơ chế broadcast, là cơ chế phát tràn các thông báo tới các máy trong mạng, gây tốn kém tài nguyên và hiệu quả tìm kiếm thấp do không đảm bảo việc quét thông tin cho toàn hệ thống.

Mạng ngang hàng có cấu trúc được hình thành sau này đã khắc phục được những nhược điểm của cơ chế broadcast, thông qua việc sử dụng bảng băm phân tán DHT (Distributed Hash Table), điển hình như Chord, CAN[12], Kademlia, Tapestry, Kelips. Theo phương pháp này, không gian ID được tổ chức dưới dạng vòng, dữ liệu trong mạng được quản lý dưới dạng (key, value), các node liên kết và biết đến nhau thông qua bảng định tuyến. Với cấu trúc này, khi một máy tính cần tìm một dữ liệu, nó chỉ cần áp dụng một giao thức chung để xác định nút mạng nào chịu trách nhiệm cho dữ liệu đó và sau đó liên lạc trực tiếp đến nút mạng đó để lấy kết quả.

Mặc dù mạng ngang hàng có cấu trúc cho thấy được những ưu điểm vượt trội thông qua việc sử dụng bảng băm DHT và bảng định tuyến để tìm kiếm và chia sẻ thông tin, tuy vậy trong quá trình hoạt động của mạng vẫn còn nhiều vấn đề chưa được giải quyết. Trong đó, có vấn đề đảm bảo việc phục hồi dữ liệu trong mạng khi các node trong mạng thường xuyên gia nhập hoặc rời khỏi mạng và khả năng cân bằng tải giữa các node chưa cao. Luận văn "Giải pháp backup dữ liệu, sử dụng cơ chế phân cụm động trong mạng ngang hàng có cấu trúc" sẽ đề xuất một phương pháp cải tiến việc backup dữ liệu, theo cơ chế phân cụm động nhằm khắc phục các vấn đề nêu trên.

Về bố cục, nội dung của luận văn bao gồm 4 chương:

Chương 1: Tổng quan về kiến trúc hệ thống mạng ngang hàng: Chương này giới thiệu về các kiến trúc mạng ngang hàng như kiến trúc tập trung, kiến trúc phân tán và kiến trúc hỗn hợp, mỗi kiến trúc có những đặc điểm riêng và đi sâu vào mô tả một số hệ thống áp dụng với từng kiến trúc.

Chương 2: Các phương pháp backup dữ liệu trên mạng ngang hàng có cấu trúc: Mô tả cơ chế backup trong mạng ngang hàng có cấu trúc, dựa trên giao thức Chord. So sánh hai phương pháp backup successor list *(phương pháp backup Chord nguyên thủy)* và phân cụm tĩnh.

Chương 3: Phương pháp phân cụm động và cơ chế backup: Đưa ra các nguyên tắc và phương pháp chung của việc tách cụm động, nêu ra phương pháp phân mảnh dữ liệu và các trường hợp xử lý việc backup dữ liệu.

Chương 4: Đánh giá hiệu quả phương pháp tách nhập cụm theo cơ chế phân cụm động: Mô tả hoạt động của chương trình mô phỏng, so sánh các tiêu chí của phân cụm tĩnh và phân cụm động.

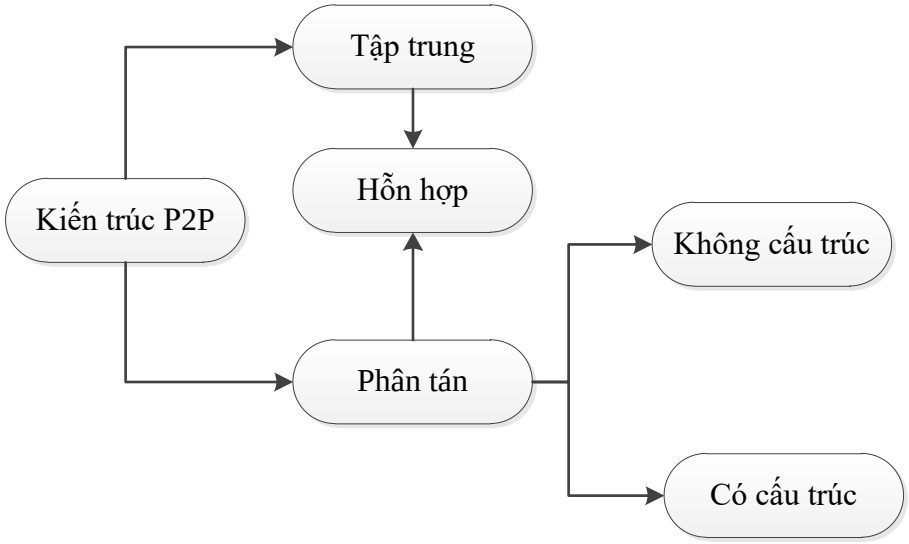
Kết luận và hướng phát triển: Tóm tắt, đề xuất hướng phát triển.

.

CHƯƠNG 1: TỔNG QUAN VỀ KIẾN TRÚC HỆ THỐNG MẠNG NGANG HÀNG

Trong chương này sẽ giới thiệu một số kiến trúc hệ thống mạng mang hàng, mô tả các đặc điểm chung, các thuộc tính và một số hệ thống áp dụng cho mỗi kiến trúc đưa ra.

Nhìn chung, mạng ngang hàng được phân thành hai hệ thống chính là hệ thống tập trung và hệ thống phân tán, dựa trên tính sẵn sàng của một hay nhiều server. Bên cạnh đó còn có hệ thống hỗn hợp là hệ thống vừa có những đặc điểm của hệ thống tập trung và hệ thống phân tán. Hình 1-1 mô tả sơ đồ phân loại kiến trúc hệ thống P2P.



Hình 1-1 Phân loại kiến trúc P2P

Các nội dung tiếp theo của chương sẽ mô tả chi tiết từng kiến trúc này.

1.1 Hệ thống P2P tập trung

Trong hệ thống P2P tập trung, có một hay nhiều server giúp cho các peer xác định vị trí tài nguyên mong muốn hoặc phối hợp các hoạt động giữa các peer với nhau. Để định vị tài nguyên, một peer gửi thông điệp tới server trung tâm để xác định địa chỉ peer mà chứa tài nguyên mong muốn. Khi xác định được peer có thông tin hay dữ liệu, nó có thể liên kết trực tiếp với các peer đó để trao đổi thông tin mà không qua server nữa [1].

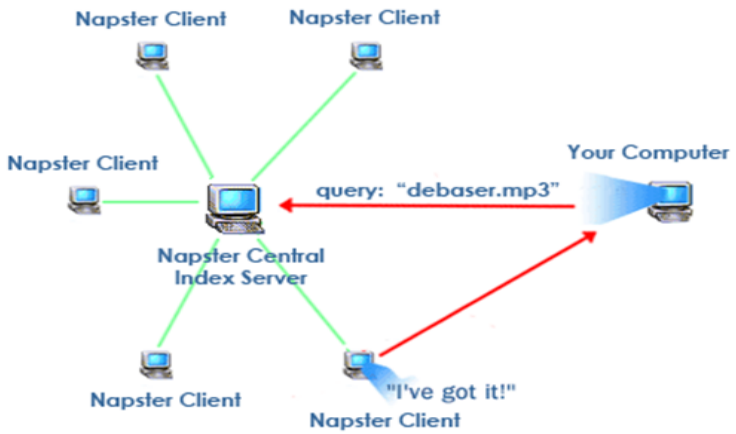
Kiến trúc hệ thống tập trung này dễ bị tấn công vào liên kết đến server, mặt khác nó còn là nút thắt cổ chai đối với hệ thống có số peer lớn, tiềm ẩn việc

làm giảm hiệu năng một cách đột ngột, ngoài ra mô hình này hạn chế khả năng mở rộng, điển hình của mô hình này là Napster [16].

Napster: Chia sẻ nội dung số

Chia sẻ file nhạc có lẽ là một trong những ứng dụng phát triển nhanh nhất Internet và Napster đóng một vai trò quan trọng trong việc thúc đẩy trao đổi file nhạc qua Internet. Trong Napster, mỗi user (peer) hoạt động như một máy tạo dữ liệu (file MP3), vì vậy hệ thống có thể tạo ra các file MP3 phân tán, các user có thể chia sẻ nội dung cho nhau trong cùng hệ thống. Để xác định vị trí các file nhạc, Napster triển khai server tập trung lưu trữ vị trí của các node mà chứa file. Ngoài ra còn cung cấp các chức năng cơ bản như là phương tiện tìm kiếm, chia sẻ file và chuyển tiếp thông điệp qua Internet. File chia sẻ cung cấp cơ chế để chuyển tới các Peer mà không sử dụng không gian lưu trữ trong server tập trung, thông điệp được truyền qua internet đồng thời cung cấp cách tìm kiếm và liên hệ qua các ứng dụng tin nhắn nhanh với những peer đang online.

Một file MP3 đưa vào hệ thống Napster chia làm 3 giai đoạn: Tham gia vào mạng Napster, tìm tài nguyên và tải file. Trước tiên thông qua các kết nối khác nhau, một user có thể gia nhập vào mạng Napster bằng cách kết nối tới server trung tâm và hoàn thành thủ tục đăng ký trên server trung tâm. Thứ hai, một peer truy vấn tới server trung tâm bằng cách gửi đi thông điệp tìm kiếm. Sau khi nhận được thông điệp, server trung tâm tìm lại danh mục trong kho lưu trữ cục bộ và trả lại danh sách các node chứa file yêu cầu. Cuối cùng truy vấn được kết nối trực tiếp với peer yêu cầu và tải file mà không thông qua server trung tâm. Hình 1-2 dưới đây mô tả hoạt động của mạng Napster.



Hình 1-2 Mô hình mạng Napster

Những thuộc tính và giới hạn của hệ thống P2P tập trung như sau:

Khả năng phục hồi lỗi, tính riêng tư và tính bảo mật: Do dựa vào server trung tâm để lưu thông tin về user online và chia sẻ file nên hệ thống có dễ bị tấn công vào đường truyền kết nối đến server. Ngoài ra, một peer truy vấn có thể chứa địa chỉ IP của các Peer khác từ server trung tâm, điều này có thể phá hủy tính ẩn danh và riêng tư của các peer. Việc biết được địa chỉ IP, các user chứa mã độc có thể tấn công trực tiếp tới các peer hoặc lấy cắp những thông tin có giá trị từ các peer này. Vì thế, tiềm ẩn những mối nguy hiểm trong mạng và không đảm bảo tính bảo mật.

Tính mở rộng: Trong mạng Napster, tất các peer phải kết nối tới server trung tâm và tất cả các truy vấn phải được xử lý ở server trung tâm trước tiên. Trường hợp server bị giới hạn về khả năng xử lý, trong khi số lượng kết nối và truy vấn tại một thời điểm vượt quá khả năng server làm thời gian trả lời kéo dài hoặc bị các cuộc tấn công từ chối dịch vụ (DoS). Vì vậy khả năng mở rộng và tính mạnh mẽ bị giới hạn.

Tính sẵn sàng: Tính sẵn sàng là mức độ hoặc khả năng truy vấn của một peer có thể tìm thấy dữ liệu mong muốn từ các peer khác. Sau khi một peer tải file MP3 từ các peer khác, nó sẽ duy trì một bản sao của những file này trong vùng lưu trữ cục bộ và có thể được sử dụng cho các peer khác truy vấn đến, do đó trao đổi file nhạc trong các peer được cải thiện được tính sẵn sàng.

Tính phân tán: Mức độ phân tán thấp, do một server trung tâm được triển khai để quản lý hoạt động của cả hệ thống, bù lại việc xử lý tìm kiếm khá hiệu quả.

Chi phí sở hữu: Một tính năng thú vị trong hệ thống P2P tập trung là chi phí sở hữu thấp, bao gồm chi phí duy trì liên kết đến các tài nguyên khác nhau trong mạng P2P. Trong trường hợp kiến trúc client-server, một server mạnh được sử dụng để lưu trữ, chia sẻ tài nguyên cho các client tải dữ liệu và cung cấp cho các dịch vụ khác của client, do đó phải tốn chi phí cao để duy trì các server có cấu hình mạnh.

Năng suất và hiệu quả: Thành công của Napster chứng minh rằng với việc kiểm soát tập trung sẽ thúc đẩy quá trình định vị tài nguyên với chi phí rẻ và hiệu quả.

Mặc dù hệ thống P2P tập trung cho thấy những điểm mạnh như đảm bảo quá trình định định vị tài nguyên cho việc tìm kiếm. Dễ dàng duy trì, tổ chức và quản lý toàn bộ hệ thống thông qua server trung tâm, tuy nhiên hệ thống này cũng cho thấy một số nhược điểm sau:

- Server trung tâm có thể trở thành nút cổ chai cho việc mở rộng hệ thống
- Server trung tâm gặp lỗi sẽ làm ảnh hưởng tới toàn bộ hệ thống.

1.2 Hệ thống P2P phân tán

Trong hệ thống phân tán các peer có quyền và trách nhiệm như nhau. Mỗi peer chỉ có thông tin một phần trong mạng và yêu cầu dữ liệu hay dịch vụ thông qua một số peer khác[2]. Như vậy việc xác định các peer yêu cầu dữ liệu hay dịch vụ nhanh là một vấn đề và thách thức đối với hệ thống này. Hệ thống P2P phân tán được chia thành hai hệ thống là hệ thống P2P phân tán không cấu trúc và hệ thống P2P phân tán có cấu trúc, khác nhau giữa hai hệ thống này là phương pháp các truy vấn chuyển đến các node.

1.2.1. Hệ thống P2P không cấu trúc

Trong hệ thống này, mỗi peer chịu trách nhiệm đối với dữ liệu riêng và duy trì thiết lập với node lân cận để liên kết, trao đổi thông tin và chuyển các truy vấn cho nhau. Việc định vị dữ liệu trong hệ thống này gặp khó khăn như:

- Khó xác định đúng peer chứa dữ liệu để để truy vấn.
- Không đảm bảo cho việc trả lời đầy đủ các truy vấn, trừ khi tìm kiếm cho toàn mạng.
- Không đảm bảo về thời gian trả lời.

Điển hình của hệ thống P2P không cấu trúc là FreeNet và Guntella, ban đầu áp dụng cơ chế tìm kiếm random walk [15] để xác định tài nguyên. Cơ chế này mặc dù không hiệu quả về thời gian trả lời, nhưng hiệu quả về tiêu thụ băng thông và số lượng thông điệp sử dụng ít. Sau này dựa theo cơ chế định tuyến flooding, hiệu quả về thời gian đáp ứng nhưng không hiệu quả về tiêu thụ băng thông và số lượng thông điệp sử dụng.

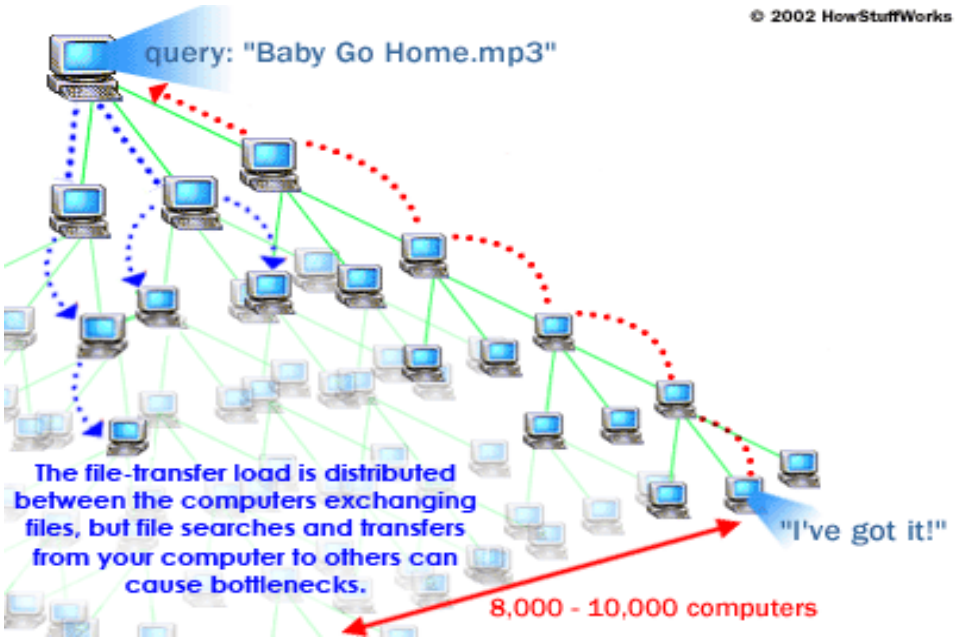
Gnutella: Hệ thống P2P thuần túy đầu tiên

Gnutella là hệ thống phân tán thuần túy, không có node trung tâm chịu trách nhiệm tổ chức mạng và không phân biệt giữa client và server. Các node trong hệ thống kết nối với nhau thông qua một phần mềm ứng dụng cụ thể. Mạng Gnutella được mở rộng khi node mới tham gia vào mạng và bị thu hẹp khi các node rời mạng. Hoạt động cơ bản của Gnutella bao gồm việc tham gia, rời mạng, tìm kiếm và tải các file.

Tham gia hoặc rời mạng: Khi một node tham gia mạng Gunutella, nó sẽ gửi thông điệp "ping" tới các node mà nó biết địa chỉ *(đã được cài đặt sẵn hoặc thông qua các node khác)* thông điệp này sẽ chuyển tới các node khác thông qua hình thức broadcast. Khi các node nhận được thông tin "ping" sẽ phản hồi thông tin "pong". Node tham gia có thể nhận thông tin về các node đó và thiết lập

thông tin các node lân cận của nó. Khi một node rời mạng, nó không cần thông báo tới các node lân cận của nó, mà thăm dò node lân cận thông qua thông điệp "ping" theo định kỳ, để xác nhận rằng các node lân cận có hoạt động trong hệ thống hay không.

Tìm kiếm và tải file: Khi một node muốn tìm một file xác định, nó hỏi các node lân cận bằng cách đưa ra thông điệp tìm kiếm, tiếp đó các node lân cận sẽ chuyển tiếp thông điệp tới các node lân cận của nó theo cách tương tự. Khi tìm được node chứa file, nó sẽ trả lại thông tin theo cách chuyển tiếp thông điệp ngược lại tới node cần tìm ban đầu. Kết quả là node gốc sau khi truy vấn sẽ có nhiều liên kết để tải file mong muốn và có thể chọn một vài node để kết nối và tải file. Tuy nhiên, mỗi thông điệp gắn với một định danh duy nhất, khi một node nhận được thông điệp mà nó đã được nhận trước đó thì nó sẽ hủy thông điệp đó để tránh các thông điệp lặp lại. Hình 1-3 mô tả quá trình tìm kiếm và trả lời kết quả tìm kiếm trong mạng Gnutella.



Hình 1-3 Mô hình trao đổi và tìm kiếm thông tin trong Gnutella

Những thuộc tính và giới hạn của hệ thống P2P phân tán không cấu trúc như sau:

Khả năng mở rộng: Cơ chế flooding của Gnuttela có hai mặt, một mặt là mỗi truy vấn có thể flooding tới nhiều node ở trong mạng, nên nó rất mạnh mẽ trong việc tìm kiếm tất cả các kết quả có thể. Ngược lại khi các node tham gia

vào mạng Gnutella ngày càng nhiều lên, và các node đưa ra các truy vấn liên tục dẫn đến mạng có thể bị tắc nghẽn, điều này làm hạn chế khả năng mở rộng của Gnutella.

Tính tự tổ chức: Khi một node lần đầu kết nối đến mạng Gnutella, nó chọn ngẫu nhiên một điểm và duy trì ở đó, dần dần nó thiết lập quan hệ và xây dựng kết nối tới các node khác. Tuy nhiên quan hệ giữa các kết nối này không cố định. Để đảm bảo chăc chắn rằng các truy vấn có thể nhanh nhất và tốt nhất có thể, nó duy trì kết nối tới các node có băng thông cao, các node này dần dần là trung tâm kết nối. Các node có băng thông thấp dần sẽ đẩy xuống kết nối với ưu tiên thấp hơn trong bảng định tuyến.

Tính ẩn danh: Gnutella là một hệ thống có mức độ ẩn danh khá tốt, thông qua sử dụng cơ chế broadcast để chuyển các truy vấn. Do cơ chế broadcast dựa vào bảng định tuyến và thay đổi theo thời gian, vì vậy khó xác định được các node truy vấn đến hay các node gửi đi. Tuy nhiên, tính ẩn danh bị phá vỡ khi node gốc chọn một hay nhiều node để thiết lập kết nối và tải file, vì khi đó nó phải cung cấp địa chỉ IP của cả node cung cấp và node yêu cầu cho nhau.

Tính sẵn sàng: Một node trong mạng Gnutella có thể kết nối hoặc hủy kết nối vào bất kỳ thời điểm nào, mà không cần báo trước do không có cơ chế kiểm soát tính sẵn sàng và ổn định, vì vậy tính sẵn sàng của Gnutella không đảm bảo.

1.2.2 Hệ thống P2P có cấu trúc

Hệ thống P2P không cấu trúc thể hiện nhược điểm là không có gì đảm bảo tìm kiếm sẽ thành công. Đối với tìm kiếm các dữ liệu phổ biến được chia sẻ trên nhiều máy, tỉ lệ thành công khá cao, ngược lại nếu dữ liệu chỉ được chia sẻ trên một vài máy thì xác suất tìm thấy là khá thấp. Tính chất này là hiển nhiên vì trong mạng ngang hàng không cấu trúc, không có bất kỳ mối tương quan nào giữa một máy và dữ liệu nó quản lý trong mạng, do đó yêu cầu tìm kiếm được chuyển một cách ngẫu nhiên đến một số máy trong mạng. Số lượng máy trong mạng càng lớn thì khả năng tìm thấy thông tin càng thấp. Một nhược điểm khác của hệ thống này là do không có định hướng, một yêu cầu tìm kiếm thường được chuyển cho một số lượng lớn máy trong mạng, làm tiêu tốn một lượng băng thông của mạng dẫn đến hiệu quả tìm kiếm chung của mạng thấp.

Mạng ngang hàng có cấu trúc khắc phục nhược điểm của mạng ngang hàng không cấu trúc bằng cách sử dụng bảng băm phân tán DHT. Hệ thống này định nghĩa liên kết giữa các node mạng trong mạng phủ theo một thuật toán cụ thể, đồng thời xác định chặt chẽ mỗi node mạng sẽ chịu trách nhiệm đối với một

phần dữ liệu chia sẻ trong mạng. Với cấu trúc này, khi một máy cần tìm một dữ liệu, nó chỉ cần áp dụng một giao thức chung để xác định node mạng nào chịu trách nhiệm cho dữ liệu đó và sau đó liên lạc trực tiếp đến node mạng đó để lấy kết quả.

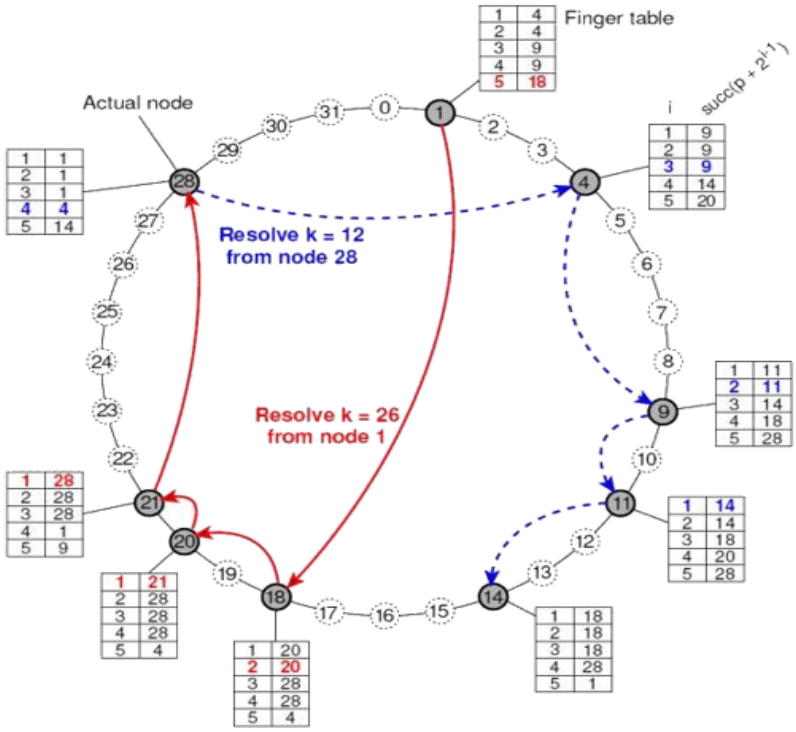
Trong hệ thống P2P có cấu trúc, tài nguyên được phân bố một cách hợp lý để không có một máy tính nào lưu giữ quá nhiều dữ liệu dẫn đến quá tải thông tin định tuyến. Do mạng là có cấu trúc nên các thông điệp chuyển đi giữa các máy tính để duy trì mạng ngang hàng được giảm xuống tối thiểu. Băng thông của mạng được dành nhiều hơn cho việc chia sẻ tài nguyên.

Việc tìm kiếm thông tin trong hệ thống P2P có cấu trúc nhanh hơn so với hệ thống P2P không cấu trúc. Nếu như trong hệ thống P2P không cấu trúc các máy tính gửi thông điệp broadcast để tìm kiếm thông tin thì trong hệ thống P2P có cấu trúc, một máy tính chỉ cần gửi thông điệp tìm kiếm qua một số máy tính. Giao thức tìm kiếm chung trong mạng sẽ đảm bảo thông tin được tìm kiếm chính xác.

Điển hình của hệ thống này là CAN, Chord, Pastry[10]. Nội dung dưới đây sẽ mô tả giao thức Chord để làm rõ cho hệ thống P2P có cấu trúc.

Giao thức Chord

Chord là một giao thức tìm kiếm phân tán sử dụng mô hình dạng vòng để kết nối các node với nhau. Giao thức này nằm trong hệ thống phân tán có cấu trúc, sử dụng bảng băm phân tán DHT để xác định các cặp khóa (key, value) phục vụ cho việc tra cứu, tìm kiếm trong mạng[8]. Hình 1-4 mô tả các node được xếp thành hình vòng tròn và sơ đồ kết nối giữa các node với nhau trong mạng Chord.



Hình 1-4 Mô hình mạng sử dụng giao thức Chord (mạng Chord)

Chord được biểu diễn dưới dạng vòng tròn, với vòng tròn có N bít sẽ có 2^n không gian định danh, mỗi node có một node liền trước (successor) và 1 node liền sau (predecessor), các node định tuyến cho nhau thông qua bảng định tyến (finger table). Mỗi dòng trong bảng định tuyến sẽ lưu thông tin một node ở xa gọi là entry. Bảng định tuyến được xác định dựa trên số bit đưa vào hệ thống, với n bit sẽ có n entry trong bảng định tuyến.

Một mạng theo giao thức Chord nếu sử dụng 4 bit cho việc định danh khóa sẽ có $2^4 = 16$ khóa (ID) và mỗi node chứa bảng định tuyến với 4 entry.

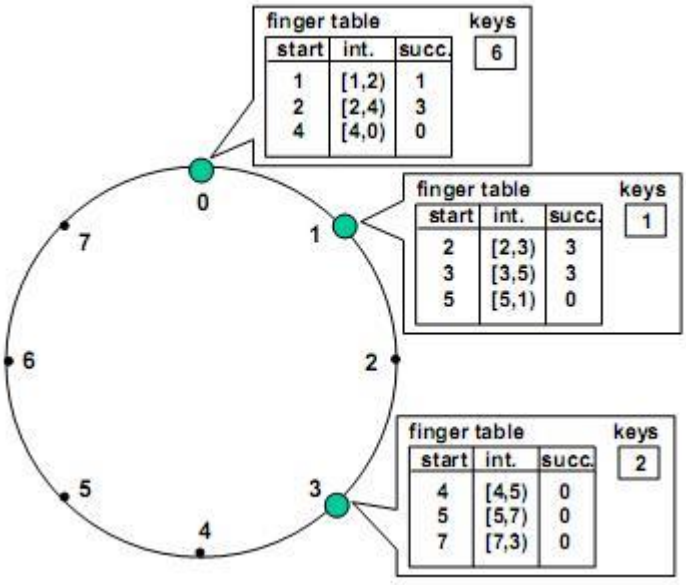
Thông tin entry trong bảng định tuyến bao gồm:

Ký hiệu	**Định nghĩa**	**Diễn giải**
finger[k].start	$(n+2^{k-1}) \bmod 2^m$, $1 \le k \le m$	Bước nhảy định tuyến tới một node trong mạng Chord
.interval	[finger[k].start, finger[k+1].start)	Khoảng không gian ID cho một bước nhảy trong mạng Chord
.node		Một node tham gia vào mạng Chord
successor		Node liền trước
predecessor		Node liền sau

Bảng 1-1 Bảng finger table [7]

Trong đó:

- k là entry thứ k của node n.
- n: Vị trí ID node n;
- m: Số bít cho định danh



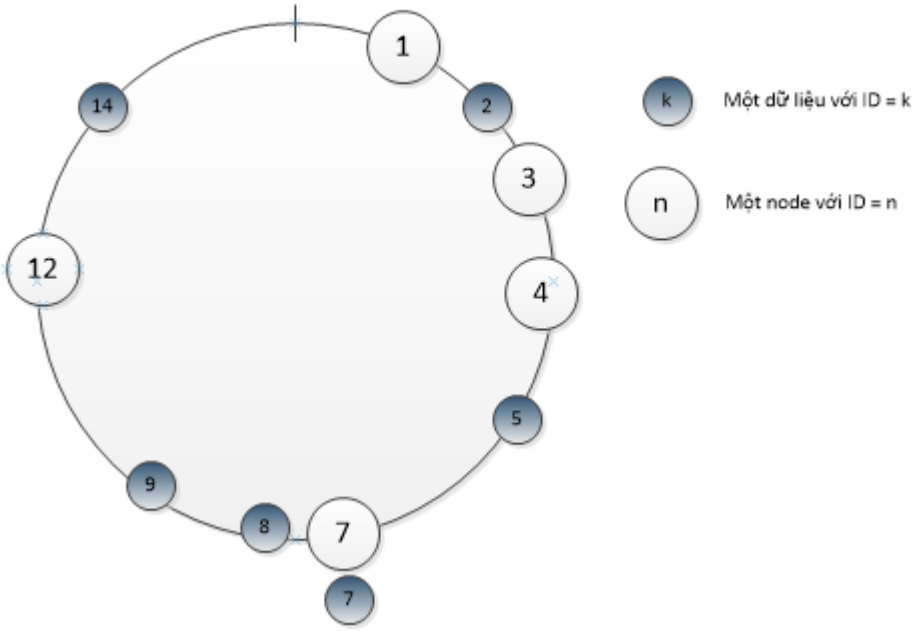
Hình 1-5 Bảng định tuyến với không gian định danh ID=8 và 3 node trong mạng (0,1,3)

Từ hình 1-5 cho thấy:

Node 0 tại entry thứ nhất có finger(1).start = $(0+2^{1-1})$ mod 2^4 = 1; finger(2).start = $(0+2^1)$ mod 2^2 = 2, do đó interval entry trong khoảng [1,2). Tương tự, finger(3).start = $(0+2^2)$ mod 2^4 = 4, do đó tại entry thứ hai interval trong khoảng [2,4).

Trong entry thứ nhất của node 0 có interval trong khoảng [1,2) trong khoảng này có node 1, nên node 1 là succsessor của node 0. Trong entry thứ hai của node 0 có interval trong khoảng [2,4) có node 3 nên node 3 là successor trong khoảng này.

Về phương pháp lưu trữ key trong mạng như sau: Mỗi file dữ liệu *(địa chỉ, file văn bản, âm thanh, hình ảnh ...)* khi đưa vào hệ thống sẽ sinh ra một key và tạo ra cặp (key, value), key được sinh ra được gán một định danh (ID) trong mạng Chord và successor node quản lý key này [3].



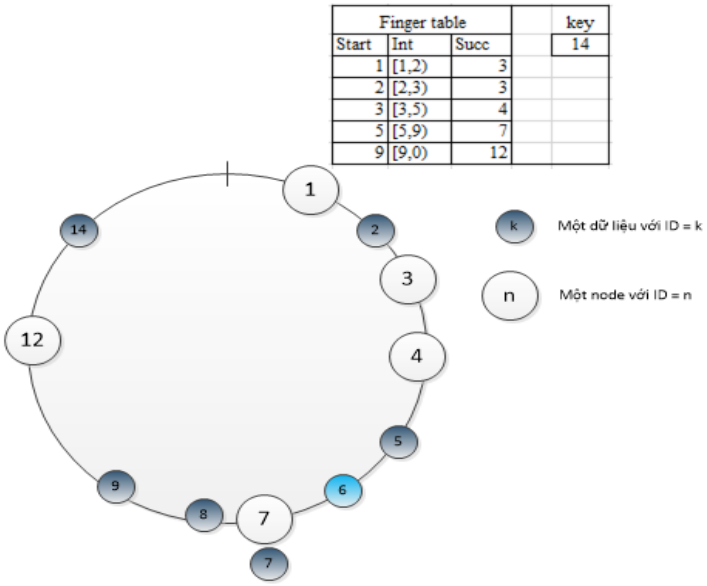
Hình 1-6 Mạng Chord với 5 node và 6 key

Hình 1-6 mô tả mạng Chord với không gian ID = 16 biểu diễn 5 node và 6 key trong mạng. Trong đó, node 1 lưu trữ key (14). Node 3 lưu trữ key (2). Node 4 không lưu trữ key nào. Node 7 lưu trữ key (5,7). Node 12 lưu trữ key (8,9)

Quá trình tìm kiếm

Khi một node muốn tìm kiếm một khóa có định danh ID, nó tìm node chịu trách nhiệm lưu trữ ID đó, nhờ vào bảng định tuyến để đến các node xa hơn, dần dần tìm đến node chịu trách nhiệm quản lý ID đó [7].

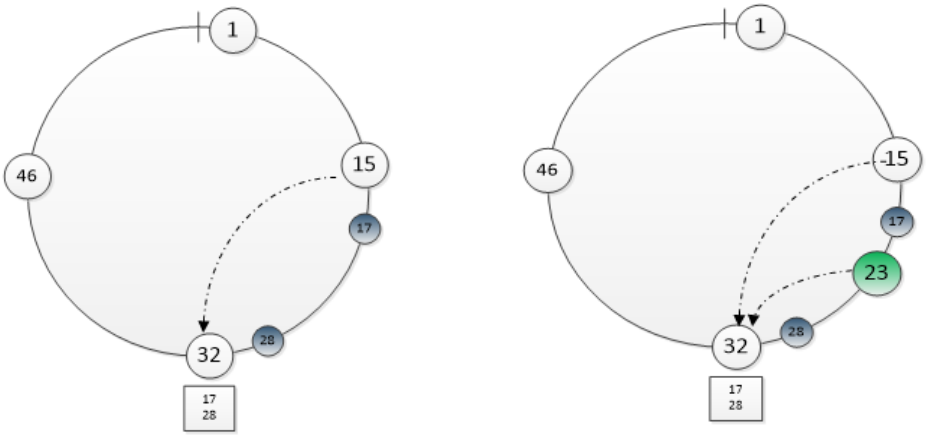
Trong hình 1-7, node 1 muốn chèn một dữ liệu với khóa là 6 nó sẽ tìm trong bảng định tuyến của nó thấy rằng trong bảng định tuyến của nó, successor node 1 là 3 nên nó định tuyến đến node 3, node 3 có successor là 4 nên nó tiếp tục chuyển tiếp tìm kiếm cho node 4. Node 4 thấy rằng 6 nằm giữa nó và successor của nó là 7 nên trả kết quả cho node 7. Sau khi nhận được câu trả lời, tầng ứng dụng trên node 1 và node 7 yêu cầu lưu trữ một số với giá trị key là 6. Các node muốn tìm kiếm key 6 đều thực hiện các quá trình tương tự nhưng không quá số entry trong bảng định tuyến. Thông thường một tìm kiếm sẽ hoàn thành trong $O(\log_2(N))$ chặng.

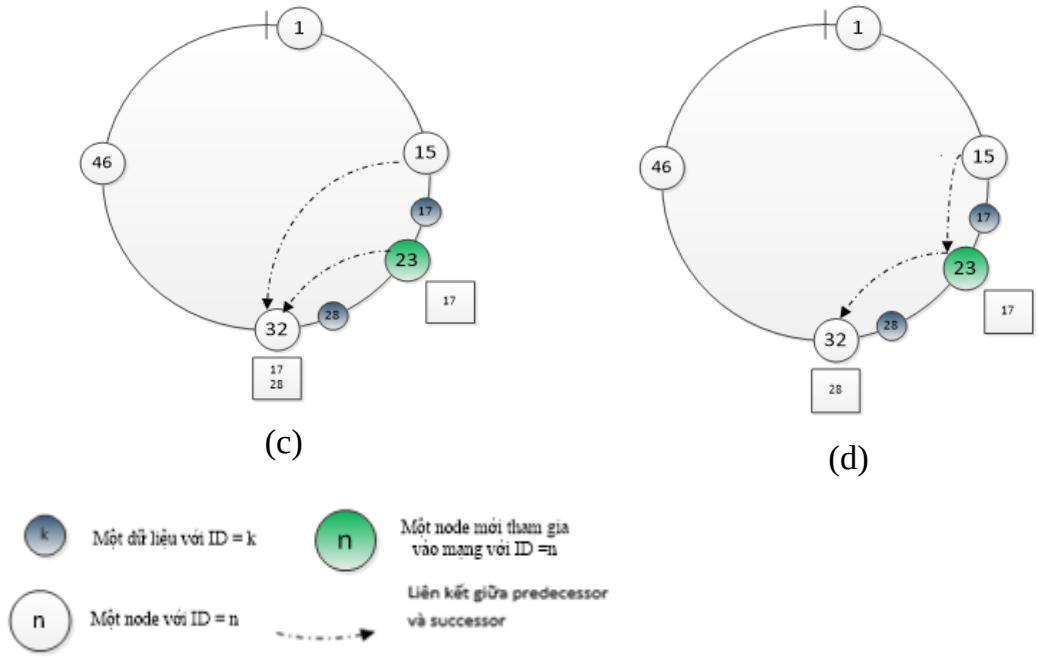


Hình 1-7 Quá trình tìm kiếm khóa của một node

Quá trình tham gia, rời mạng và duy trì dữ liệu

Khi một node tham gia vào mạng, thông qua bảng băm DHT nó sẽ sinh ra một ID. Node mới tham gia khi này sẽ nằm ở giữa node predecesor và successor trước đó, do đó nó phải cập nhật lại thông tin bảng định tuyến cũng như thông tin quản lý khóa trong không gian ID của nó [9]. Bảng định tuyến được copy từ bảng định tuyến của node successor, thông qua thuật toán định kỳ nó sẽ cập nhật lại thông tin bảng định tuyến. Việc di chuyển khóa được cập nhật bằng cách node successor kiểm tra xem có đang quản lý khóa nào nhỏ hơn ID của node mới tham gia không, nếu nhỏ hơn thì chuyển khóa đó cho node mới tham gia quản lý.





Hình 1-8 Mô tả các bước tham gia mạng của một node

Hình 1-8 là một ví dụ mô tả quá trình tham gia mạng của một node

Giả sử 15 có successor là 32, node 32 quản lý key 17 và 28 *(hình a)*. Khi node mới với ID = 23 tham gia vào mạng, sau quá trình tìm kiếm node 23 biết 32 là successor, nó trỏ successor vào node 32 và báo cho node 32 biết *(hình b)*.

Node 32 sau đó trỏ predecessor vào node 23, node 23 copy các key nhỏ hơn ID của nó (17) được lưu trữ trong node 32 *(hình c)*. Định kỳ cập nhật thông tin node 15 hỏi node 32 về predecessor của node 32 và biết được là node 23, nó cập nhật là thông tin successor của nó và báo cho node 23 biết, node 23 cập nhật predecessor của nó trỏ vào node 15 *(hình d)*.

Nhân bản và khả năng chịu lỗi

Trong trường hợp các node rời mạng đột ngột hoặc các node liền nhau cùng rời khỏi mạng đột ngột, dẫn đến các dữ liệu trên các node này bị mất và các liên kết để định tuyến đến node cũng như việc tìm kiếm dữ liệu thông qua key sẽ không được tìm thấy. Chord giải quyết bằng cách cho mỗi node lưu một danh sách $\log_2 N$ node theo sau nó trong không gian ID nhằm mục đích [3]:

- Nếu một node phát hiện successor của nó không hoạt động nó sẽ thay thế bằng node ngay cạch trong danh sách các successor node của nó.
- Dữ liệu chỉ bị mất hay liên kết vòng bị ngắt khi danh dách $\log_2 N$ node rời mạng đồng thời.

1.3 Hệ thống P2P hỗn hợp

Ưu điểm chính của hệ thống tập trung là có thể cung cấp nhanh và định vị tài nguyên tin cậy, tuy nhiên hệ thống này bị giới hạn tính mở rộng do hạn chế của việc sử dụng các server. Trái lại, hệ thống P2P phân tán lại tốt hơn hệ thống tập trung ở tính mở rộng, nhưng chúng yêu cầu một thời gian lâu hơn để xác định vị trí tài nguyên.

Hệ thống P2P hỗn hợp tận dụng được các ưu điểm so với hệ thống phân tán và hệ thống tập trung. Trong hệ thống P2P hỗn hợp có một số peer xử lý nhiều chức năng hơn và chịu trách nhiệm nhiều hơn các peer khác còn gọi là supernode[13]. Mặc dù các supernode thực hiện một vài chức năng của server trung tâm nhưng nó cũng có một vài điểm khác biệt như sau:

- Một supernode không mạnh bằng server trung tâm và chỉ chịu trách nhiệm quản lý các peer trong mạng.
- Một server như mô hình Napster giúp đỡ các peer để định vị các file mà không chia sẻ file. Tuy nhiên một supernode không chỉ phối hợp các hoạt động trong peer, mà chính nó còn thực hiện các hoạt động tương tự và đóng góp tài nguyên của nó như là các peer thông thường khác. Hình 1-9 mô tả mô hình mạng hỗn hợp với các suppernode trung tâm và các client kết nối với các suppernode.

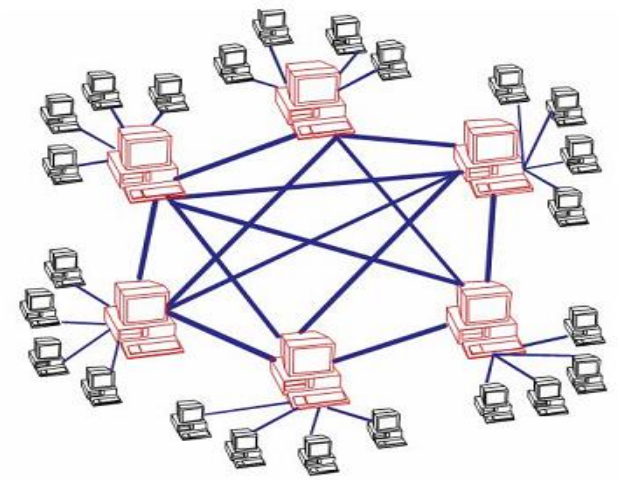
Điển hình của hệ thống P2P hỗn hợp là mô hình hệ thống Bestpeer, dưới đây sẽ mô tả rõ hơn về mô hình mạng này:

BestPeer

BestPeer được thiết kế như là một nền tảng chung cho việc phát triển các ứng dụng P2P và có bốn chức năng sau:

- Hệ thống Bestpeer cho phép các peer có thể thực thi các hoạt động ngay trong peer, có nghĩa là các dữ liệu thô có thể được xử lý trực tiếp tại node đó. Vì vậy hệ thống tối ưu được hiệu quả băng thông mạng, hơn nữa nó cho phép tùy chỉnh, các ứng dụng mới có thể được mở rộng dễ dàng với BestPeer.
- Cho phép các peer trong hệ thống không chỉ chia sẻ dữ liệu mà còn tính toán tài nguyên, xử lý các yêu cầu thay cho các peer khác.
- Sử dụng phương pháp động, cho phép các peer duy trì với các peer khác có băng thông cao, khả năng trả lời nhanh với các yêu cầu.

- Sử dụng server LIGLO (location independent global names lookup) để xác định được các peer độc lập với địa chỉ IP của nó. Như vậy, mỗi lần một peer tham gia vào hệ thống và thay đổi địa chỉ IP, thì nó vẫn nhận ra đó là một peer duy nhất. Mô hình hệ thống Bestpeer được thể hiện qua hình 1-4.



Hình 1-9 Mô hình hệ thống P2P hỗn hợp Bestpeer.

Những thuộc tính và giới hạn của hệ thống P2P hỗn hợp như sau:

Khả năng chị lỗi: Nhờ sử dụng các suppernode *(với Bestpeer là các server LIGLO),* giúp hệ thống tránh được lỗi gây ra tại một điểm và có khả năng chịu lỗi cao.

Tính bảo mật: Nhờ có sự kết hợp giữa công nghệ agent-base và công nghệ tính toán P2P, hệ thống có thể chia sẻ hoặc ngăn chặn các thông tin bảo mật, riêng tư trong quá trình trao đổi giữa các peer.

Tính mở rộng: Bằng cách sử dụng công nghệ mobile agent, khả năng mở rộng tốt hơn so với hệ thống P2P tập trung. Khi mobile agent thực thi hoạt động cục bộ tại các peer, agent có thể tùy chỉnh với mục đích khác nhau kết quả là nhiều ứng dụng có thể được thực thi dễ dàng.

Tự cấu hình lại: Hệ thống P2P hỗn hợp có khả năng tự cấu hình lại, thông qua hai cách tiếp cận Maxcount và MinHop. Maxcount được sử dụng để đảm bảo rằng, một peer liên kết với các peer lân cận có thể chứa đựng số lượng đối tượng lớn nhất. MinHop là cách kết nối đến các peer mà có số hop nhỏ nhất.

Mạng P2P hỗn hợp cũng có một số nhược điểm như, chi phí cho các supernode cao hơn các peer trong mạng, có thể bị tấn công botnet vào các supper node làm ảnh hưởng toàn bộ hệ thống [5].

CHƯƠNG 2: CÁC PHƯƠNG PHÁP BACKUP DỮ LIỆU TRÊN MẠNG NGANG HÀNG CÓ CẤU TRÚC

Trong chương này mô tả hai phương pháp backup khác nhau, đó là phương pháp successor list và phân cụm tĩnh, sử dụng giao thức Chord. Giao thức Chord là giao thức chính được sử dụng cho nghiên cứu, cải tiến của luận văn thông qua việc mở rộng phân cụm động và phân cụm tĩnh so với phương pháp Chord nguyên thủy (successor list).

2.1 Cơ chế backup theo successor list

Backup theo successor list là phương pháp backup nguyên thủy trong mạng Chord. Dựa trên bảng băm phân tán DHT, sử dụng mã xóa IDA (information dispersal algorithm) [9], nhằm phân tán và lưu trữ khối với bảng băm.

DHT là một hàm băm được cài đặt như một hệ thống phân tán. Cũng như một hàm băm thông thường, DHT cung cấp ánh xạ từ key đến value. Điểm khác của DHT so với hàm băm thông thường là các value trong một DHT được lưu đến các node khác nhau trong mạng, chứ không phải lưu trong một cấu trúc dữ liệu cục bộ. Nhờ khả năng phân tán làm cho DHT trở nên mạnh mẽ, hiệu quả và đáp ứng được những ứng dụng thực tế. Các ứng dụng này yêu cầu DHT duy trì tính sẵn sàng của dữ liệu ngay cả khi gặp lỗi và hiệu quả trong trường hợp xử lý một khối dữ liệu lớn.

Theo phương pháp mã xóa, bảng băm chia khối dữ liệu (file dữ liệu) ra làm *f* mảnh, trong đó với *k* mảnh là có thể khôi phục lại được khối dữ liệu. Các mảnh dữ liệu ở đây là riêng biệt nhau, chứa thông tin độc lập và duy nhất. Chẳng hạn, khối dữ liệu chia ra làm 14 mảnh, nhưng với 7 mảnh dữ liệu có thể khôi phục lại khối dữ liệu 14 mảnh. Để duy trì các mảnh dữ liệu luôn đảm bảo có thể khôi phục được khối dữ liệu ban đầu, DHT chuyển đổi các mảnh giữa các node khi các node tham gia hoặc rời mạng.

- **Đảm bảo tính sẵn sàng khối dữ liệu**

Giống như khả năng chịu lỗi của nhiều hệ thống lưu trữ khác, bảng băm sử dụng mã xóa để làm tăng tính sẵn sàng với chi phí thấp.

Khi thêm khối dữ liệu: put (k,b)[6]

Khi một ứng dụng muốn thêm một khối dữ liệu mới, nó gọi hàm băm *put(k,b)* thực hiện như sau:

```
Void put (k,f) // đặt một mảnh vào mỗi successor
{
Frags=IDAencode (f)
Succs=lookup (k,14)
For i (0…13)
Send (succs[i].ipaddr, k, frags[i])
}
```

Lấy khối dữ liệu: get(k)

Để lấy khối dữ liệu, một client phải định vị và truy hồi dủ các mảnh theo thuật toán phân mảnh thông tin IDA để lắp ghép lại thành khối dữ liệu ban đầu. Khi một ứng dụng client gọi *get(k)* bảng băm tại client trước tiên khởi tạo việc tìm kiếm qua hàm *lookup(k,7*) để tìm danh sách các node có khả năng lưu trữ các mảnh của khối dữ liệu. Kết quả tìm kiếm sẽ trả về danh sách từ 7 đến 14 node successor trực tiếp của khóa k.

Sau đó *get()* chọn 7 successor với độ trễ thấp nhất để thiết lập đồng bộ, gửi mỗi node một RPC (remote procedure call) để yêu cầu một mảnh của khóa k theo phương pháp đồng bộ song song. Với mỗi RPC quá thời gian cho phép *(time out)* hoặc bị lỗi, *get()* gửi một mảnh yêu cầu RPC để kết nối lại với danh sách các successor tìm thấy qua hàm lookup() mà chưa kết nối để thiết lập kết nối lại [6].

Trong trường hợp gọi hàm *lookup()* nhưng kết quả trả về ít hơn 7 successor chứa các mảnh, *get()* hỏi một trong successor của nó tìm kiếm mở rộng thêm các node khác để tạo lại khối dữ liệu. Nếu không thể xây dựng lại được khối dữ liệu sau quá trình trao đổi, tìm kiếm trên hệ thống, get() trả lại kết quả không thành công.

Một ứng dụng có thể gọi hàm *get(k)* nhiều lần để lấy khóa cho sẵn. Khi các node tham gia hoặc rời hệ thống, các mảnh cần phải chuyển đến các successor node của nó. Nếu tỷ lệ tham gia, rời mạng của các node tăng cao có thể dẫn đến mảnh dữ liệu bị sai vị trí và dẫn đến việc không lấy được mảnh bị mất. Để khắc phục việc này, bảng băm đưa ra cơ chế duy trì mảnh dữ liệu để phục hồi lại các mảnh đã bị mất.

```
block get (k)
 {
    //Chọn các mảnh từ các successors
     frags = []; //Mảng trống
     succs = lookup (k,7)
     sort_by_latency (succs)
     for ( i = 0; i < succs && i <14; i++ )
          {
           //Tải các mảnh dữ liệu
           <ret, data> = download (key, succ[i])
```

```
        if(ret ==OK)
            frags.push (data)
        // Giải mã các mảnh để khôi phục lại khối dữ liệu
            <ret, block> = IDAdecode (frags)
            if (ret == OK)
                return (SHA-1(block) != k) ? FAILURE : block
            if (i == succs – 1)
                  {
                    newsuccs = get_successor_list ( succs [i] )
                    sort_by_latency (newsuccs)
                    succs.append (newsuccs)
                   }
          }
      return FAILURE
}
```

Hình 2-1 Thủ tục thực hiện hàm get(k)

- **Duy trì mảnh dữ liệu**

Trạng thái lý tưởng khi tồn tại đủ số mảnh của khối dữ liệu, tuy nhiên các node tham gia và rời mạng liên tục dẫn tới bị lỗi ở một số node làm cho các mảnh bị mất hoặc bị đặt sai vị trí. Để duy trì trạng thái lý tưởng, bảng băm sử dụng hai giao thức là giao thức duy trì cục bộ và giao thức duy trì toàn cục.

Giao thức duy trì cục bộ phục hồi số mảnh còn thiếu, còn giao thức toàn cục di chuyển các mảnh đặt sai vị trí vào vào đúng số node, khôi phục lại vị trí, ngoài ra nó cũng có chức năng phục hồi các mảnh đã bị mất, xóa đi các mảnh dữ liệu dư thừa trong hệ thống. Chức năng của hai giao thức này được thể hiện qua thuật toán tại hình 2-2 và 2-3 bên dưới [6].

```
global_maintenance (void)
  {
    a= myID
    while (1)
       {
        <key, frags> = database.next(a)
        succs = lookup (key, 16)
        if ( myID isbetween succ[0] and succ [15] )
        // Lưu lại key
        a = myID
        else {
              //Key bị sai vị trí
               For each s in succs [0..13] {
               Response = send_db_keys (s, database [key ….succs[0])
               For each key in response.desired_keys
               if (database.contains (key))
               upload (s, database.lookup (key))
               database.delete (key)
               }
```

```
        database.delete_range ([pred...succs[0])
        a = succs[0]
       }
```

Hình 2-2 Thủ tục của giao thức duy trì toàn cục

```
local_maintenance (void)
   {
    while (1) {
                foreach ( s in mySuccessors [0..12] )
                synchronize ( s, database [mypredID … myID] )
               }
    }

missing (key)
    {
      //Khi 'key' không tồn tại trong máy chủ chạy đoạn mã này
       block = get (key)
       frag = IDA_get_random_fragment (block)
       merkle_tree.insert (key, frag)
     }
```

Hình 2-3 Thủ tục giao thức duy trì cục bộ

Mặc dù hệ thống DHT cung cấp khả năng chịu lỗi và khả năng mở rộng, tuy nhiên các nghiên cứu gần đây chỉ ra rằng khi tần suất các node tham gia hoặc vào mạng quá cao, các node liền kề bị quá tải và không backup được, khối dữ liệu có thể bị mất [11]. Để tránh điều này trong các nội dung tiếp theo chúng tôi đã đề xuất phương pháp phân cụm, phân mảnh dữ liệu để đảm bảo quá trình tham gia, rời mạng của các node với tần suất cao có thể duy trì backup dữ liệu ở mức ổn định.

2.2 Phân cụm tĩnh trong mạng Chord

2.2.1 Phương pháp tách cụm tĩnh

Ý tưởng chính của phương pháp là chia mạng Chord thành một số cụm với không gian ID mỗi cụm bằng nhau dựa vào bảng băm phân tán DHT, mỗi cụm sẽ lưu trữ cục bộ và thực hiện duy trì dữ liệu trong cụm, đảm bảo dữ liệu luôn sẵn sàng ngay cả khi các node tham gia hoặc rời mạng.

Trong mỗi cụm đưa ra một số node có khả năng lưu trữ với dung lượng lớn để đảm bảo việc backup dữ liệu luôn được cân bằng tải giữa các node trong một cụm.

2.2.2 Phương pháp backup file

Để duy trì tính sẵn sàng của file dữ liệu ngay cả có sự vào ra của các node trong mạng, mỗi file được mã hóa thành *n* mảnh sử dụng hình thức mã xóa [6,14], các mảnh này được lưu ở một số node trong mạng. Đặc trưng của mã xóa là với *k* mảnh của file *(k<n)* được tập hợp thì có thể khôi phục lại file ban đầu. Ở đây k và n được định nghĩa trước trong hệ thống. Phương pháp backup dựa trên việc phân cụm là các mảnh dữ liệu của file được phân bố vào các node trong một cụm, như vậy trong một cụm luôn duy trì số mảnh của một file lớn hơn k nhằm phục hồi lại dữ liệu file[4].

2.2.2.1 Quản lý thông tin cụm

Không gian khóa DHT được chia thành m phần bằng nhau (m cụm), biên của cụm thứ k sẽ được lưu ở node đầu và node cuối cụm, các node có định danh ID nằm ở giữa định danh đầu cụm và định danh cuối cụm thì thuộc cụm đó. Như vậy trong mỗi cụm có một node đầu cụm, một node cuối cụm. Node cuối cụm này nhưng cũng là đầu cụm kế tiếp.

Các node trong cùng một cụm sẽ trao đổi, cập nhật thông tin cho nhau và cập nhật thông tin cho cả cụm. Thông tin của một cụm được gửi đi trong một lần cập nhật bao gồm:

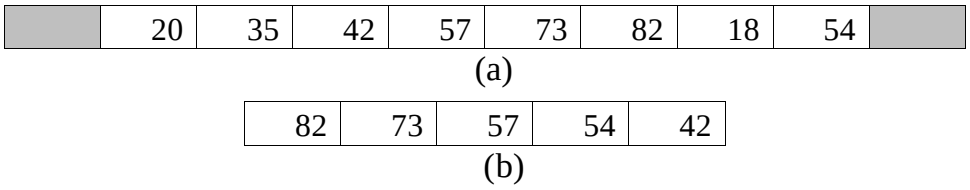
- Dải định danh ID của cụm
- Thông tin của nút đầu tiên của cụm
- Danh sách các nút có dung lượng lưu trữ lớn. Trong đó, dung lượng lưu trữ của một node là số lượng dữ liệu của node đó có khả năng lưu trữ.
- Danh sách các nút rời mạng trong một khoảng thời gian định kỳ.

Node đầu cụm khởi tạo quá trình cập nhật bằng cách gửi thông điệp cập nhật tới successor node của nó. Successor node cập nhật thông điệp của nó và gửi thông điệp tới node tiếp theo, cứ tiếp tục như vậy khi node cuối cùng của cụm nhận được thông điệp, nó sẽ gửi thông điệp trở lại node đầu.

Khi một thông điệp cập nhật được gửi tới một node, node đó sẽ kiểm tra khả năng lưu trữ trong danh sách các node lưu trữ tốt nhất. Nếu khả năng lưu trữ của nó lơn hơn bất kỳ node nào trong danh sách nó sẽ chèn thông tin của nó vào

danh sách, bao gồm địa chỉ và khả năng lưu trữ của nó và loại đi thông tin của node có khả năng lưu trữ thấp nhất trong danh sách.

Danh sách các node lưu trữ tốt nhất trong cụm được sử dụng trong việc chọn ra các node lưu trữ các mảnh dữ liệu. Bằng cách sử dụng danh sách các node tốt nhất trong cụm chúng ta có thể tránh vấn đề cân bằng tải giữa các node trong một cụm.



Hình 2-4. Hình a mô tả 8 node trong một cụm với khả năng lưu trữ (20,35,42,57,73,82,18,54). Hình b mô tả danh sách 5 node có dung lượng lưu trữ lớn được lấy ra từ hình a.

Khi một node trong cụm tham gia vào mạng, nó sẽ nhận thông tin cụm từ node successor của nó về danh sách danh sách các node lưu trữ tốt nhất trong cụm.

Cập nhật thông điệp sẽ gửi định kỳ, tuy nhiên khi số node rời mạng lớn hơn giá trị ngưỡng thì node đầu cụm sẽ gửi một thông điệp cập nhật ngay lập tức mà không cần đợi kết thúc chu kỳ cập nhật.

2.2.2.2 Truy vấn và sao lưu dữ liệu

Trong phương pháp backup được đưa ra, một node chịu trách nhiệm về khóa của một file DHT sẽ quản lý tính sẵn có của file đó *(quản lý file đó còn tồn tại hay không và thông tin lưu trữ các mảnh của file)*. Khóa của một file DHT là khóa duy nhất sinh ra từ việc băm nội dung của file và được sử dụng để truy vấn phục hồi một file. Một quá trình sao lưu dữ liệu được thực hiện qua các bước sau:

Bước 1: Khi một node có một file dữ liệu mới đưa vào (node nguồn), trước tiên nó tạo khóa của file thông qua DHT, và gửi thông điệp yêu cầu sao lưu tới node chịu trách nhiệm với khóa DHT.

Bước 2: Node chịu trách nhiệm gửi danh sách các node lưu trữ tốt nhất trong cụm tới node nguồn.

Bước 3: Sau khi nhận danh sách các nút lưu trữ tốt nhất, node nguồn sẽ lựa chọn ngẫu nhiên một số node để sao lưu từ danh sách, sau đó tạo ra các mảnh dữ liệu từ file bằng cách sử dụng mã xóa và gửi các mảnh dữ liệu tới các node sao lưu để lưu trữ dữ liệu. Ngoài ra khóa DHT và địa chỉ node chịu trách nhiệm với khóa DHT của file cũng gửi tới các node sao lưu. Nếu một node không chấp nhận mảnh dữ liệu nó sẽ thông báo tới node nguồn và node nguồn sẽ lựa chọn

node khác để sao lưu. Khi quá trình sao lưu kết thúc, node nguồn thông báo tới node chịu trách nhiệm với khóa chính về danh sách các node mà đã lưu trữ các mảnh dữ liệu.

Bước 4: Node chịu trách nhiệm về khóa DHT của file lưu vào cơ sở dữ liệu của nó, và thông báo việc sao lưu file tới node nguồn và danh sách các node sao lưu.

Việc lựa chon ngẫu nhiên các node sao lưu từ danh sách các node lưu trữ tốt nhất, nhằm tránh quá tải cho việc sao lưu. Khi một node muốn truy vấn file sao lưu, nó sẽ gửi thông điệp truy vấn tới node chịu trách nhiệm khóa của file DHT. Node chịu trách nhiệm sẽ tra cứu cơ sở dữ liệu của nó về khóa DHT và gửi trở lại địa chỉ node nguồn và danh sách các node sao lưu file. Node truy vấn lựa chọn ngẫu nhiên các node sao lưu và gửi thông điệp truy vấn tới các node này cho tới khi đủ k mảnh để khôi phục lại toàn bộ file ban đầu.

2.2.2.3 Duy trì tính ổn định file

Khi một node rời mạng chủ động, nó sẽ chuyển dữ liệu lưu trữ của nó cho successor node và gửi thông tin thông báo tới node đầu cụm về trạng thái rời mạng. Tuy nhiên, nếu một node rời mạng do bị lỗi (rời mạng đột ngột), dữ liệu bao gồm mảnh dữ liệu và thông tin về các file sao lưu được lưu trong node đó sẽ bị mất, trong trường hợp này chúng ta cần duy trì ít nhất k mảnh của bất kỳ file nào trong mạng để đảm bảo file luôn sẵn sàng. Hơn nữa, thông tin của file sao lưu cần được phục hồi để cho phép truy vấn. Trong trường hợp này successor node sẽ dò tìm node rời dựa trên cơ chế của thuật toán DHT. Successor node sau đó gửi một thông điệp cập nhật tới node đầu cụm để thông báo về trạng thái rời mạng.

Node đầu cụm sẽ gửi một thông điệp cập nhật bao gồm danh sách các node rời mạng theo chu kỳ hoặc ngay lập tức tùy thuộc vào số node rời mạng trong một chu kỳ.

Khi một node nhận được thông điệp cập nhật. Nó kiểm tra danh sách các node rời mạng có quản lý các mảnh dữ liệu mà node cập nhật đang lưu trữ key file của các mảnh không:

Nếu một node rời mạng có lưu các mảnh thì node kiểm tra cập nhật sẽ tính lại số mảnh dữ liệu còn lại mà key của một file quản lý mảnh đó.

Nếu số mảnh dữ liệu của một file nhỏ hơn ngưỡng giá trị k *(k là số mảnh tối thiểu để có thể backup được thành 1 file gốc),* node chịu trách nhiệm sẽ thực hiện backup dữ liệu để phục hồi các mảnh bị mất. Để làm được điều này, node

gửi một yêu cầu sao lưu, danh sách các mảnh bị mất và danh sách các node lưu trữ tốt nhất trong cụm tới node nguồn (node chứa file gốc):

Nếu node nguồn vẫn tồn tại trong mạng, nó sẽ tạo ra các mảnh bị mất từ file gốc và gửi các mảnh này tới các node sau lưu với việc lựa chọn ngẫu nhiên từ danh sách các node lưu trữ tốt nhất.

Nếu node nguồn rời mạng, node chịu trách nhiệm cho khóa chính của file sẽ truy vấn các mảnh dữ liệu từ các node sao lưu, xây dựng lại file và tạo ra các mảnh sao lưu bị mất, sau đó gửi các mảnh này tới các node lưu trữ tốt nhất để lưu trữ và cập nhật danh sách các node sao lưu.

2.3 Kết luận

Phương pháp phân cụm tĩnh được đánh giá về tỷ lệ phục file ban đầu thành công cao hơn so với phương pháp successor list của giao thức Chord nguyên thủy, đồng thời chi phí duy trì các mảnh dữ liệu lại thấp hơn [4], từ đó cho thấy hiệu quả của việc phân cụm tĩnh khá rõ rệt, dễ thấy nhất đó là thông qua việc phân cụm thời gian cập nhật thông tin trong một cụm nhanh hơn so với việc truy vấn, cập nhật thông tin cho toàn mạng, ngoài ra việc áp dụng phương pháp phân mảnh và sử dụng mã xóa cũng làm giảm việc khôi phục toàn bộ các mảnh của một file ban đầu mà chỉ cần k mảnh.

Mặc dù phương pháp phân cụm tĩnh mang lại hiệu quả nhất định so với phương pháp successor list tuy nhiên phương pháp này vẫn còn một vài hạn chế như:

- Xác suất nhiều node rời mạng trước khi thực hiện việc duy trì là lớn
- Không cập nhật được danh sách các node tốt nhất

Từ những vấn đề của phương pháp successor list và phân cụm tĩnh, luận văn đưa thêm phương pháp phân cụm động, là phương pháp cố định số node trong một cụm và linh động số cụm để đảm bảo quá trình cập nhật thông tin trong một cụm luôn ở mức ổn định, giảm thời gian backup dữ liệu, tăng tỷ lệ phục hồi thành công file. Nội dung phân cụm động được trình bày ở các chương tiếp theo.

CHƯƠNG 3: PHƯƠNG PHÁP PHÂN CỤM ĐỘNG VÀ CƠ CHẾ BACKUP.

Ý tưởng chính của phương pháp này là chia cụm và giới hạn số node trong một cụm, để đảm bảo quá trình backup định kỳ thường xuyên và ổn định hơn. Bên cạnh đó, phương pháp phân cụm động cũng xử lý các cụm liền kề mà có số node trong mỗi cụm nhỏ, có thể nhập cụm lại nhằm đảm bảo cân bằng tải cho các node thông qua việc làm giảm số lượng cụm, ổn định số node trong cụm và làm cho cân bằng tải trong cụm tốt hơn, tỷ lệ phục hồi file thành công cao hơn.

3.1 Nguyên tắc chung

Nguyên tắc phân chia cụm và hoạt động trong mỗi cụm

Trong phương pháp phân cụm động chúng tôi đã đưa ra một số nguyên tắc phân chia cụm nhằm đảm bảo tính thống nhất hoạt động trong cả mạng và hoạt động trong mỗi cụm, bao gồm các nguyên tắc sau:

- Căn cứ số lượng các node trong cụm để xác định việc chia cụm hoặc nhập cụm
- Mỗi cụm sẽ có 1 node làm node đầu cụm có thông tin 2 cụm liền kề nhau
- Khi tách cụm thực hiện chia đôi không gian ID của cụm ban đầu thành hai cụm bằng nhau
- Khi nhập cụm thực hiện nhập 2 cụm liền kề nhau.

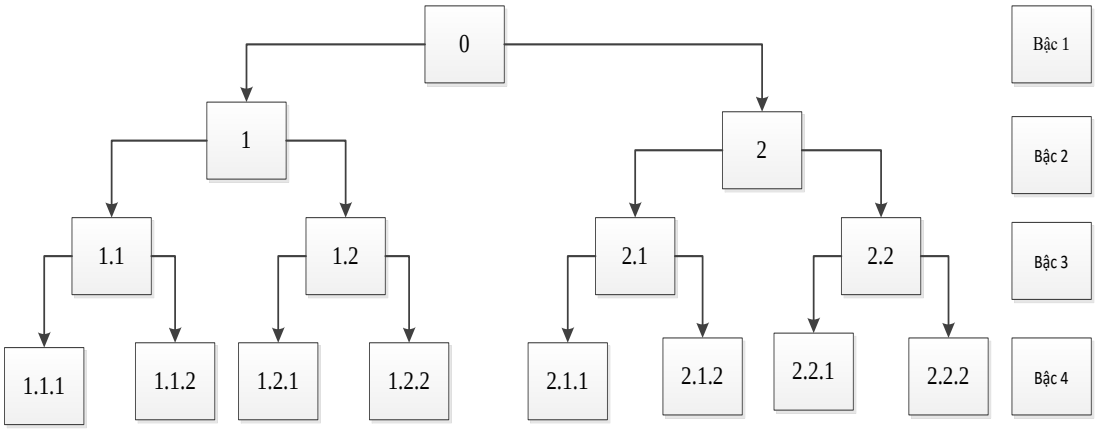
Hoạt động trong mỗi cụm bao gồm hoạt động tham gia hoặc rời mạng của các node trong cụm, do việc tham gia hoặc rời mạng của các node này ảnh hưởng tới việc duy trì và phục hồi dữ liệu trong cụm, nên phải có cơ chế định kỳ cập nhật thông tin và định kỳ backup dữ liệu để luôn đảm bảo tính sẵn sàng dữ liệu trong cụm.

Nguyên tắc đánh số cụm:

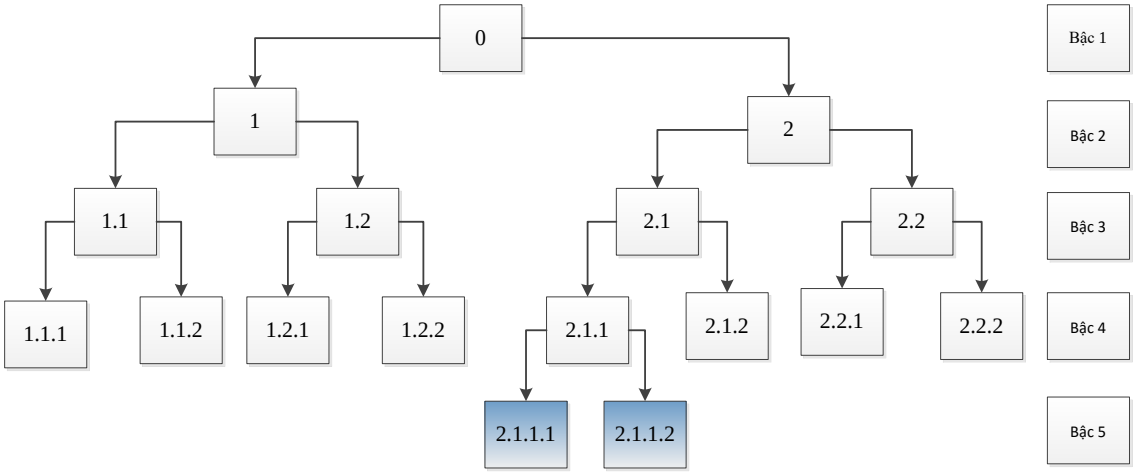
Khi chia tách, sát nhập cụm thông tin trong các cụm thay đổi, cần có cơ chế đánh số cụm để đảm bảo các cụm có thông tin với nhau, và xác định thông tin các cụm liền kề và các cụm lân cận nó. Nguyên tắc đánh số cụm như sau:

- Mỗi lần tách cụm: Mỗi cụm mới sẽ bổ sung thêm 1 giá trị vào cuối cụm. Theo chiều từ trái qua phải, cụm bên trái nhận giá trị 1, cụm bên phải nhận giá trị 2 (hình 3-2).
- Mỗi lần nhập cụm: Mỗi cụm sẽ bỏ đi 1 giá trị ở cuối cụm. Theo chiều từ trái qua phải, cụm bên trái nhận giá trị 1, cụm bên phải nhận giá trị 2 (hình 3-3).

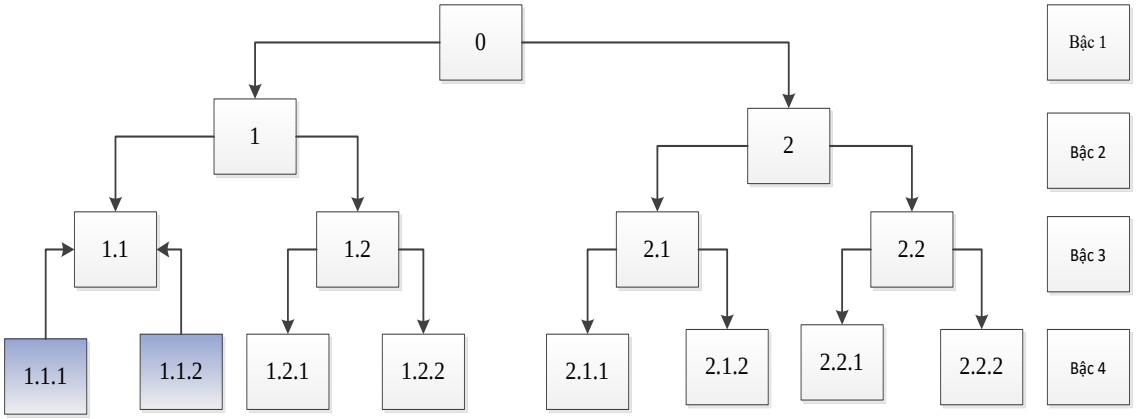
Bậc của cụm nhằm thể hiện số lần tách cụm ra từ cụm ban đầu, thông qua bậc của cụm và chỉ số cụm có thể xác định được thông tin hai cụm liền nhau. Nếu hai cụm cùng bậc và có chỉ số cụm giống nhau, chỉ khác nhau chỉ số cuối thì hai cụm liền kề nhau. Hình 3-1 mô tả cụm được chia tách với 4 bậc và mô tả quá trình đánh số các cụm qua các lần phân chia, các cụm được phân chia và đánh số theo hình thức cây nhị phân, các cụm đang tồn tại thể hiện ở các node lá.



Hình 3-1 Phương pháp đánh số cụm và phân bậc



Hình 3-2 Quá trình tách cụm 2.1.1 thành hai cụm 2.1.1.1 và 2.1.1.2

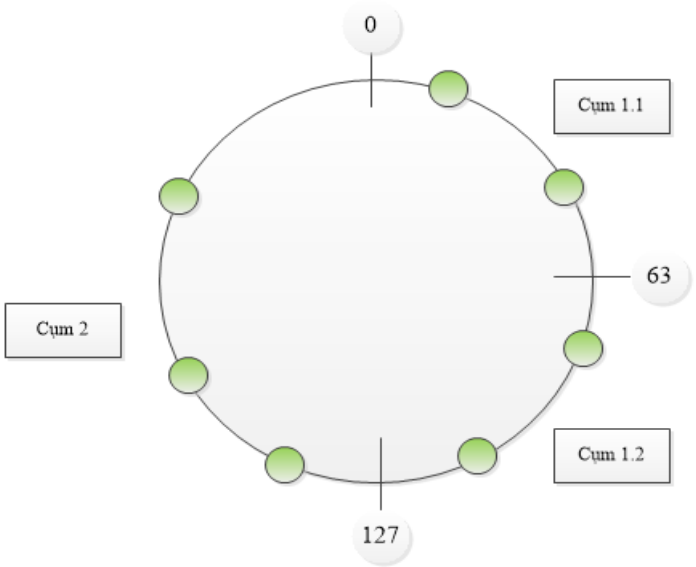


Hình 3-3 Quá trình nhập cụm 1.1.1 và 1.1.2 để thành cụm 1.1

Ví dụ: Giả sử mạng Chord lấy 8 bit làm không gian định danh ID cho các node và quy ước nếu số node trong cụm >=20 node thực hiện tách cụm. Nếu số node của 2 nhánh liền kề <=10 thực hiện việc nhập cụm.

Ban đầu có một node đầu cụm, cụm ban đầu được đánh số 0. Khi có 20 node tham gia vào mạng, node đầu cụm thực hiện việc tách thành 2 cụm theo phương pháp chia đôi không gian ID. Khi đó, không gian ID của cụm 1 từ 0-127 và không gian ID của cụm 2 từ 128 – 255.

Nếu cụm 1 có số node >=20, cụm 1 tiếp tục tách thành cụm 1.1 và 1.2. Khi đó, không gian ID của cụm 1.1 từ 0 – 63, không gian ID của cụm 1.2 từ 164 – 127 (hình 3-4).



Hình 3-4: Mạng chord với 3 cụm 1.1, 1.2 và 2

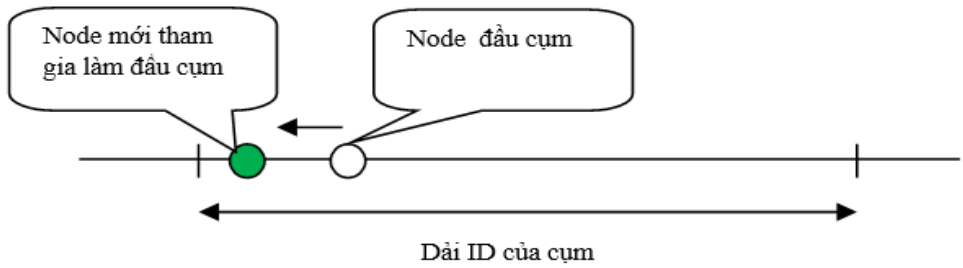
Trong quá trình tách cụm và chia đôi không gian ID, nếu các cụm được hình thành sau quá trình tách cụm vẫn vượt quá yêu cầu về số lượng node trong cụm (>=20), thì quá trình tách cụm lại tiếp tục thực hiện cho đến khi các cụm được hình thành có số lượng node < 20.

Do quá trình tham gia và rời mạng của các node nên có thể số lượng node ở các cụm bị giảm đi. Theo định kỳ cập nhật thông tin, node đầu của cụm có thông tin hai cụm 1.1 và 1.2 kiểm tra thấy tổng số node hai cụm này <=10 *(giả sử cụm 1.1 có 4 node, cụm1. 2 có 5 node*), node đầu cụm quyết định nhập hai cụm này thành một cụm, bằng cách bỏ thông tin chỉ số cuối mỗi cụm *(cụm 1.1 bỏ chỉ số 1, cụm 1.2 bỏ chỉ số 2)* để trở thành một cụm là cụm 1, khi đó không gian ID của cụm 1 là tổng không gian ID của cụm 1.1 và 1.2 và có không gian ID trong khoảng từ 0 đến 127.

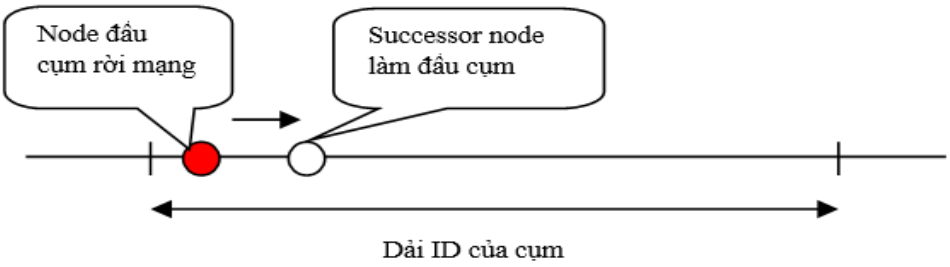
Các trường hợp xử lý của node đầu cụm

Trong một cụm, node đầu cụm có chức năng xử lý nhiều thông tin hơn các node khác, bao gồm tính toán tổng số node trong cụm, lưu thông tin của hai cụm liền kề. Vì vậy khi một node tham gia hoặc rời mạng thì phải xem xét các trường hợp để xác định lại node đầu cụm:

- Trường hợp khi một node tham gia vào cụm nhưng lại đứng trước node đầu cụm, thì node đầu cụm sẽ chuyển thông tin node mới tham gia vào làm nhiệm vụ đầu cụm (hình 3-5).
- Trường hợp một node rời mạng là node đầu cụm, node đầu cụm chuyển chức năng xử lý cho node successor của nó để đóng vai trò là node đầu cụm (hình 3-6).



Hình 3-5: Quá trình chuyển node đầu cụm cho node mới tham gia nhưng ở trước node đầu cụm



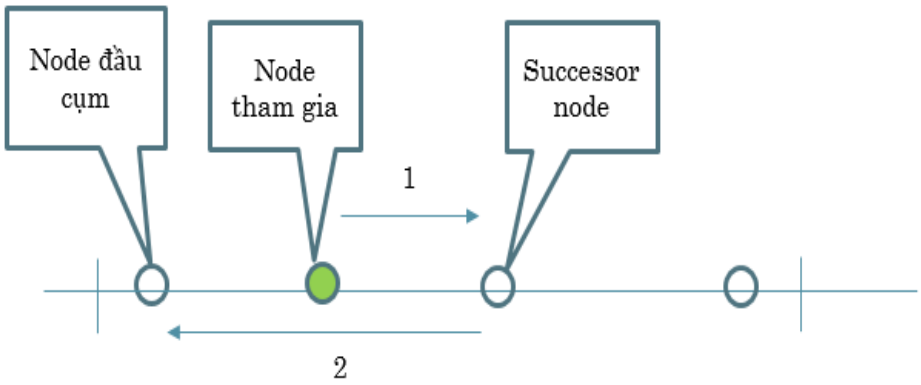
Hình 3-6 Quá trình chuyển thông tin do node đầu cụm rời mạng

Trong quá trình cập nhật thông tin về số lượng node trong một cụm, từ đó xem xét trường hợp tách cụm hoặc nhập cụm. Node đầu cụm là node xem xét đưa ra quyết định chia tách cụm hoặc sát nhập cụm, sau đó cập nhật thông tin, backup dữ liệu định kỳ trong cụm.

3.2 Phương pháp tách nhập cụm

Khi một node tham gia vào hệ thống, nó thực hiện cập nhật thông tin qua các bước sau:

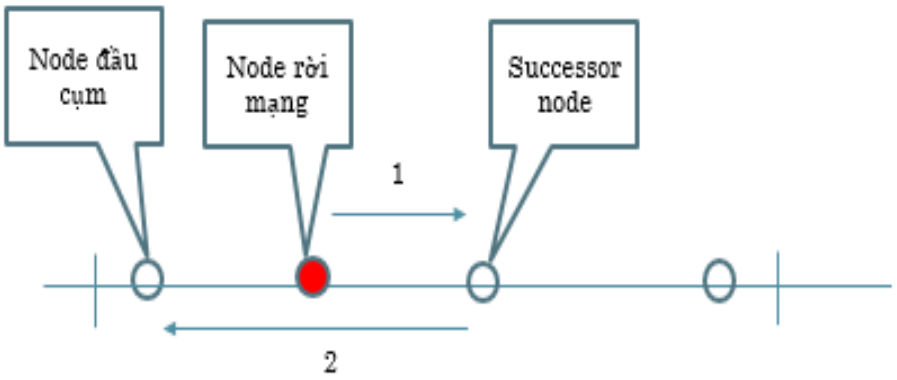
1. Thông báo cho successor node tình trạng tham gia vào hệ thống.
2. Successor node thông báo cho node đầu cụm về node mới tham gia vào hệ thống.
3. Node đầu cụm sẽ tăng tổng số node trong cụm lên 1 đơn vị (hình 3-7).



Hình 3-7 Mô tả việc tham gia một node vào hệ thống

Khi một node tham gia vào hệ thống, nó thực hiện cập nhật thông tin qua các bước sau:

1. Thông báo cho successor node tình trạng rời hệ thống.
2. Successor node thông báo cho node đầu cụm về node rời hệ thống.
3. Node đầu cụm sẽ giảm tổng số node trong cụm lên 1 đơn vị (hình 3-8).



Hình 3-8 Mô tả một node rời hệ thống

Mục đích của việc tính tăng, giảm ở node đầu cụm nhằm xem xét tổng số lượng node trong cụm để thực hiện tách cụm hay nhập cụm

Định kỳ các cụm cập nhật lại thông tin trong cụm. Node đầu cụm sau khi cập nhật số lượng các node trong cụm để xem xét các bước sau:

- Có tách tách/nhập cụm không?

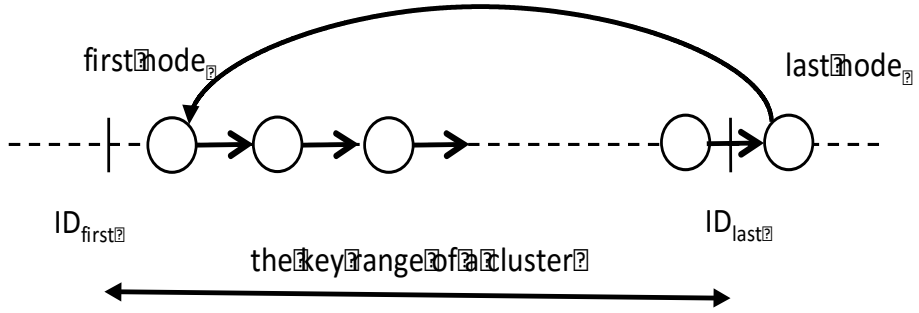
  Nếu tổng số node trong cụm, lớn hơn số node tối đa trong một cụm thì node đầu cụm thực hiện việc tách cụm.

  Nếu tổng số node của 2 cụm liền kề nhau, nhỏ hơn số node tối thiểu của hai cụm, thì thực hiện việc nhập cụm.

  Việc xác định số node tối đa trong một cụm và số node tối thiểu của hai cụm, tùy thuộc vào việc xác định độ lớn ban đầu của hệ thống, để từ đó thiết lập thông số cho phù hợp, đảm bảo tối ưu được khả năng backup.
- Sau khi tách/nhập cụm, node đầu cụm thực hiện cập nhật tới successor node, cứ như vậy đến node cuối cụm. Sau đó node cuối cụm cập nhật lại cho node đầu cụm.

Thông tin node đầu cụm gửi các node trong cụm để cập nhật bao gồm:

- Tổng số các node trong cụm
- Thông tin ID đầu cụm, cuối cụm
- Danh sách các node có dung lượng cao phục vụ cho việc lưu trữ
- Danh sách các nút rời mạng trong một khoảng thời gian định kỳ.



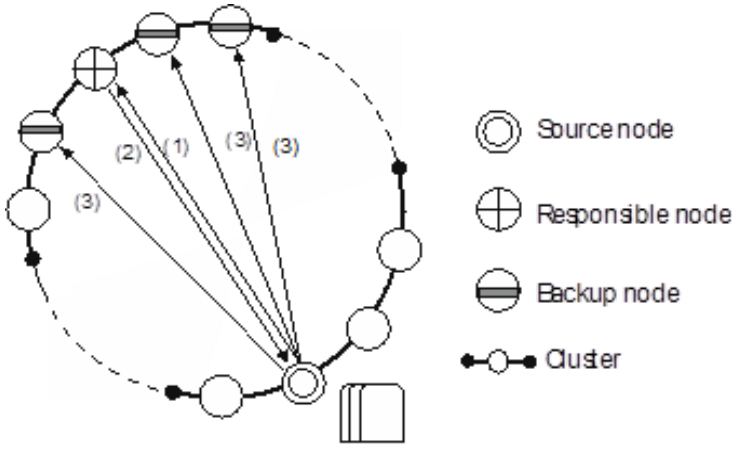
Hình 3-9 Quá trình cập nhật dữ liệu trong một cụm

3.3 Phân mảnh khi đưa một file mới vào mạng

Mặc định mỗi file khi được đưa vào mạng sẽ được chia thành *n* mảnh để phục vụ cho việc backup file.

Đối với 1 file mới đưa vào mạng Chord, dựa theo thuật toán DHT băm nội dung của file thành key, key được sinh ra sẽ được successor node quản lý, từ đó biết được cụm nào và danh sách các node tốt nhất, tiếp theo successor node sẽ thông báo cho node lưu trữ file gốc thực hiện quá trình backup các mảnh vào các node tốt nhất.

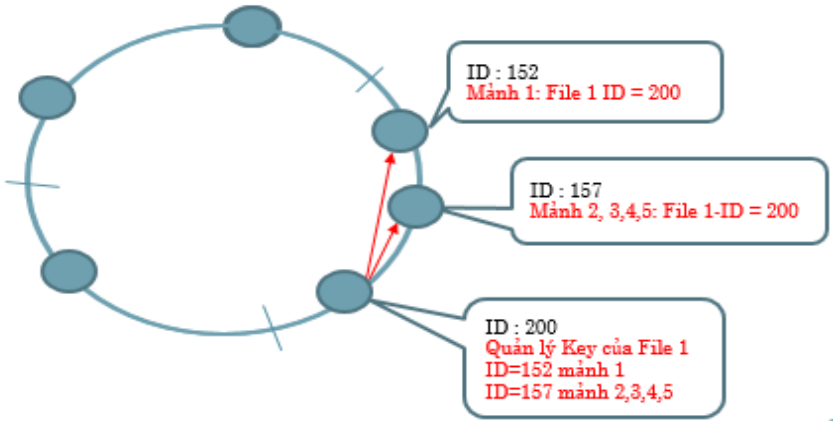
Như vậy key của một file vừa có thông tin của file gốc vừa có thông tin các mảnh. Việc tìm kiếm dữ liệu thông qua key, từ key truy vấn tới các mảnh và trả lại thông tin tìm kiếm. File gốc được sử dụng trong trường hợp các mảnh còn lại không đủ số lượng để phục hồi lại file gốc, khi đó sử dụng file gốc ban đầu để tạo thêm các mảnh mới.



Hình 3-10 Quá trình backup và phân mảnh một file mới đưa vào mạng

Node quản lý key file sẽ lưu thông tin về các node lưu trữ các mảnh dữ liệu.

Node lưu trữ các mảnh sẽ lưu thông tin về node key file.



Hình 3-11 Mô tả cách quản lý giữa key của file và các mảnh

Hình 3-11 là một ví dụ mô tả một node với ID là 200 quản ký key của file 1, file này được phân làm 6 mảnh. Trong đó, node có ID =152 chứa mảnh 1, có thông tin liên kết với node có ID = 200. Node có ID = 157 chứa 5 mảnh của file một, có thông tin liên kết với node có ID = 200. Như vậy một node có thể vừa quản lý key của một file, vừa lưu các mảnh của một file khác và có sự liên kết giữa node quản lý key của file và node lưu trữ các mảnh của file đó.

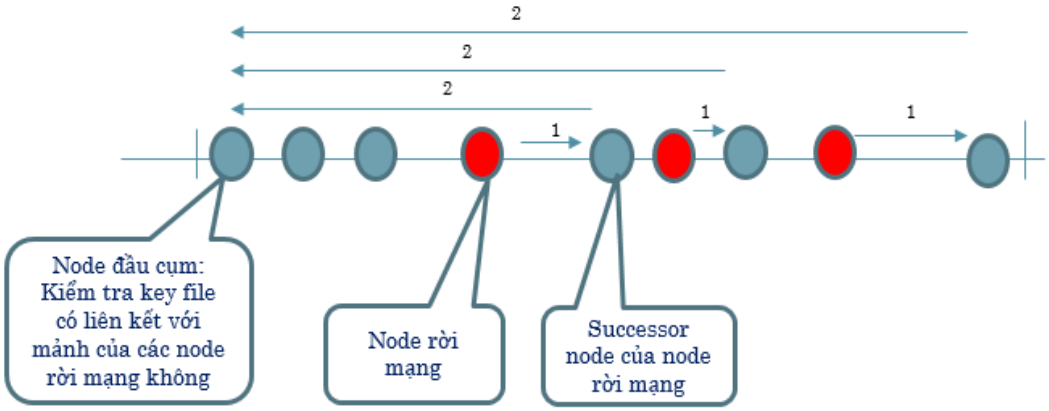
3.4 Backup khi các node rời mạng

3.4.1 Backup khi các mảnh dữ liệu nằm trong cụm

Khi 1 node rời khỏi mạng Chord, nó sẽ thực hiện các bước sau:

1. Thông báo cho successor node tình trạng rời mạng.
2. Successor node thông báo cho node đầu cụm thông tin ID node rời mạng.

Định kỳ, node đầu cụm tập hợp danh sách các node rời mạng và thông báo cho các node trong hệ thống các node trong cụm đã rời mạng.



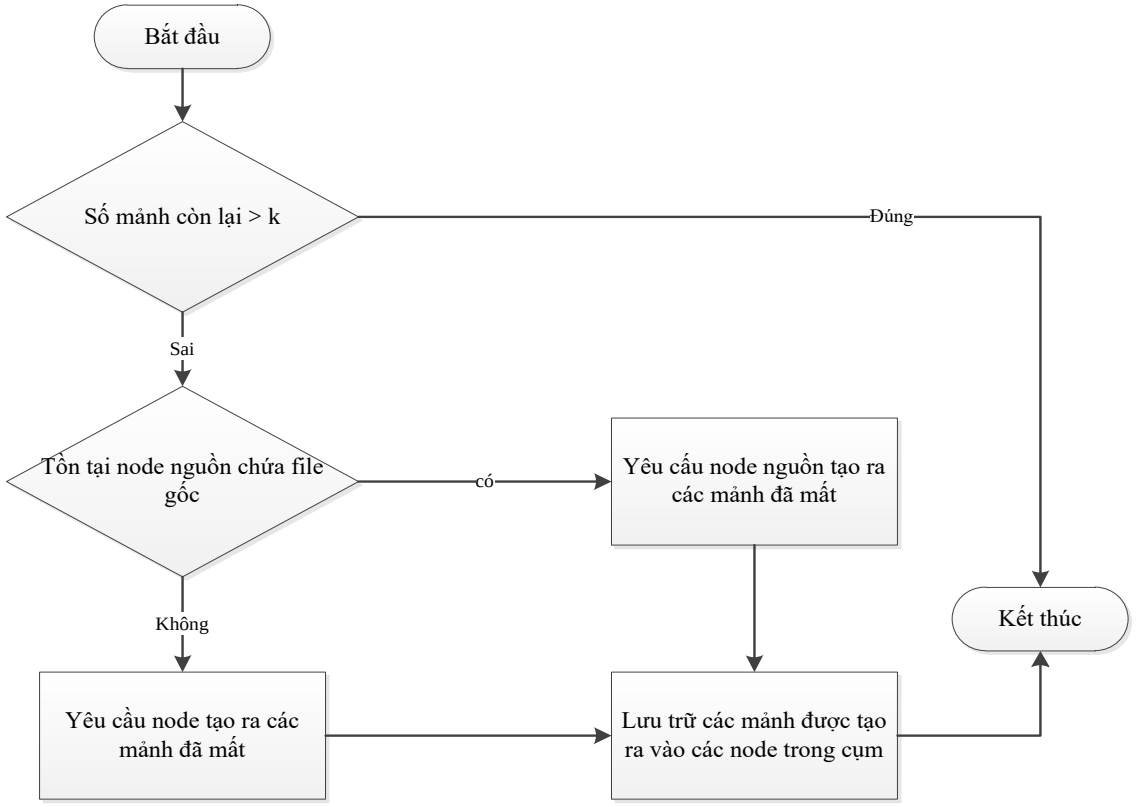
Hình 3-12 Quá trình các node rời mạng và cập nhật thông tin

Mỗi khi một node nhận được thông tin về danh sách các node rời mạng trong cụm, nó thực hiện kiểm tra lần lượt các key của file mà nó quản lý để kiểm tra các lại các mảnh dữ liệu mà key quản lý.

Trường hợp các key của file kiểm tra thấy số lượng các mảnh còn lại nhỏ hơn giá trị ngưỡng các mảnh *(k mảnh)* có thể phục hồi lại file, node chịu trách nhiệm quản lý key sẽ thực hiện backup lại các mảnh đã mất.

Trường hợp các key của file kiểm tra thấy tổng số các mảnh còn lại không có khả năng phục hồi lại file gốc, node chịu trách nhiệm quản lý key sẽ tìm lại node chứa file gốc để backup lại các mảnh. Nếu node chứa file gốc bị bị rời mạng thì không backup được các mảnh, đồng thời nó sẽ thông báo các node lưu trữ các mảnh xóa các mảnh đó trong hệ thống.

Việc khôi phục các mảnh dữ liệu đã mất theo cơ chế sau: Đối với mỗi file đưa vào mặc định được chia làm *n* mảnh, nếu hệ thống còn *k mảnh (k<n)* thì có thể phục hồi lại được file ban đầu.



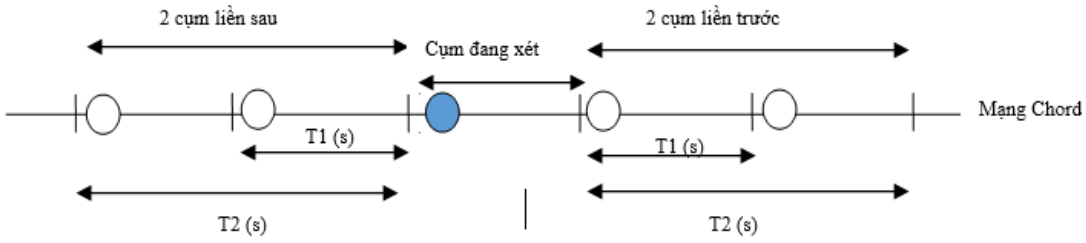
Hình 3-13 Lưu đồ kiểm tra và backup các mảnh bị mất

3.4.2 Backup khi các mảnh dữ liệu nằm ngoài cụm

Do quá trình tách cụm nhiều lần, mảnh dữ liệu có thể nằm ở giữa 2 hoặc nhiều cụm ở xa. Do đó cần có cơ chế backup lại các mảnh này nhằm duy trì các

mảnh trong một cụm, giảm việc di duyển các mảnh ở xa và cập nhật thông tin trong một cụm được nhanh nhất. Phương pháp đưa ra như sau:

Node đầu cụm sẽ có thông tin 4 node đầu cụm liền kề (2 node đầu cụm liền kề trước và 2 node đầu cụm liền kề sau).



Hình 3-14 Mô tả một node định kỳ kiểm tra backup hai cụm đứng trước và sau.

- Định kỳ T1 giây 2 node đầu cụm gần nhất sẽ gửi thông tin cho node đầu cụm xử lý và định kỳ T2 giây (T2>T1) 2 node đầu cụm xa hơn gửi thông tin cho node đầu cụm xử lý.
- Node đầu cụm sau khi có thông tin về các node rời mạng cũng tiến hành kiểm tra để backup dữ liệu tương tự như backup khi các mảnh dữ liệu nằm trong cụm.

Do việc định kỳ trao đổi thông tin về các node rời mạng cho nhau nên 5 cụm liền kề để kiểm tra các mảnh dữ liệu mà key của file liên kết đến, vì vậy nó có thể backup được các mảnh dữ liệu khi các mảnh nằm trong 5 cụm liền kề.

Trường hợp các mảnh dữ liệu nằm ngoài 5 cụm liền kề, khi các node kiểm tra các mảnh để backup dữ liệu biết được các mảnh ở xa sẽ backup lại các mảnh vào trong cụm.

CHƯƠNG 4: ĐÁNH GIÁ HIỆU QUẢ PHƯƠNG PHÁP TÁCH NHẬP CỤM SỬ DỤNG CƠ CHẾ PHÂN CỤM ĐỘNG

4.1 Chương trình mô phỏng

Chương trình mô phỏng phân cụm động được mở rộng từ chương trình mô phỏng của Jonathan Ledlie [14] và được xây dựng trên ngôn ngữ Microsoft Visual studio C++;

Các cụm được xây dựng để mô tả hoạt động mô phỏng thông qua các hàm sau:

- *long clusterInfo::clusterSegMantaince():* Duy trì các mảnh dữ liệu trong cụm, theo định kỳ cập nhật thông tin cụm, mỗi khi kiểm tra backup của một node trong cụm. Hàm sẽ kiểm tra các key của file xem trong mỗi key đó, các mảnh mà key quản lý còn lại bao nhiêu, rồi xem xét việc có phải backup lại các mảnh đã bị mất hay không.
- *void clusterInfo::updateTheBestServer():* Xử lý việc cập nhật các server tốt nhất trong cụm, trong đó server tốt nhất là các server có dung lượng lưu trữ các mảnh lớn nhất. Định kỳ cập nhật thông tin cụm, hàm sẽ kiểm tra lại các server có capacity lớn nhất cụm và đưa vào danh sách các server tốt nhất. Khi một file mới đưa vào mạng hoặc định kỳ backup lại các mảnh, dựa trên các server tốt nhất về capacity để gán các mảnh vào các server này.
- *map<double,PhysicalServer*> theBestPhysicalSever:* Hàm chứa danh sách các server tốt nhất, sắp xếp các server có dung lượng lớn nhất trong cụm nhằm mục đích phân tải các mảnh dữ liệu vào các node tốt nhất, để làm giảm việc quá tải của một node.
- *map<double,PhysicalServer*> lstServer:* Hàm quản lý danh sách tất cả các server trong một cụm, ánh xạ giữa ID của server với các server.
- *bool addPs(double psKey, PhysicalServer *ps):* Hàm thêm các server vào cụm

Các Server (node) được xây dựng để mô tả hoạt động mô phỏng thông qua các hàm sau:

- *bool PhysicalServer::checkRecoverFile(double keyF):* Hàm kiểm tra có phục hồi file dữ liệu gốc với khóa *k* truyền vào hay không. Hàm trả lại "true" nếu số mảnh còn lại đủ để backup được file gốc hoặc tìm thầy file gốc. Hàm trả lại false nếu ngược lại.
- *void PhysicalServer::keyDistributed(double keyF):* Hàm phân bổ các mảnh vào trong cụm, tính chi phí khi phải backup lại các mảnh.
- *long PhysicalServer::keyMaintaince(double keyF):* Duy trì các mảnh dữ liệu của một khóa
- *map<double,vector<double> > linkBackupPs:* Hàm chứa liên kết giữa key với các mảnh chứa trong các node
- *int birth ():* Xử lý quá trình tham gia của một node
- *int death ():* Xử lý quá trình rời mạng của một node.

Một số hàm khác:

- *void initClusters():* Khởi tạo cụm ban đầu, gán giá trị cho các thuộc tính trong cụm ban đầu.
- *void joinCluster():* Hàm xử lý việc nhập cụm
- *void splitCluster():* Hàm xử lý việc tách cụm
- *void scanClusterInfo():* Hàm quét các trường hợp tách, nhập cụm
- *char nextEvent (int &nodeid):* Hàm xử lý các sự kiện một node tham gia hoặc rời mạng
- *int initNodes (string filename, string fileDistribution, PhysicalServer *&ps):* Hàm khởi tạo một node và node đó được gán capacity và key file khi khởi tạo xong.

Chương trình mô phỏng phân cụm động

Chương trình mô phỏng xuất phát từ vòng tròn Chord chưa có node nào tham gia vào mạng và số cụm ban đầu là 1. Chương trình khởi tạo với các server được gán với các thông số:

- Khả năng về dung lượng lưu trữ các mảnh (capacity)
- Số file được phân bố cho mỗi server (file distribution)

Chương trình bắt đầu chạy theo từng vòng, ban đầu vòng 1 tạo ra các node tham gia vào mạng (node birth), từ các vòng sau sẽ có các node tham gia vào mạng (node birth) và các node rời mạng (node death).

Mỗi khi có một node tham gia hoặc rời mạng chương trình sẽ tính lại số node để xem xét trường hợp tách hoặc nhập cụm.

Định kỳ cập nhật các node tốt nhất trong mạng và backup các mảnh dữ liệu.

Đưa ra các thông số đo đếm thời gian chạy, tổng số truy vấn, số các truy vấn thành công và chi phí duy trì các mảnh dữ liệu, trong đó:

- Tổng số truy vấn: là tổng số truy vấn tìm key của file trong hệ thống để xác định việc tìm thấy hay không thấy key.
- Số truy vấn thành công: là tổng số truy vấn tìm thấy file gốc hoặc số mảnh còn lại vẫn duy trì được file gốc mà không cần phải backup.
- Phí duy trì: là phí để phục hồi các mảnh đã mất để tạo ra đủ các mảnh của một file ban đầu.

Tạo ra các file đầu vào:

- Tạo ra các file chứa các node tham gia (birth) và rời mạng (death) với tổng số 4096 node *(churnfile),* thời gian sống trung bình mỗi node là 15 phút, 30 phút, 1 giờ, 2 giờ.
- Tạo ra file dung lượng được gán cho một node *(capacity).* File này được tạo ngẫu nhiên với dung lượng trong dải từ 5 đến 235 đơn vị. Giá trị capacity trung bình của một node là 120 đơn vị.
- Tạo ra các file chứa các key file *(keyfile)* với dung lượng key file khác nhau 5%, 10%, 15%, 20%, 30%.

Phương pháp tính phân mảnh, duy trì và truy vấn

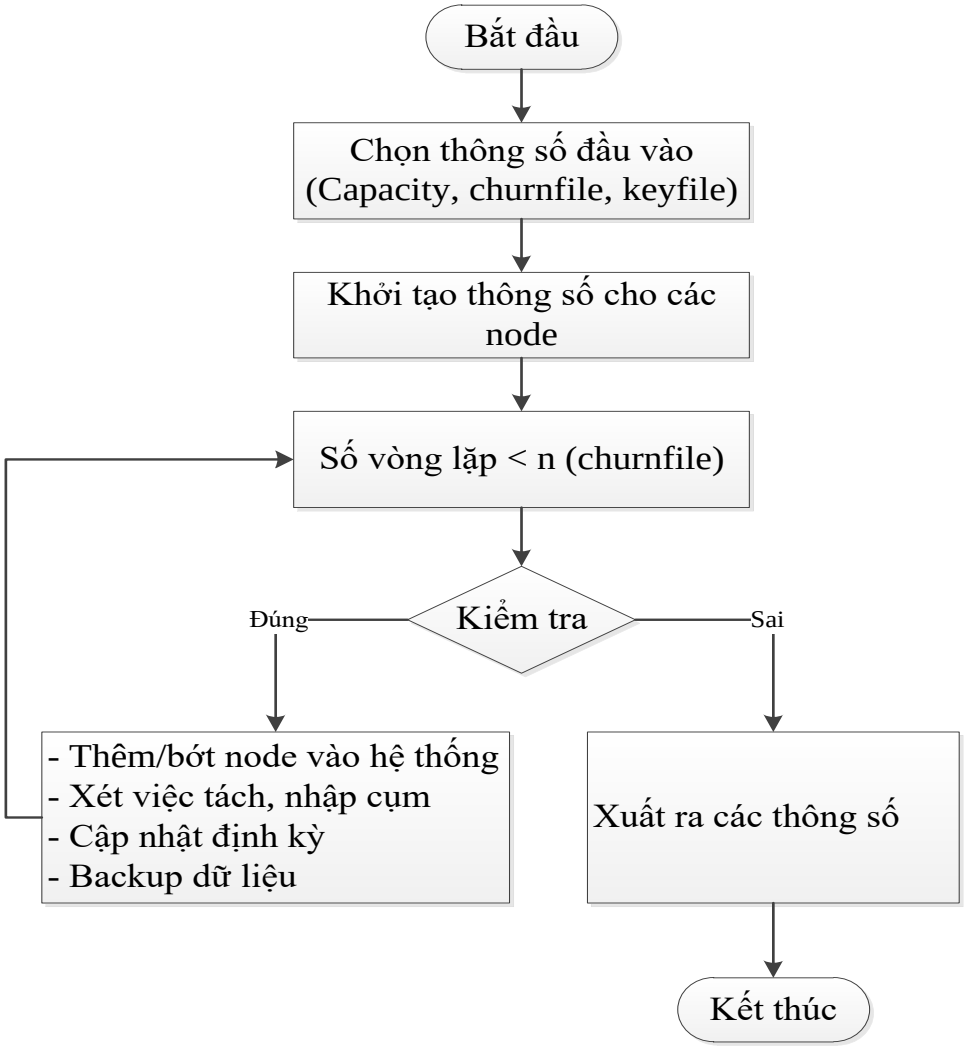
- Phương pháp phân mảnh: Mỗi file dữ liệu được chia làm 6 mảnh, định kỳ cập thông tin, nếu số mảnh <=4 sẽ thực hiện duy trì, backup lại mảnh, nếu số mảnh >=3 thì có thể phục hồi được các mảnh còn lại. Nếu số mảnh <3 thì phải tìm file gốc để backup lại các mảnh, nếu không tìm thấy file gốc thì không backup lại được file và các mảnh.
- Phương pháp tính phí duy trì: Khi đến ngưỡng backup (số mảnh <=4), sẽ phải tạo thêm các mảnh để đảm bảo việc duy trì dữ liệu. Mỗi mảnh tạo ra khi được tính vào một đơn vị.
- Phương pháp tính các truy vấn: Định kỳ các node kiểm tra lại các key của file trong tập key của hệ thống, mỗi một key được tính là *một truy*

vấn. Một truy vấn thành công nếu node nguồn chứa file gốc còn sống và có số mảnh có đủ để backup lại file.

Các thông số đo:

- Dung lượng (capacity) của một node: Trung bình một node có khả năng chứa 120 mảnh dữ liệu, mỗi file phân ra làm 6 mảnh và có một key file quản lý các mảnh. Trong chương trình mô phỏng tạo ra file đầu vào lưu trữ key của file là 5%, có ngĩa là chương trình phân bổ các key file cho cả không gian ID của mạng (4096 ID) đồng đều cho mỗi ID 5 key của file.
- Thời gian sống của một node: Được sinh ra khi tạo file chứa các node tham gia hoặc rời mạng khi tạo file.

Mô tả chương trình mô phỏng theo lưu đồ dưới đây:



Một số điểm phân biệt giữa chương trình mô phỏng phân cụm động và phân cụm tĩnh

Phân cụm tĩnh	**Phân cụm động**
Khởi tạo ban đầu với số cụm xác định	Khởi tạo ban đầu với chỉ 1 cụm
Số node trong một cụm không giới hạn	Giới hạn số node trong một cụm
Các mảnh dữ liệu luôn nằm trong một cụm	Các mảnh dữ liệu có thể nằm ở nhiều cụm khác nhau
Số lượng cụm cố định	Số lượng cụm thay đổi tùy thuộc vào số node tham gia hoặc rời mạng

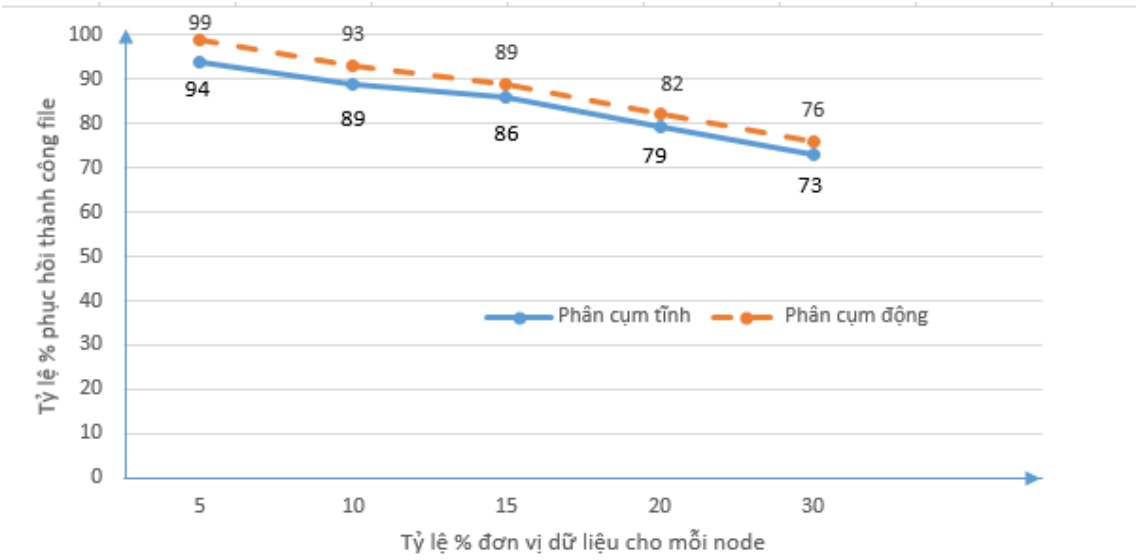
Bảng 4-1: So sánh sự khác nhau giữa phân cụm tĩnh và phân cụm động

4.2 Đánh giá và so sánh một số thông số của phương pháp tách nhập cụm theo cơ chế phân cụm động so với phân cụm tĩnh.

4.2.1 Tỷ lệ khôi phục file ban đầu thành công (khi cố định thời gian sống 1 node và tăng số file)

Các thông số đầu vào cho chương trình mô phỏng:

- Cố định thời gian sống của mỗi node: 30 phút
- Thay đổi số lượng dữ liệu được lưu trữ trong một node lần lượt là 5%, 10%, 15%, 20%, 30%
- Capacity: 120k
- Phân cụm động: Tách cụm khi >= 30 node, nhập cụm khi tổng 2 cụm <= 20 node
- Phân cụm tĩnh: Cố định 20 cụm



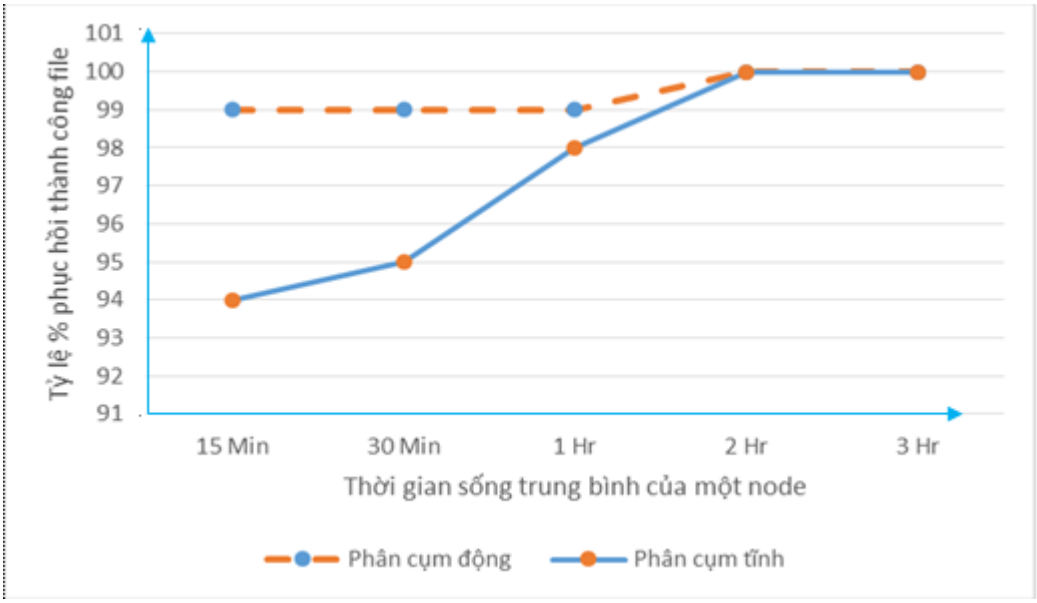
Biểu đồ 4-1 So sánh tỷ lệ khôi phục file ban đầu thành công giữa phân cụm tĩnh và phân cụm động

Từ kết quả của chương trình cho thấy khi dữ liệu đưa vào node tăng lên thì tỷ lệ truy vấn thành công giảm theo, với 5 đơn vị dữ liệu đưa vào một node tỷ lệ thành công là 99% và thấp dần xuống 76% khi dữ liệu đưa vào là 30 đơn vị. Kết quả này cũng cho thấy tỷ lệ thành công của phương pháp phân cụm động cao hơn so với phân cụm tĩnh, trung bình khoảng 3% do thời gian cập nhật thông tin trong một cụm nhanh hơn nên quá trình backup tốt hơn và tỷ lệ truy vấn thành công cao hơn.

4.2.2 Tỷ lệ khôi phục file ban đầu thành công (cố định số lượng file và thay đổi thời gian sống)

Các thông số đầu vào cho chương trình mô phỏng:

- Thay đổi thời gian sống của một node: 15 phút, 30 phút, 1 giờ và 2 giờ, 3 giờ.
- Cố định số lượng dữ liệu lưu trữ trong một node là 20%.
- Capacity: 120k
- Phân cụm động: Tách cụm khi >= 30 node, nhập cụm khi tổng 2 cụm <= 20 node
- Phân cụm tĩnh: Cố định 20 cụm



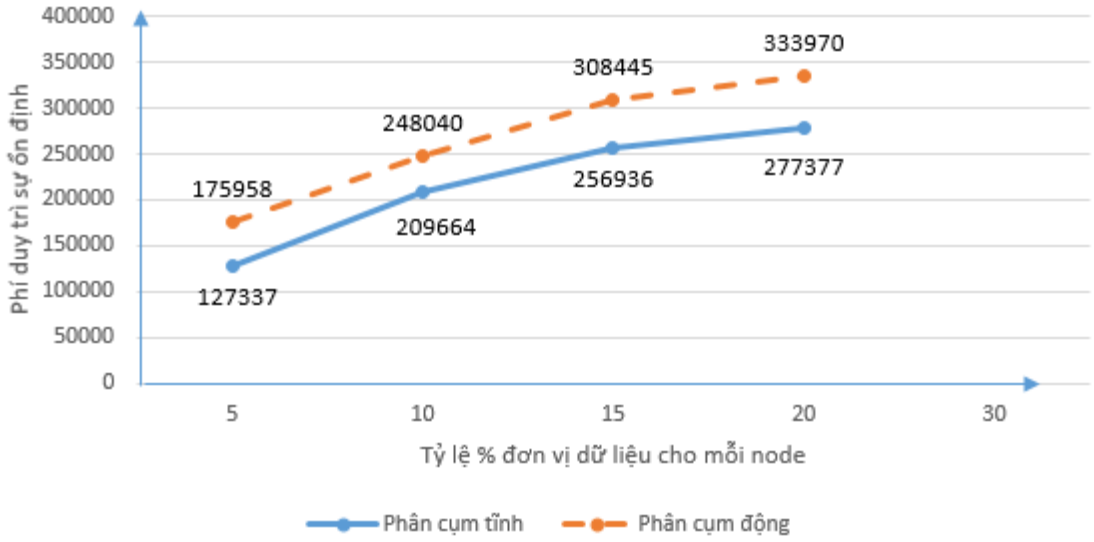
Biểu đồ 4-2 So tỷ lệ file ban đầu thành công giữa phân cụm tĩnh và phân cụm động khi thay đổi thời gian sống của một node.

Kết quả trên biểu đồ 4-2 cho thấy, thời gian sống của một node càng lâu, tỉ lệ rời mạng của các node trong cụm thấp hơn, các mảnh dữ liệu bị phân tán ra các cụm ít hơn dẫn tới tỷ lệ tìm thấy các mảnh trong cụm cao hơn và tỷ lệ phục hồi thành công file cao. Tỷ lệ phục hồi thành công file của phương pháp phân cụm tĩnh thấp hơn so với phân cụm động do quá trình cập nhật các node tốt nhất trong cụm của phân cụm động nhanh hơn nên backup được nhiều mảnh đã mất cho các node tốt nhất, dẫn tới tỷ lệ truy vấn thành công cao hơn, tỷ lệ phục hồi thành công file cao hơn. Trường hợp thời gian sống của một node từ 2 giờ trở lên, tỷ lệ phục hồi thành công file của cả phân cụm tĩnh và phân cụm động là 100%.

4.2.3 Chi phí cho việc duy trì các mảnh là bao nhiêu.

Các thông số đầu vào cho chương trình mô phỏng:

- Cố định thời gian sống của mỗi node: 30 phút
- Thay đổi số lượng dữ liệu được lưu trữ trong một node lần lượt là 5%, 10%, 15%, 20%
- Capacity: 120k
- Phân cụm động: Tách cụm khi >= 30 node, nhập cụm khi tổng 2 cụm <= 20 node
- Phân cụm tĩnh: Cố định 20 cụm



Biểu đồ 4-3 So sánh chi phí duy trì các mảnh giữa phân cụm tĩnh và phân cụm động.

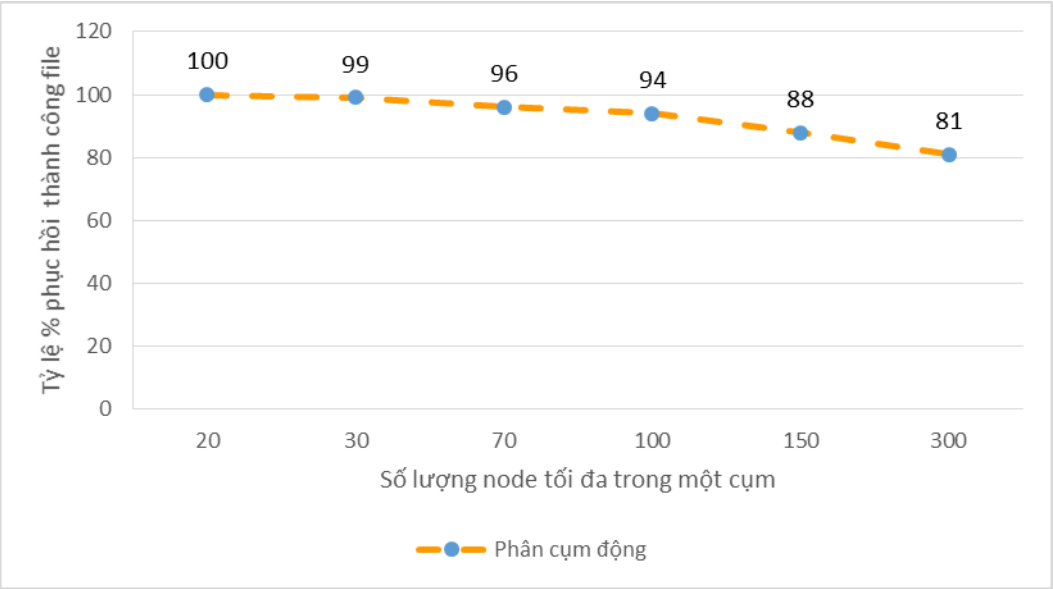
Kết quả trên biểu đồ chỉ ra rằng chi phí để backup dữ liệu cho các mảnh của phương pháp phân cụm động cao hơn phân cụm tĩnh, điều này được giải thích như sau:

- Phương pháp phân cụm động được phân chia thành nhiều cụm trong quá trình chạy, đo đó các mảnh dữ liệu có thể nằm ở nhiều cụm khác nhau. Việc định kỳ backup dữ liệu giữa các cụm với nhau mất nhiều thời gian, do các mảnh ngoài cụm cập nhập thông tin chậm hơn trong một cụm, dẫn tới một số mảnh dữ liệu không có thông tin. Khi khôi phục lại file ban đầu phải mất chi phí để phục hồi các mảnh này nên tốn chi phí hơn
- Theo phương pháp phân cụm tĩnh dữ liệu luôn nằm ở các node trong cụm do đó chi phí để tìm thấy và phục hồi các dữ liệu thấp hơn so với phân cụm động.

4.2.4 So sánh file ban đầu thành công khi thay đổi số lượng node trong cụm

Các thông số đầu vào cho chương trình mô phỏng:

- Cố định thời gian sống của mỗi node: 1 giờ
- Số lượng dữ liệu được lưu trữ trung bình trong một node là 5%
- Capacity: 120k
- Số node tách, nhập cụm chạy mô phỏng lần lượt là (30-20, 70-50, 100-70, 150-100, 300-100).



Biểu đồ 4-4 Tỷ lệ phục hồi công file khi thay đổi số lượng node tách, nhập trong một cụm

Thí nghiệm mô phỏng trong trường hợp thay đổi số lượng node tách, nhập cụm cho thấy khi số node tách, nhập cụm thấp tỷ lệ phục hồi thành công file cao hơn so với số node tách, nhập cụm lớn. Điều này chứng tỏ với cụm có số lượng node nhỏ việc cập nhật định kỳ nhanh hơn so với cụm có số lượng node lớn, từ đó việc phục hồi file và các mảnh dữ liệu nhanh hơn dẫn tới tỷ lệ phục hồi thành công file cao hơn.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Sự phát triển rộng rãi của các máy tính trên mạng, bên cạnh những ưu điểm của hệ thống tập trung (client-server), hệ thống phân tán cũng có những ưu điểm riêng biệt của nó. Với việc nghiên cứu và khắc phục những điểm yếu trong hệ thống phân tán như khả năng mở rộng, tính bảo mật, tính duy trì dữ liệu cũng như cân bằng tải, trong tương lai việc ứng dụng hệ thống mạng ngang hàng sẽ trở nên ngày càng phổ biến.

Thông qua việc nghiên cứu về backup dữ liệu theo cơ chế phân cụm động, phần nào cũng cho thấy những ưu điểm và linh hoạt trong mạng ngang hàng có cấu trúc sử dụng thông qua giao thức Chord. Kết quả đánh giá phương pháp phân cụm động và phân cụm tĩnh cho thấy, mỗi phương pháp có ưu nhược điểm khác nhau, tùy từng trường hợp có thể áp dụng theo cơ chế phân cụm động hay phân cụm tĩnh.

Nhìn chung cơ chế phân cụm tĩnh phù hợp với những mạng được ước lượng trước số node tham gia hoặc rời mạng trong hệ thống, qua đó việc chia cụm cố định sẽ phù hợp để đảm bảo cả việc backup, thời gian backup cũng như duy trì các mảnh dữ liệu.

Với cơ chế phân cụm động, không phụ thuộc vào số lượng các node tham gia hoặc rời mạng, thời gian backup được ổn định, tỷ lệ khôi phục thành công file dữ liệu cao hơn nhưng chi phí duy trì thì tốn hơn, đòi hỏi những node tham gia gia mạng với cấu hình cao hơn để tăng thời gian xử lý backup.

Mặc dù đã đạt được một số kết quả cho thấy ở trên, tuy nhiên việc mô phỏng này vẫn còn một số hạn chế cần được bổ sung, nghiên cứu thêm để phù hợp với thực tế như: tính khoảng cách của các node khi tham gia vào hệ thống, từ việc xác định được khoảng cách các node tham gia vào hệ thống sẽ phân bổ vào các cụm hợp lý hơn nhằm giảm tải cho việc duy trì dữ liệu.

Trong tương lai có thể mở rộng nội dung của luận văn thông qua việc tính khoảng cách các node khi tham gia vào cụm.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Nguyễn Hoài Sơn, Hồ Sĩ Đàm (2008), "*Tìm kiếm thông tin theo các giá trị thuộc tính trên mạng ngang hàng có cấu trúc*", Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội.

[2] Nguyễn Đại Thọ (2007), "*Công nghệ mạng ngang hàng*", Bộ môn Mạng & Truyền thông Máy tính Khoa Công nghệ Thông tin, trường Đại học Công nghệ - Đại học Quốc gia Hà Nội.

[3] Ngô Hoàng Giang (2008), *"Đánh giá hiệu năng của một số thuật toán bảng băm phân tán DHT và đưa ra giải pháp cải tiến hiệu năng của thuật toán Chord",* luận văn thạc sỹ Công nghệ thông tin trường đại học Bách khoa Hà Nội.

Tiếng Anh

[4] Nguyen Dinh Nghia, Nguyen Hoai Son (2016), "A Cluster-based File Replication Scheme for DHT-based File Backup Systems", "*Advanced Technologies for Communications (ATC), 2016 International Conference on",* ISSN: 2162-1039, No 16520217.

[5] Kale A.R and SHIRBHATE D.D (Mar 2012), "An advanced hybrid peer to peer botnet", "*International Journal of wireless Comunication*", ISSN: 2231-3559, Vol.2.

[6] John cates (2003), *"Robust and Efficient Data Management for Distributed Hash table",* Submitted to Departmant of Electrical and computer science - Massachusetts institute of technology, USA.

[7] IonStoca RobMorris, David Karger, M.Frans Kaashoek, Hari Balakrishnan (2001), "Chord: A Ccalable peer-to-peer lookup service for internet Applications", *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*", ISBN:1-58113-411-8, Vol.31.

[8] L. Garc´es-Erice, P.A. Felber, E.W. Biersack, G. Urvoy-Keller K.W. Ross (March 2004), "Data Indexing in Peer-to-Peer DHT Networks", *"Proceedings of the 24th International Conference on Distributed Computing Systems",* ISBN: 0-7695-2086-3.

[9] Micheael Rabin (April 1989), "Efficient dispersal of information for security, load balancing, and fault tolerance", "*Journal of the Association for Computing Machinery*", Vol. 36, No.2

[10] Sameh El-Alsary and Seif Haridi (July 2004), "*An overview of structured P2P overlay network*", Swedish Institute of Computer Science, Swedish.

[11] S. Legtchanko, P. Sen, Cilles Muller (April 2009), *"*Churn-resilent replecation strategе for peer to peer distributed hash-tables*"*, "*Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems*", ISBN: 978-3-642-05117-3, No 6897.

[12] S. Ratnasamy, P. Francis, M. Handley and R. Karp, (Aug. 2001), "A Scalable Content-Addressable Network", *"Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications",* ISBN:1-58113-411-8, Vol.31.

[13] Alberto Montresor (Arp 2016), "*Distributed algorithms peer to peer system*", University of trento, Italy.

[14] J.Ledlie, M.Seltzer. (Mar 2005), "Distributed, Secure Load Balancing with Snew, Heterogeneity and Churrn", *"In Proceedings of 24th Annual Joint Conference of the IEEE Computer and Comunications Societies",* ISSN: 0743-166X, Vol.2.

[15] Christos Gkantsidis, Milena Mihail (Mar 2004),*"Random walks in peer to peer networks", "Performance Evaluation - P2P computing systems"* ISSN: 0743-166X, No 8410756.

[16] Vu Q.H, LuLu M, Ooi P.C (2010), *"Peer to peer computing principles and applications",* Spinger 2010, XVI, 317 p. ISPN 978-3-642-03513-5.