TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÀI TẬP LỚN MÔN QUẢN TRỊ HỆ THỐNG THÔNG TIN**

**QUẢN LÝ NỘI DUNG DOANH NGHIỆP**

*Người hướng dẫn*: **Hồ Thị Linh**

*Người thực hiện*: **Nguyễn Minh Hải – 51703074**

**Vũ Đình Đạt – 51703057**

Lớp : **17050303**

Khoá :  **21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÀI TẬP LỚN MÔN QUẢN TRỊ HỆ THỐNG THÔNG TIN**

**QUẢN LÝ NỘI DUNG DOANH NGHIỆP**

*Người hướng dẫn*: **Hồ Thị Linh**

*Người thực hiện*: **Nguyễn Minh Hải – 51703074**

**Vũ Đình Đạt – 51703057**

Lớp : **17050303**

Khoá :  **21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

LỜI CÁM ƠN

Trong suốt quá trình làm bài tiểu luận, nhóm của em đã được sự giúp đỡ của giảng viên và bạn bè xung quanh. Với lòng biết ơn sâu sắc, tòan thể nhóm em xin gửi lời cám ơn chân thành đến cô Hồ Thị Linh đã tận tâm hướng dẫn chúng em qua từng buổi học trên lớp cũng như những buổi nói chuyện, thảo luận về những thắc mắc. Nhờ có những lời hướng dẫn, dạy bảo đó, bài luận văn này của nhóm đã hoàn thành một cách xuất sắc nhất. Một lần nữa, em xin gửi lời cảm ơn chân thành đến cô.

Bài tập lớn được thực hiện trong một thời gian rất ngắn cùng với vốn kiến thức có hạn. Do vậy, khó tránh khỏi những thiếu sót, nhóm em mong nhận được những lời góp ý của quý Thầy Cô trong khoa để có thể hoàn thiện bản thân hơn.

Nhóm em xin chân thành cám ơn!

TIỂU LUẬN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng nhóm tôi và được sự hướng dẫn khoa học của Giảng viên Hồ Thị Linh. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong luận văn còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình**. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Minh Hải*

*Vũ Đình Đạt*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TP. Hồ Chí Minh, ngày tháng năm

(ký tên và ghi rõ họ tên)

**TÓM TẮT**

Trong phần phần **1.1**, chúng ta làm rõ các mục tiêu và thảo luận về các khía cạnh chức năng và kiến trúc của các hệ thống cơ sở dữ liệu song song. Cụ thể, chúng ta thảo luận về các ưu điểm và hạn chế tương ứng của các kiến trúc hệ thống song song (shared-memory, shared-disk, shared-nothing) cùng với một số khía cạnh quan trọng bao gồm quan điểm của cả người dùng cuối cùng, quản trị viên cơ sở dữ liệu và nhà phát triển hệ thống. Sau đó, chúng tôi trình bày các kỹ thuật để đặt dữ liệu trong phần **1.2**, xử lý truy vấn trong phần **1.3** và cân bằng tải trong phần **1.4.** Trong phần **1,5**, chúng tôi trình bày việc sử dụng các kỹ thuật quản lý dữ liệu song song trong các cụm cơ sở dữ liệu, một loại hệ thống cơ sở dữ liệu song song quan trọng được triển khai trên một cụm PC.

**MỤC LỤC**

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**Parallel Database Systems**

Các ứng dụng sử dụng nhiều dữ liệu yêu cầu hỗ trợ cho các cơ sở dữ liệu rất lớn (ví dụ: hàng trăm terabyte hoặc petabyte). Ví dụ về các ứng dụng này là thương mại điện tử, lưu trữ dữ liệu và khai thác dữ liệu. Cơ sở dữ liệu lớn thường được truy cập thông qua số lượng lớn các giao dịch đồng thời (ví dụ: thực hiện các đơn đặt hàng trực tuyến trên một cửa hàng điện tử) hoặc các truy vấn phức tạp (ví dụ: các truy vấn hỗ trợ quyết định). Các loại truy cập đầu tiên là đại diện của các ứng dụng Xử lý giao dịch trực tuyến (OLTP), trong khi loại thứ hai là đại diện cho các ứng dụng Xử lý phân tích trực tuyến (OLAP). Hỗ trợ cơ sở dữ liệu rất lớn một cách hiệu quả cho OLTP hoặc OLAP có thể được giải quyết bằng cách kết hợp tính toán song song và quản lý cơ sở dữ liệu phân tán.

# Các hệ thống cơ sở dữ liệu song song có thể khai thác tính song song trong quản lý dữ liệu để cung cấp các máy chủ cơ sở dữ liệu hiệu suất cao và tính sẵn sàng cao. Vì vậy, họ có thể hỗ trợ cơ sở dữ liệu rất lớn với độ tải rất cao.

Hầu hết các nghiên cứu về các hệ thống cơ sở dữ liệu song song đã được thực hiện trong bối cảnh mô hình quan hệ, nó cung cấp một cơ sở tốt cho cơ sở dữ liệu song song. Trong chương này, chúng tôi trình bày cách tiếp cận hệ thống cơ sở dữ liệu song song như một giải pháp để quản lý dữ liệu hiệu suất cao và tính sẵn sàng cao. Chúng tôi thảo luận về những ưu điểm và bất lợi của các kiến trúc hệ thống song song khác nhau và chúng tôi trình bày các kỹ thuật thực hiện chung.

Việc thực hiện các hệ thống cơ sở dữ liệu song song tự nhiên dựa vào các kỹ thuật cơ sở dữ liệu phân tán. Tuy nhiên, các vấn đề quan trọng là vị trí dữ liệu, xử lý truy vấn song song và cân bằng sự truyền tải vì số lượng nodes có thể cao hơn nhiều so với trong một phần mềm quản lý CSDL phân tán. Hơn nữa, một máy tính song song thường cung cấp khả năng giao tiếp nhanh, đáng tin cậy có thể được khai thác để thực hiện hiệu quả việc quản lý và nhân rộng giao dịch phân tán. Do đó, mặc dù các nguyên tắc cơ bản giống như trong quản lý CSDL phân tán, các kỹ thuật cho các hệ thống cơ sở dữ liệu song song khá khác nhau.

# Kiến trúc hệ thống cơ sở dữ liệu song song:

Trong phần này, chúng tôi cho thấy giá trị của các hệ thống song song để quản lý cơ sở dữ liệu hiệu quả. Chúng tôi thúc đẩy nhu cầu cho các hệ thống cơ sở dữ liệu song song bằng cách xem xét các yêu cầu của các hệ thống thông tin rất lớn sử dụng các xu hướng công nghệ phần cứng hiện tại. Chúng tôi trình bày các chức năng và kiến trúc của các hệ thống cơ sở dữ liệu song song. Cụ thể, chúng tôi trình bày và so sánh các kiến trúc chính: shared memory, shared-disk, shared-nothing và kiến trúc hỗn hợp.

## 14.1.1 Đối tượng:

Xử lý song song khai thác các máy tính đa bộ xử lý để chạy các chương trình ứng dụng bằng cách sử dụng một số bộ xử lý kết hợp, để cải thiện hiệu suất. Ứng dụng nổi bật của nó đã có từ lâu trong điện toán khoa học bằng cách cải thiện thời gian đáp ứng của các ứng dụng số [Kowalik, 1985; Sharp, 1987]. Sự phát triển trong cả máy tính song song đa năng sử dụng bộ vi xử lý tiêu chuẩn và kỹ thuật lập trình song song [Osterhaug, 1989] đã cho phép xử lý song song để xâm nhập vào trường xử lý dữ liệu.

Hệ thống cơ sở dữ liệu song song kết hợp quản lý cơ sở dữ liệu và xử lý song song để tăng hiệu suất và tính sẵn sàng. Lưu ý rằng hiệu suất cũng là đối tượng của các máy cơ sở dữ liệu trong thập niên 70 và 80 [Hsiao, 1983]. Vấn đề mà quản lý cơ sở dữ liệu thông thường phải đối mặt từ lâu đã được gọi là “ nút cổ chai I / O ” [Boral và DeWitt, 1983], do thời gian truy cập đĩa cao liên quan đến thời gian truy cập bộ nhớ chính (thường nhanh hơn hàng trăm nghìn lần).

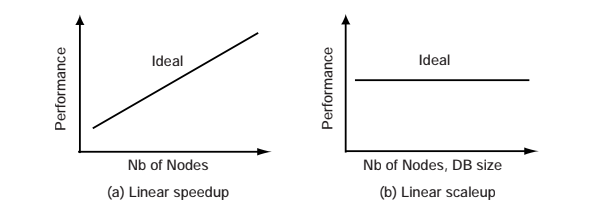
Ban đầu, các nhà thiết kế máy cơ sở dữ liệu đã giải quyết vấn đề này thông qua phần cứng chuyên dùng, ví dụ, bằng cách giới thiệu các thiết bị lọc dữ liệu trong các đầu đĩa. Tuy nhiên, cách tiếp cận này thất bại vì chi phí / hiệu suất kém so với giải pháp phần mềm, có thể dễ dàng hưởng lợi từ tiến bộ phần cứng trong công nghệ silicon. Một ngoại lệ đáng chú ý cho những thất bại này là thiết bị lọc dựa trên phần cứng CAFS-ISP [Babb, 1979] được gói trong bộ điều khiển đĩa để tìm kiếm kết hợp nhanh. Ý tưởng đẩy các chức năng cơ sở dữ liệu gần hơn với đĩa đã nhận được sự quan tâm mới với việc giới thiệu các bộ vi xử lý đa năng trong các bộ điều khiển đĩa, do đó dẫn đến các đĩa thông minh [Keeton et al., 1998]. Ví dụ: các chức năng cơ bản yêu cầu quét tuần tự tốn kém, ví dụ: chọn các thao tác trên các bảng có các biến vị ngữ mờ, có thể được thực hiện hiệu quả hơn ở cấp đĩa vì chúng tránh làm quá tải bộ nhớ DBMS với các khối đĩa không liên quan. Tuy nhiên, việc khai thác các đĩa thông minh đòi hỏi phải điều chỉnh DBMS, đặc biệt là bộ xử lý truy vấn để quyết định có sử dụng các chức năng của đĩa hay không. Vì không có công nghệ đĩa thông minh tiêu chuẩn, việc thích nghi với các công nghệ đĩa thông minh khác nhau làm tổn hại đến tính di động của DBMS.

Tuy nhiên, kết quả quan trọng trong giải pháp chung cho nút cổ chai I/O.Chúng ta có thể tóm tắt giải pháp này khi tăng băng thông I / O thông qua song song. Ví dụ, nếu chúng ta lưu trữ một cơ sở dữ liệu có kích thước D trên một đĩa có thông lượng T, thông lượng hệ thống bị giới hạn bởi T. Ngược lại, nếu chúng ta phân vùng cơ sở dữ liệu qua n đĩa, mỗi ổ có dung lượng D/n và thông lượng T’(tương đương với T), chúng ta sẽ có được thông lượng hoàn chỉnh là n \* T’ có thể được tiêu thụ tốt hơn bởi nhiều bộ xử lý (ideally n). Lưu ý rằng giải pháp hệ thống cơ sở dữ liệu bộ nhớ chính [Eich, 1989], cố gắng duy trì cơ sở dữ liệu trong bộ nhớ chính, là bổ sung chứ không phải thay thế. Đặc biệt, nút cổ chai truy cập bộ nhớ trong các hệ thống bộ nhớ chính cũng có thể được xử lý bằng cách sử dụng song song theo cách tương tự. Do đó, các nhà thiết kế hệ thống cơ sở dữ liệu song song đã cố gắng phát triển các giải pháp định hướng phần mềm để khai thác các máy tính song song.

Một hệ thống cơ sở dữ liệu song song có thể được định nghĩa đơn giản là một DBMS được triển khai trên một máy tính song , chỉ có thể yêu cầu viết lại các giao diện hệ điều hành thường dùng, đến sự kết hợp tinh vi giữa các chức năng hệ thống cơ sở dữ liệu và xử lý song song bên trong kiến ​​trúc phần cứng / phần mềm mới. Như mọi khi, chúng ta có sự đánh đổi truyền thống giữa tính di động (với một số nền tảng) và hiệu quả. Cách tiếp cận tinh vi có khả năng khai thác tốt hơn các cơ hội được cung cấp bởi một bộ xử lý đa năng với chi phí cho tính truyền đi. Thật thú vị điều này mang lại những lợi thế khác nhau cho các nhà sản xuất máy tính và nhà cung cấp phần mềm. Do đó, điều quan trọng là đặc trưng hóa các điểm chính trong không gian của kiến ​​trúc hệ thống song song thay thế. Để làm như vậy, chúng tôi sẽ thực hiện chính xác giải pháp hệ thống cơ sở dữ liệu song song và các chức năng cần thiết. Điều này sẽ hữu ích trong việc so sánh các kiến ​​trúc hệ thống cơ sở dữ liệu song song.

Các mục tiêu của các hệ thống cơ sở dữ liệu song song được bao phủ bởi các DBMS phân tán (hiệu suất, tính sẵn sàng, khả năng mở rộng). Hơn hết, một hệ thống cơ sở dữ liệu song song nên cung cấp được các lợi thế sau.

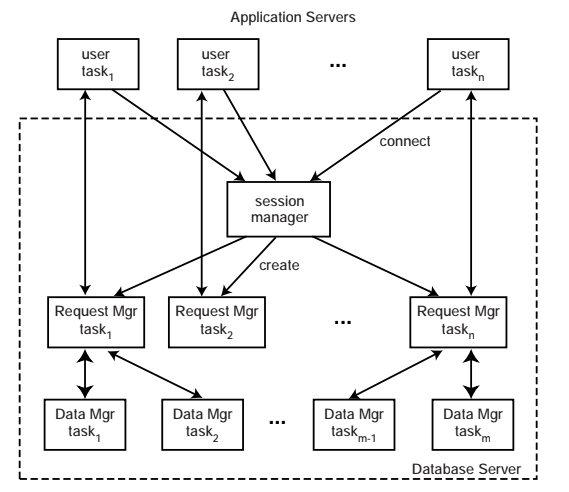
1. **Hiệu suất cao**: Điều này có thể đạt được thông qua một số giải pháp bổ sung: hỗ trợ hệ điều hành hướng cơ sở dữ liệu, quản lý dữ liệu song song, tối ưu hóa truy vấn và cân bằng truyền tải. Việc hệ điều hành bị hạn chế và “nhận thức” về các yêu cầu cơ sở dữ liệu cụ thể (ví dụ: quản lý bộ đệm) đơn giản hóa việc thực hiện các chức năng cơ sở dữ liệu cấp thấp và do đó giảm được chi phí. Chẳng hạn, chi phí của một tin nhắn có thể giảm đáng kể xuống còn vài trăm hướng dẫn bằng cách sử dụng các giao thức truyền thông. Tính song song có thể tăng thông lượng, sử dụng tính song song giữa các truy vấn và giảm thời gian phản hồi giao dịch, sử dụng tính tương đương trong truy vấn nội bộ. Tuy nhiên, việc giảm thời gian phản hồi của một truy vấn phức tạp thông qua song song quy mô lớn cũng có thể làm tăng tổng thời gian của nó (bằng cách bổ sung thêm) và làm tác động thông lượng như là một tác dụng phụ. Do đó, điều này rất quan trọng để tối ưu hóa và song song hóa các truy vấn để giảm thiểu chi phí song song, ví dụ: bằng cách hạn chế mức độ song song cho truy vấn. Cân bằng độ truyền tải là khả năng hệ thống phân chia một khối lượng công việc nhất định cho tất cả các bộ xử lý. Tùy thuộc vào kiến ​​trúc hệ thống song song, nó có thể đạt được trạng thái tĩnh bằng cách thiết kế cơ sở dữ liệu vật lý phù hợp hoặc tự động trong thời gian chạy.
2. **Tính sẵn sàng cao:** Bởi vì một hệ thống cơ sở dữ liệu song song bao gồm nhiều thành phần dự phòng, nó cũng có thể làm tăng tính khả dụng của dữ liệu và khả năng chịu lỗi. Trong một hệ thống song song cao có nhiều nodes, xác suất xảy ra lỗi node bất cứ lúc nào cũng tương đối cao. Sao chép dữ liệu tại một số nodes rất hữu ích để hỗ trợ chuyển đổi dự phòng, một kỹ thuật chống lỗi cho phép tự động chuyển hướng các giao dịch từ một node bị lỗi sang một node khác lưu trữ một bản sao của dữ liệu. Điều này cung cấp dịch vụ liên tục cho người dùng. Tuy nhiên, điều cần thiết là một lỗi node không làm mất cân bằng tải, ví dụ, bằng cách tăng gấp đôi tải trên bản sao có sẵn. Giải pháp cho vấn đề này yêu cầu phân vùng các bản sao theo cách mà chúng cũng có thể được truy cập song song.
3. **Khả năng mở rộng**: Trong một hệ thống song song, việc tăng kích thước cơ sở dữ liệu hoặc tăng nhu cầu hiệu suất (ví dụ: thông lượng) sẽ dễ dàng hơn. Khả năng mở rộng là khả năng mở rộng hệ thống một cách trơn tru bằng cách thêm sức mạnh xử lý và lưu trữ vào hệ thống. Hệ thống cơ sở dữ liệu song song sẽ chứng minh hai lợi thế mở rộng: tăng tốc tuyến tính và tăng quy mô tuyến tính, xem **Hình 1.1**. Tăng tốc tuyến tính đề cập đến việc tăng hiệu suất tuyến tính cho kích thước cơ sở dữ liệu không đổi trong khi số lượng nodes (tức là, sức mạnh xử lý và lưu trữ) được tăng tuyến tính. Quy mô tuyến tính đề cập đến một hiệu suất được duy trì để tăng tuyến tính cả về kích thước cơ sở dữ liệu và số lượng nodes. Hơn nữa, việc mở rộng hệ thống cần yêu cầu tổ chức lại cơ sở dữ liệu hiện có.



**Hình 1.1**: Mở rộng

# Kiến trúc chức năng:

Giả sử kiến trúc client / server, các chức năng được hỗ trợ bởi hệ thống cơ sở dữ liệu song song có thể được chia thành ba hệ thống con giống như trong một DBMS điển hình. Mặc dù vậy, sự khác biệt phải liên quan đến việc thực hiện các chức năng này, nó phải xử lý song song, phân vùng dữ liệu và sao chép và giao dịch phân tán. Tùy thuộc vào kiến trúc, một bộ xử lý node có thể hỗ trợ tất cả (hoặc một tập hợp con) của các hệ thống con này. **Hình 1.2** cho thấy kiến trúc sử dụng các hệ thống con này.



**Hình 1.2:** Kiến trúc chung của một hệ thống cơ sở dữ liệu song song.

1. **Quản lý phiên:** đóng vai trò giám sát giao dịch, cung cấp hỗ trợ cho các tương tác của khách hàng với server. Cụ thể, nó thực hiện các kết nối và ngắt kết nối giữa các tiến trình client và hai hệ thống con khác. Do đó, nó khởi tạo và đóng phiên người dùng (có thể chứa nhiều giao dịch). Trong trường hợp phiên OLTP, trình quản lý phiên có thể kích hoạt thực thi mã giao dịch được tải sẵn trong các mô-đun quản lý dữ liệu.
2. **Quản lý giao dịch**: nhận các giao dịch của khách hàng liên quan đến việc biên dịch và thực hiện truy vấn. Nó có thể truy cập thư mục cơ sở dữ liệu chứa tất cả thông tin meta về dữ liệu và chương trình. Bản thân thư mục nên được quản lý như một cơ sở dữ liệu trong máy chủ. Tùy thuộc vào giao dịch, nó kích hoạt các giai đoạn biên dịch khác nhau, kích hoạt thực thi truy vấn và trả về kết quả cũng như mã lỗi cho ứng dụng khách hàng. Bởi vì nó giám sát việc thực hiện giao dịch và cam kết, nó có thể kích hoạt quy trình phục hồi trong trường hợp giao dịch không thành công. Để tăng tốc độ thực hiện truy vấn, nó có thể tối ưu hóa và song song truy vấn tại thời gian biên dịch.
3. **Quản lý dữ liệu:** Nó cung cấp tất cả các hàm cấp thấp cần thiết để chạy song song các truy vấn đã biên dịch, nghĩa là thực thi toán tử cơ sở dữ liệu, hỗ trợ giao dịch song song, quản lý bộ đệm, v.v. Nếu người quản lý giao dịch có khả năng biên dịch được sự kiểm soát dòng dữ liệu, sau đó đồng bộ hóa và giao tiếp giữa các mô đun quản lý dữ liệu là có thể. Mặt khác, kiểm soát giao dịch và đồng bộ hóa phải được thực hiện bởi một mô-đun quản lý giao dịch.

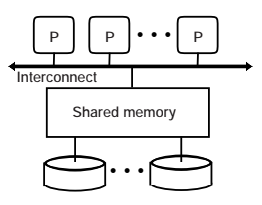
# Kiến trúc DBMS song song:

# Như bất kỳ hệ thống nào, một hệ thống cơ sở dữ liệu song song thể hiện sự thỏa hiệp trong các lựa chọn thiết kế nhằm cung cấp các lợi thế đã nói ở trên với chi phí / hiệu suất tốt. Một quyết định thiết kế hướng dẫn là cách các yếu tố phần cứng chính, tức là bộ xử lý, bộ nhớ chính và đĩa, được kết nối thông qua một số mạng kết nối nhanh. Có ba kiến trúc máy tính song song cơ bản tùy thuộc vào cách chia sẻ bộ nhớ chính hoặc ổ đĩa: shared-memory, shared-disk and shared-nothing.. Các kiến trúc lai như NUMA hoặc cụm cố gắng kết hợp các lợi ích của các kiến trúc cơ bản. Trong phần còn lại của mục này, khi mô tả các kiến trúc song song, chúng tôi tập trung vào bốn yếu tố phần cứng chính: kết nối, bộ xử lý (P), bộ nhớ chính (M) và đĩa. Để đơn giản, chúng tôi bỏ qua các yếu tố khác như bộ đệm của bộ xử lý và I / O bus.

* + - 1. **Shared-Memory:**

Trong cách tiếp cận bộ nhớ dùng chung (Hình 14.3), bất kỳ bộ xử lý nào cũng có quyền truy cập vào bất kỳ mô-đun bộ nhớ hoặc đơn vị đĩa thông qua kết nối nhanh (ví dụ: bus tốc độ cao hoặc cross-bar switch). Tất cả các bộ xử lý đều nằm dưới sự kiểm soát của một hệ điều hành.

Các thiết kế máy tính lớn hiện tại và bộ đa xử lý đối xứng (SMP) tuân theo phương pháp này. Ví dụ về các hệ thống cơ sở dữ liệu song song bộ nhớ chia sẻ bao gồm XPRS, DBS3 và Volcano, cũng như các phần của DBMS thương mại chính trên SMP. Theo một nghĩa nào đó, việc triển khai DB2 trên IBM3090 với 6 bộ xử lý là ví dụ đầu tiên. Tất cả các sản phẩm cơ sở dữ liệu song song bộ nhớ chia sẻ ngày nay có thể khai thác song song truy vấn để cung cấp thông lượng giao dịch cao và song song truy vấn nội bộ để giảm thời gian phản hồi của các truy vấn hỗ trợ quyết định.



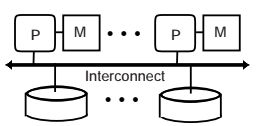
**Hình 14.3**: Kiến trúc Shared-memory.

Bộ nhớ chia sẻ có hai ưu điểm mạnh: đơn giản và độ tải cân bằng. Vì thông tin meta (thư mục) và thông tin điều khiển (ví dụ: bảng khóa) có thể được chia sẻ bởi tất cả các bộ xử lý, nên việc viết phần mềm cơ sở dữ liệu không khác lắm so với các máy tính xử lý đơn. Đặc biệt, song song truy vấn đến miễn phí. Song song truy vấn nội bộ đòi hỏi một số song song nhưng vẫn khá đơn giản. Cân bằng tải rất dễ đạt được vì có thể đạt được trong thời gian chạy bằng cách sử dụng bộ nhớ dùng chung bằng cách phân bổ từng tác vụ mới cho bộ xử lý ít bận nhất.

Shared-memory có ba vấn đề: chi phí cao, khả năng mở rộng hạn chế và tính sẵn sàng thấp. Chi phí cao phát sinh bởi kết nối đòi hỏi phần cứng khá phức tạp do phải liên kết từng bộ xử lý với từng mô-đun bộ nhớ hoặc đĩa. Với bộ xử lý nhanh hơn (ngay cả với bộ nhớ cache lớn hơn), các truy cập xung đột vào shared-memory tăng nhanh và làm giảm hiệu suất. Do đó, khả năng mở rộng được giới hạn ở một vài chục bộ xử lý, thường là tối đa 16 cho chi phí / hiệu suất tốt nhất bằng cách sử dụng bo mạch 4 bộ xử lý. Cuối cùng, do không gian bộ nhớ được chia sẻ bởi tất cả các bộ xử lý, lỗi bộ nhớ có thể ảnh hưởng đến hầu hết các bộ xử lý ở đó bằng cách làm tổn thương tính khả dụng. Giải pháp là sử dụng bộ nhớ song công với kết nối dự phòng.

* + - 1. **Shared-Disk:**

Trong cách tiếp cận shared-disk (Hình 14.4), bất kỳ bộ xử lý nào cũng có quyền truy cập vào bất kỳ đơn vị đĩa nào thông qua truy cập kết nối nhưng độc quyền (không chia sẻ) vào bộ nhớ chính của nó. Mỗi nút xử lý bộ nhớ nằm dưới sự kiểm soát của bản sao hệ điều hành riêng. Sau đó, mỗi bộ xử lý có thể truy cập các trang cơ sở dữ liệu trên đĩa được chia sẻ và lưu trữ chúng vào bộ nhớ riêng. Vì các bộ xử lý khác nhau có thể truy cập vào cùng một trang trong các chế độ cập nhật xung đột, tính nhất quán của bộ đệm chung là cần thiết. DBMS song song đầu tiên sử dụng shared-disk là Oracle với việc triển khai hiệu quả trình quản lý khóa phân tán để thống nhất bộ đệm. Các nhà cung cấp DBMS lớn khác như IBM, Microsoft và Sybase cung cấp các triển khai shared-disk.



**Hình 14.4:** Kiến trúc Shared-disk.

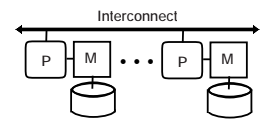
Shared-disk có một số lợi thế: chi phí thấp hơn, khả năng mở rộng cao, cân bằng tải, tính sẵn có và dễ dàng di chuyển từ các hệ thống tập trung. Chi phí kết nối thấp hơn đáng kể so với shared-memory vì công nghệ bus tiêu chuẩn có thể được sử dụng. Cho rằng mỗi bộ xử lý có đủ bộ nhớ chính, can thiệp vào shared disk có thể được giảm thiểu. Do đó, khả năng mở rộng có thể tốt hơn, điển hình lên tới cả trăm bộ xử lý. Vì lỗi bộ nhớ có thể được phân lập từ các node khác, tính khả dụng có thể cao hơn. Cuối cùng, việc di chuyển từ một hệ thống tập trung sang shared-disk tương đối đơn giản vì dữ liệu trên đĩa không cần phải được sắp xếp lại.

Shared-disk chịu sự phức tạp cao hơn và các vấn đề hiệu suất tiềm năng. Nó đòi hỏi các giao thức hệ thống cơ sở dữ liệu phân tán, chẳng hạn như khóa phân tán và cam kết hai pha. Như chúng ta đã thảo luận trong các chương trước, chúng có thể phức tạp. Hơn nữa, việc duy trì tính nhất quán của bộ đệm có thể phải chịu chi phí truyền thông cao giữa các node. Cuối cùng, truy cập vào Shared-disk là một nút cổ chai tiềm năng.

* + - 1. **Shared-Nothing:**

Trong phương pháp Shared-Nothing (Hình 14,5), mỗi bộ xử lý có quyền truy cập độc quyền vào bộ nhớ chính và các đơn vị đĩa. Tương tự như shared-disk, mỗi node bộ nhớ xử lý nằm dưới sự kiểm soát của bản sao hệ điều hành riêng. Sau đó, mỗi node có thể được xem như một trang cục bộ (có cơ sở dữ liệu và phần mềm riêng) trong một hệ thống cơ sở dữ liệu phân tán. Do đó, hầu hết các giải pháp được thiết kế cho cơ sở dữ liệu phân tán như phân mảnh cơ sở dữ liệu, quản lý giao dịch phân tán và xử lý truy vấn phân tán có thể được sử dụng lại. Sử dụng kết nối nhanh, có thể chứa số lượng lớn các node. Trái ngược với SMP, kiến trúc này thường được gọi là Bộ xử lý song song lớn (MPP).

Nhiều nguyên mẫu nghiên cứu đã áp dụng kiến trúc không chia sẻ, ví dụ, BUBBA, EDS, GAMMA, GRACE và PRISMA , vì nó có thể mở rộng. Sản phẩm DBMS song song lớn đầu tiên là Máy tính cơ sở dữ liệu Teradata có thể chứa một nghìn bộ xử lý trong phiên bản đầu tiên. Các nhà cung cấp DBMS lớn khác như IBM, Microsoft và Sybase cung cấp các triển khai không chia sẻ.



**Hình 14.5:** Kiến trúc Shared-Nothing.

Như được chứng minh bởi các sản phẩm hiện có, shared-nothing có ba ưu điểm chính: chi phí thấp hơn, khả năng mở rộng cao và tính sẵn sàng cao. Lợi thế chi phí tốt hơn so với shared-disk, nó yêu cầu kết nối đặc biệt cho các đĩa. Bằng cách thực hiện một thiết kế cơ sở dữ liệu phân tán, tạo điều kiện cho sự phát triển gia tăng trơn tru của hệ thống bằng cách thêm các nút mới, khả năng mở rộng có thể tốt hơn (trong hàng ngàn node). Với việc phân vùng dữ liệu cẩn thận trên nhiều đĩa, có thể đạt được tốc độ tuyến tính và tăng tốc tuyến tính cho các khối lượng công việc đơn giản. Cuối cùng, bằng cách sao chép dữ liệu trên nhiều node, tính sẵn sàng cao cũng có thể đạt được.

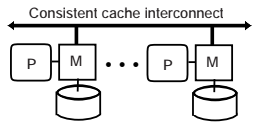
Shared-nothing phức tạp hơn để quản lý so với shared-memory hoặc shared-disk. Độ phức tạp cao hơn là do việc thực hiện cần thiết các hàm cơ sở dữ liệu phân tán giả định số lượng lớn các node. Ngoài ra, cân bằng tải khó khăn hơn để đạt được vì nó phụ thuộc vào hiệu quả của phân vùng cơ sở dữ liệu cho các khối lượng công việc truy vấn. Không giống như bộ nhớ chia sẻ và đĩa chia sẻ, cân bằng tải được quyết định dựa trên vị trí dữ liệu chứ không phải tải thực tế của hệ thống. Hơn nữa, việc bổ sung các node mới trong hệ thống có lẽ yêu cầu tổ chức lại cơ sở dữ liệu để giải quyết các vấn đề cân bằng tải.

* + - 1. **Kiến trúc lai**

Các kết hợp khác nhau của ba kiến trúc cơ bản có thể có được sự đánh đổi khác nhau giữa chi phí, hiệu suất, khả năng mở rộng, tính sẵn có, v.v. và chi phí của shared-memory hoặc shared nothing. Trong phần này, chúng tôi thảo luận về hai kiến trúc lai phổ biến: NUMA và cụm.

**NUMA**

Với shared-memory, mỗi bộ xử lý có quyền truy cập bộ nhớ đồng nhất (UMA), với thời gian truy cập không đổi, vì cả bộ nhớ ảo và bộ nhớ vật lý đều được chia sẻ. Một lợi thế lớn là mô hình lập trình dựa trên bộ nhớ ảo được chia sẻ rất đơn giản. Với shared-disk hoặc shared-nothing, cả bộ nhớ ảo và bộ nhớ chung đều được phân phối, mang lại khả năng mở rộng cho số lượng lớn bộ xử lý. Mục tiêu của NUMA là cung cấp mô hình lập trình bộ nhớ dùng chung và tất cả các lợi ích của nó, trong một kiến ​​trúc có thể mở rộng với bộ nhớ phân tán. Thuật ngữ NUMA phản ánh thực tế rằng việc truy cập vào shared-memory (hầu như) có thể có chi phí khác nhau tùy thuộc vào việc bộ nhớ vật lý là cục bộ hay từ xa đến bộ xử lý. Lớp đa bộ xử lý NUMA thành công nhất là Cache Coherent NUMA (CC-NUMA) [Goodman và Woest, 1988; Lenoski và cộng sự, 1992]. Với CC-NUMA, bộ nhớ chính được phân phối vật lý giữa các node như với shared-nothing hoặc shared-disk. Tuy nhiên, bất kỳ bộ xử lý nào cũng có quyền truy cập vào tất cả các bộ nhớ khác của bộ xử lý (xem Hình 14.6). Mỗi node có thể là một SMP. Tương tự shared-disk, các bộ xử lý khác nhau có thể truy cập cùng một dữ liệu trong chế độ cập nhật xung đột, do đó cần có các giao thức nhất quán bộ đệm chung. Để làm cho việc truy cập bộ nhớ từ xa hiệu quả, giải pháp khả thi duy nhất là thực hiện tính nhất quán của bộ đệm trong phần cứng thông qua kết nối bộ đệm nhất quán đặc biệt [Lenoski et al., 1992]. Vì tính nhất quán của bộ nhớ chia sẻ và bộ nhớ cache được hỗ trợ bởi phần cứng, truy cập bộ nhớ từ xa rất hiệu quả, chỉ vài lần (thường từ 2 đến 3 lần) chi phí truy cập cục bộ.



**Hình 14.6:** Bộ nhớ cache kết hợp NUMA (CC-NUMA).

Hầu hết các nhà sản xuất SMP hiện đang cung cấp hệ thống NUMA có thể mở rộng lên đến hàng trăm bộ xử lý. Lập luận mạnh mẽ cho NUMA là nó không yêu cầu viết lại phần mềm ứng dụng. Tuy nhiên, một số cách viết lại vẫn cần thiết trong công cụ cơ sở dữ liệu (và hệ điều hành) để tận dụng tối đa lợi thế của địa phương truy cập .

**Cluster**

Một cluster là một tập hợp các node máy chủ độc lập được kết nối với nhau để chia sẻ tài nguyên và tạo thành một hệ thống duy nhất. Các tài nguyên được chia sẻ, được gọi là tài nguyên cụm, có thể là phần cứng như đĩa hoặc phần mềm như dịch vụ quản lý dữ liệu. Các node máy chủ được tạo thành từ các thành phần ngoài kệ, từ các thành phần PC đơn giản đến SMP mạnh hơn. Sử dụng nhiều thành phần sẵn có là điều cần thiết để có được tỷ lệ chi phí / hiệu suất tốt nhất trong khi khai thác sự tiến bộ liên tục trong các thành phần phần cứng. Ở dạng rẻ nhất, kết nối có thể là một mạng cục bộ. Tuy nhiên, hiện nay có các kết nối tiêu chuẩn nhanh cho các cụm (ví dụ: Myrinet và Infiniband) cung cấp băng thông cao (Gigabits / giây) với độ trễ thấp cho lưu lượng tin nhắn.

So với một hệ thống phân tán, một cụm được tập trung về mặt địa lý (tại một địa điểm) và được làm bằng các nút đồng nhất. Kiến trúc của nó có thể được chia sẻ không có gì hoặc chia sẻ đĩa. Shared-nothing clusters đã được sử dụng rộng rãi vì chúng có thể cung cấp tỷ lệ chi phí / hiệu suất tốt nhất và quy mô lên đến cấu hình rất lớn (hàng nghìn node). Tuy nhiên, vì mỗi đĩa được kết nối trực tiếp với máy tính thông qua xe buýt, nên việc thêm hoặc thay thế các nút cụm yêu cầu sắp xếp lại đĩa và dữ liệu. Shared-disk tránh việc sắp xếp lại như vậy nhưng yêu cầu các đĩa phải được truy cập toàn cầu bởi các nút cụm. Có hai công nghệ chính để chia sẻ đĩa trong một cụm: lưu trữ gắn mạng (NAS) và mạng vùng lưu trữ (SAN). NAS là một thiết bị chuyên dụng cho các đĩa được chia sẻ qua mạng (thường là TCP / IP) bằng cách sử dụng giao thức hệ thống tệp phân tán, chẳng hạn như Hệ thống tệp mạng (NFS). NAS rất phù hợp cho các ứng dụng thông lượng thấp như sao lưu dữ liệu và lưu trữ từ đĩa cứng PC. Tuy nhiên, nó tương đối chậm và không phù hợp để quản lý cơ sở dữ liệu vì nó nhanh chóng trở thành nút cổ chai với nhiều node. Mạng vùng lưu trữ (SAN) cung cấp chức năng tương tự nhưng với giao diện cấp thấp hơn. Để hiệu quả, nó sử dụng giao thức dựa trên khối do đó giúp quản lý tính nhất quán của bộ đệm (ở cấp độ khối) dễ dàng hơn. Trên thực tế, các đĩa trong SAN được gắn vào mạng thay vì bus như trong lưu trữ được đính kèm trực tiếp (DAS), nhưng nếu không, chúng được xử lý như các đĩa cục bộ có thể chia sẻ. Các giao thức hiện tại cho SAN mở rộng các đối tác đĩa cục bộ của chúng để chạy qua mạng (ví dụ: i-SCSI mở rộng SCSI và ATA-over-Ethernet mở rộng ATA). Do đó, SAN cung cấp thông lượng dữ liệu cao và có thể mở rộng lên đến số lượng lớn các nút. Giới hạn duy nhất của nó đối với việc shared-nothing là chi phí sở hữu cao hơn.

Một kiến ​​trúc cụm có những lợi thế quan trọng. Nó kết hợp tính linh hoạt và hiệu suất của bộ nhớ chia sẻ tại mỗi nút với khả năng mở rộng và tính sẵn có của shared-nothing hoặc shared-disk. Hơn nữa, việc sử dụng các node shared-memory ngoài luồng với kết nối cụm tiêu chuẩn giúp nó trở thành một giải pháp thay thế hiệu quả về mặt chi phí cho các bộ xử lý đa cấp cao cấp độc quyền như NUMA hoặc MPP. Cuối cùng, sử dụng SAN giúp giảm bớt việc quản lý đĩa và vị trí dữ liệu.

* + - 1. **Thảo luận:**

Chúng ta hãy so sánh ngắn gọn ba kiến ​​trúc cơ bản dựa trên những lợi thế tiềm năng của chúng (hiệu suất cao, tính sẵn sàng cao và khả năng mở rộng). Công bằng mà nói, đối với một cấu hình nhỏ (ví dụ: dưới 20 bộ xử lý), shared-memory có thể cung cấp hiệu suất cao nhất vì cân bằng tải tốt hơn. Kiến trúc Shared-disk và shared-nothing có gì vượt trội hơn so với shared-memory về khả năng mở rộng. Vài năm trước, shared-nothing là lựa chọn duy nhất cho các hệ thống cao cấp. Tuy nhiên, tiến bộ gần đây trong các công nghệ kết nối đĩa như SAN làm cho đĩa chia sẻ trở thành một lựa chọn khả thi với ưu điểm chính là đơn giản hóa việc quản trị dữ liệu và triển khai DBMS. Đặc biệt, Shared-disk hiện là kiến ​​trúc ưa thích cho các ứng dụng OLTP vì việc hỗ trợ các giao dịch ACID và kiểm soát tương tranh phân tán dễ dàng hơn. Nhưng đối với cơ sở dữ liệu OLAP thường rất lớn và chủ yếu chỉ đọc, shared-nothing là kiến ​​trúc được ưa thích. Hầu hết các nhà cung cấp DBMS lớn hiện cung cấp triển khai DBMS không chia sẻ gì cho OLAP, ngoài phiên bản đĩa chia sẻ cho OLTP. Việc thực thi duy nhất là Oracle sử dụng shared-disk cho cả OLTP và OLAP.

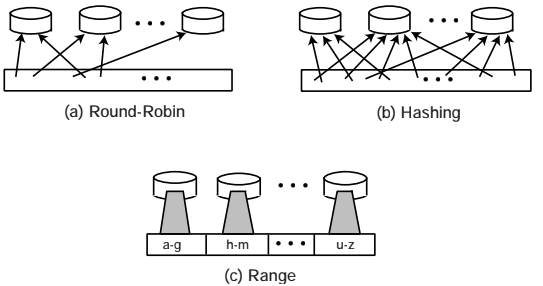
Các kiến ​​trúc lai, chẳng hạn như NUMA và cluster, có thể kết hợp hiệu quả và tính đơn giản của bộ nhớ dùng chung và khả năng mở rộng và chi phí của shared-disk hoặc shared-nothing. Đặc biệt, họ có thể khai thác tiến trình liên tục trong SMP và sử dụng các node shared-memory với tỷ lệ chi phí / hiệu suất tuyệt vời. Cả NUMA và cluster có thể mở rộng lên đến cấu hình lớn (hàng trăm node). Ưu điểm chính của NUMA so với cụm là mô hình lập trình đơn giản (shared-memory) giúp dễ dàng quản trị và điều chỉnh cơ sở dữ liệu. Tuy nhiên, bằng cách sử dụng các node và kết nối PC tiêu chuẩn, các cụm cung cấp tỷ lệ chi phí / hiệu suất tổng thể tốt hơn và bằng cách sử dụng shared-nothing, chúng có thể mở rộng đến cấu hình rất lớn (hàng nghìn node).

# Vị trí dữ liệu song song:

Trong phần này, chúng tôi giả định một kiến ​​trúc shared-nothing vì đây là trường hợp chung nhất và các kỹ thuật triển khai của nó cũng được áp dụng, đôi khi ở dạng đơn giản hóa cho các kiến ​​trúc khác. Vị trí dữ liệu trong một hệ thống cơ sở dữ liệu song song thể hiện tương đồng với phân mảnh dữ liệu trong cơ sở dữ liệu phân tán. Một điểm tương đồng rõ ràng là sự phân mảnh có thể được sử dụng để tăng tính song song. Trong phần tiếp theo, chúng tôi sử dụng thuật ngữ phân vùng và phân vùng thay vì phân mảnh ngang và phân mảnh ngang, tương phản với chiến lược thay thế, bao gồm phân cụm một mối quan hệ tại một node. Thuật ngữ giảm dần đôi khi được sử dụng để phân vùng. Phân mảnh dọc cũng có thể được sử dụng để tăng tính song song và cân bằng tải nhiều như trong cơ sở dữ liệu phân tán. Một điểm tương đồng khác là vì dữ liệu lớn hơn nhiều so với các chương trình, nên việc thực thi sẽ xảy ra, càng nhiều càng tốt, nơi dữ liệu cư trú. Tuy nhiên, có hai sự khác biệt quan trọng với cách tiếp cận cơ sở dữ liệu phân tán. Đầu tiên, không cần tối đa hóa xử lý cục bộ (tại mỗi node) vì người dùng không được liên kết với các node cụ thể. Thứ hai, cân bằng tải khó khăn hơn nhiều để đạt được với sự có mặt của một số lượng lớn các node. Vấn đề chính là để tránh sự tranh chấp tài nguyên, điều này có thể dẫn đến toàn bộ hệ thống bị đập (ví dụ: một node kết thúc thực hiện tất cả công việc trong khi các node khác không hoạt động). Vì các chương trình được thực thi khi dữ liệu cư trú, vị trí dữ liệu là một vấn đề hiệu suất quan trọng.

Vị trí dữ liệu phải được thực hiện để tối đa hóa hiệu suất hệ thống, có thể được đo bằng cách kết hợp tổng số lượng công việc được thực hiện bởi hệ thống và thời gian phản hồi của các truy vấn riêng lẻ. Song song liên truy vấn dẫn đến tổng công việc tăng lên. Mặt khác, việc phân cụm tất cả dữ liệu cần thiết cho một chương trình sẽ giảm thiểu việc giao tiếp và do đó, toàn bộ công việc được hệ thống thực hiện khi thực hiện chương trình đó. Về vị trí dữ liệu, chúng tôi có sự đánh đổi sau: tối đa hóa thời gian phản hồi hoặc song song truy vấn liên kết dẫn đến phân vùng, trong khi giảm thiểu tổng số lượng công việc dẫn đến phân cụm. Quản trị viên cơ sở dữ liệu chịu trách nhiệm kiểm tra định kỳ tần suất truy cập đoạn và khi cần thiết, di chuyển và sắp xếp lại các đoạn.

Một giải pháp thay thế cho vị trí dữ liệu là phân vùng đầy đủ, theo đó mỗi quan hệ được phân chia theo chiều ngang trên tất cả các nút trong hệ thống. Có ba chiến lược cơ bản để phân vùng dữ liệu: vòng tròn, băm và phân vùng phạm vi (Hình 14.7).



**Hình 14.7 :** Các lược đồ phân vùng khác nhau.

**Phân vùng vòng tròn :** là chiến lược đơn giản nhất, nó đảm bảo phân phối dữ liệu thống nhất. Với n phân vùng, bộ thứ tự thứ i được gán cho phân vùng (i mod n). Chiến lược này cho phép truy cập tuần tự vào một mối quan hệ được thực hiện song song. Tuy nhiên, việc truy cập trực tiếp vào các bộ dữ liệu riêng lẻ, dựa trên một vị ngữ, yêu cầu truy cập toàn bộ quan hệ.

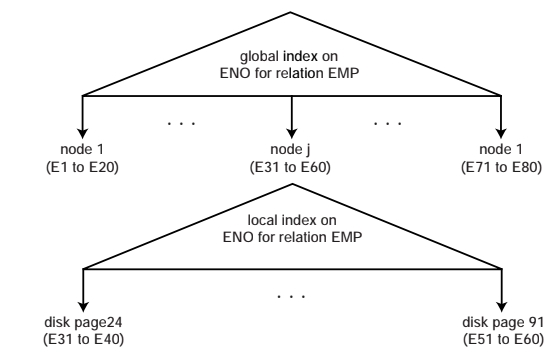
**Phân vùng băm :** áp dụng hàm băm cho một số thuộc tính mang lại số phân vùng. Chiến lược này cho phép các truy vấn khớp chính xác trên thuộc tính lựa chọn được xử lý bởi chính xác một nút và tất cả các truy vấn khác được xử lý song song bởi tất cả các node.

**Phân vùng phạm vi** : phân phối các bộ dữ liệu dựa trên các khoảng giá trị (phạm vi) của một số thuộc tính. Ngoài việc hỗ trợ các truy vấn khớp chính xác (như băm), nó rất phù hợp cho các truy vấn phạm vi. Chẳng hạn, một truy vấn có một biến vị ngữ A giữa A1 và A2 có thể được xử lý bởi các node duy nhất chứa các bộ dữ liệu có giá trị A nằm trong phạm vi [A1, A2]. Tuy nhiên, phân vùng phạm vi có thể dẫn đến sự thay đổi cao trong kích thước phân vùng.

So với quan hệ phân cụm trên một đĩa đơn (có thể rất lớn), phân vùng đầy đủ mang lại hiệu suất tốt hơn . Mặc dù phân vùng đầy đủ có lợi thế hiệu suất rõ ràng, việc thực thi song song cao có thể gây ra chi phí hiệu năng nghiêm trọng cho các truy vấn phức tạp liên quan đến các phép nối. Hơn nữa, phân vùng đầy đủ không thích hợp cho các mối quan hệ nhỏ trải qua một vài khối đĩa. Những hạn chế cho thấy cần phải tìm ra sự thỏa hiệp giữa phân cụm và phân vùng đầy đủ (nghĩa là phân vùng biến).

Một giải pháp là thực hiện vị trí dữ liệu bằng cách phân vùng biến. Mức độ phân vùng, tức là, số lượng node mà mối quan hệ bị phân mảnh, là một hàm của kích thước và tần suất truy cập của mối quan hệ. Chiến lược này liên quan nhiều hơn so với phân cụm hoặc phân vùng đầy đủ vì những thay đổi trong phân phối dữ liệu có thể dẫn đến việc sắp xếp lại. Ví dụ, một mối quan hệ ban đầu được đặt trên tám nút có thể có số lượng thẻ của nó tăng gấp đôi bởi các lần chèn tiếp theo, trong trường hợp đó nên được đặt trên 16 node. Trong một hệ thống song song cao với phân vùng thay đổi, việc sắp xếp lại định kỳ để cân bằng tải là rất cần thiết và nên thường xuyên trừ khi khối lượng công việc khá tĩnh và chỉ có một vài cập nhật. Việc sắp xếp lại như vậy sẽ được minh bạch đối với các chương trình được biên dịch chạy trên máy chủ cơ sở dữ liệu. Đặc biệt, các chương trình không nên được biên dịch lại do sắp xếp lại. Do đó, các chương trình được biên dịch nên độc lập với vị trí dữ liệu, có thể thay đổi nhanh chóng. Sự độc lập như vậy có thể đạt được nếu hệ thống thời gian chạy hỗ trợ truy cập liên kết vào dữ liệu phân tán. Điều này khác với một DBMS phân tán, nơi mà truy cập kết hợp đạt được tại thời điểm biên dịch bởi bộ xử lý truy vấn bằng thư mục dữ liệu. Một giải pháp cho truy cập liên kết là có một cơ chế chỉ mục toàn cầu được nhân rộng trên mỗi node . Chỉ số toàn cầu cho biết vị trí của một mối quan hệ trên một tập hợp các nút. Về mặt khái niệm, chỉ số toàn cầu là một chỉ mục hai cấp với một cụm chính trên tên quan hệ và một cụm nhỏ trên một số thuộc tính của mối quan hệ. Chỉ số toàn cầu này hỗ trợ phân vùng biến, trong đó mỗi quan hệ có một mức độ phân vùng khác nhau. Cấu trúc chỉ mục có thể dựa trên băm hoặc trên một tổ chức giống như cây B . Trong cả hai trường hợp, các truy vấn khớp chính xác có thể được xử lý hiệu quả với một lần truy cập nút. Tuy nhiên, với băm, các truy vấn phạm vi được xử lý bằng cách truy cập tất cả các nút có chứa dữ liệu từ yêu cầu truy vấn r. Sử dụng chỉ mục cây B (thường lớn hơn nhiều so với chỉ mục băm) cho phép xử lý các truy vấn phạm vi hiệu quả hơn, trong đó chỉ các nút chứa dữ liệu trong phạm vi được chỉ định mới được truy cập.

Ví dụ 14.1. Hình 14.8 cung cấp một ví dụ về chỉ mục toàn cầu và chỉ mục cục bộ cho mối quan hệ EMP (ENO, ENAME, DEPT, TITLE) của ví dụ cơ sở dữ liệu kỹ thuật mà chúng tôi đã sử dụng trong cuốn sách này.



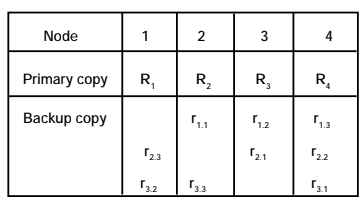
**Hình 14.8 :** Ví dụ về chỉ số toàn cầu và địa phương.

Giả sử rằng chúng tôi muốn xác định vị trí các phần tử trong mối quan hệ EMP với giá trị ENO, E50, E50. Chỉ mục cấp đầu tiên trên tên tập hợp ánh xạ tên EMP lên chỉ mục trên thuộc tính ENO cho quan hệ EMP. Sau đó, chỉ mục cấp hai ánh xạ thêm giá trị cụm E50 E vào số nút j. Một chỉ mục cục bộ trong mỗi nút cũng là cần thiết để ánh xạ một mối quan hệ lên một tập hợp các trang đĩa trong nút. Chỉ mục cục bộ có hai cấp độ, với một cụm chính về tên quan hệ và một cụm nhỏ trên một số thuộc tính. Thuộc tính phân cụm nhỏ cho chỉ mục cục bộ giống như thuộc tính toàn cầu. Do đó, định tuyến kết hợp được cải thiện từ nút này sang nút khác dựa trên (tên quan hệ, giá trị cụm). Chỉ mục cục bộ này ánh xạ thêm giá trị cụm này ‘E5’ vào trang số 91.

Kết quả thử nghiệm cho phân vùng biến đổi của một khối lượng công việc bao gồm một hỗn hợp các giao dịch ngắn (tín dụng ghi nợ) và các giao dịch phức tạp chỉ ra rằng khi phân vùng được tăng lên, thông lượng tiếp tục tăng cho các giao dịch ngắn. Tuy nhiên, đối với các giao dịch phức tạp liên quan đến một số liên kết lớn, phân vùng tiếp theo sẽ giảm thông lượng do chi phí liên lạc.

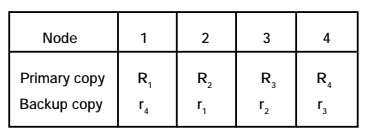
Một vấn đề nghiêm trọng trong việc sắp xếp dữ liệu là xử lý các phân phối dữ liệu bị lệch có thể dẫn đến phân vùng không dạng và làm mất cân bằng tải. Phân vùng phạm vi nhạy cảm với độ lệch hơn so với phân vùng vòng tròn hoặc phân vùng băm. Một giải pháp là xử lý các phân vùng không đồng nhất một cách thích hợp, ví dụ, bằng cách phân đoạn thêm các phân vùng lớn. Sự tách biệt giữa các nút logic và vật lý cũng hữu ích vì một nút logic có thể tương ứng với một số node vật lý.

Một yếu tố phức tạp cuối cùng là sao chép dữ liệu cho tính sẵn sàng cao. Giải pháp đơn giản là duy trì hai bản sao của cùng một dữ liệu, một bản sao chính và bản sao lưu, trên hai nút riêng biệt. Đây là kiến ​​trúc đĩa nhân đôi được quảng bá bởi nhiều nhà sản xuất máy tính. Tuy nhiên, trong trường hợp lỗi nút, tải của nút với bản sao có thể tăng gấp đôi, do đó làm tổn thương cân bằng tải. Để tránh vấn đề này, một số chiến lược sao chép dữ liệu có tính sẵn sàng cao đã được đề xuất cho các hệ thống cơ sở dữ liệu song. Một giải pháp thú vị là phân vùng xen kẽ Teradata, phân vùng tiếp theo phân vùng sao lưu trên một số node. Hình 14.9 minh họa phân vùng xen kẽ của mối quan hệ R qua bốn nút, trong đó mỗi bản sao chính của phân vùng, ví dụ, R1, được chia thành ba phân vùng, ví dụ: r11, r12 và r13, mỗi node tại một node sao lưu khác nhau. Trong chế độ thất bại, tải của bản sao chính được cân bằng giữa các nút sao lưu dự phòng. Nhưng nếu hai nút không thành công, thì mối quan hệ không thể được truy cập do đó làm tổn thương tính khả dụng. Tái thiết bản sao chính từ các bản sao lưu riêng biệt của nó có thể tốn kém. Trong chế độ bình thường, duy trì tính nhất quán của bản sao cũng có thể tốn kém.



**Hình 14.9 :** Ví dụ về phân vùng xen kẽ.

Một giải pháp tốt hơn là phân vùng chuỗi Gamma, lưu trữ bản sao chính và bản sao lưu trên hai nút liền kề (Hình 14.10). Ý tưởng chính là xác suất hai nút liền kề thất bại thấp hơn nhiều so với xác suất hai nút bất kỳ bị hỏng. Trong chế độ thất bại, tải của nút thất bại và các nút dự phòng được cân bằng giữa tất cả các nút còn lại bằng cách sử dụng cả các nút sao chép chính và dự phòng. Ngoài ra, duy trì tính nhất quán sao chép là rẻ hơn. Một vấn đề mở là làm thế nào để thực hiện vị trí dữ liệu có tính đến sao chép dữ liệu. Tương tự như phân bổ đoạn trong cơ sở dữ liệu phân tán, điều này nên được coi là một vấn đề tối ưu hóa.



**Hình 14.10:** Ví dụ về phân vùng xích

Tài liệu tham khảo

Principles Of Distributed Database Systems - M. Tamer Ozsu, Patrick Valduriez