Chapter 1

1. Kiến trúc máy tính là gì?

Kiến trúc máy tính là nghệ thuật thiết kế máy tính dựa trên các nguyên lý kỹ thuật và các đánh giá hiệu năng định lượng. (định lượng quan trọng nhất 🡺 ra 1 con số)

2. Các thành phần chính của kiến trúc máy tính là gì?

* Kiến trúc tập lệnh: đánh địa chỉ ô nhớ, tập lệnh, mã hóa lệnh…
* Tổ chức: cách kết nối bộ nhớ, cách kết nối các vi xử lý, đường ống, bộ đệm (vi kiến trúc)…
* Phần cứng: giá trị xung nhịp đồng hồ, hệ thống bộ nhớ,…

3. Mục đích của kiến trúc máy tính là gì?

Thiết kế: kiến trúc tập lệnh, tổ chức, phần cứng nhằm đáp ứng các tiêu chí định lượng:

* Các yêu cầu chức năng (ứng dụng máy chủ, học tập, chơi game…)
* Giá cả
* Mức tiêu thụ năng lượng
* Hiệu năng (thời gian thực hiện)
* Tính sẵn sàng
* Tính tin cậy

4. Nêu xu thế công nghệ của CPU, RAM, lưu trữ

* Năng lực xử lý (số transitor trên một chip) tăng gấp đôi sau 18 tháng (luật Moore)
* Kích thước bộ nhớ tăng gấp đôi sau 18 tháng
* Dung lượng đĩa cứng tăng gấp đôi sau 18 tháng
* Tốc độ truy cập đĩa cứng tăng gấp đôi sau 10 năm
* Tốc độ truy cập RAM và đĩa cứng tăng vài % một năm

5. Hiệu năng là gì?

Hiệu năng trong ngữ cảnh công nghệ đề cập đến tốc độ xử lý, sức mạnh tính toán, và khả năng thực hiện các tác vụ một cách linh hoạt mà không gặp trở ngại

Hiệu năng (X) =

6. Nêu các nguyên lý thiết kế định lượng.

* Sử dụng song song hóa để tăng hiệu năng
* Nguyên lý địa phương: phần mềm thường sử dụng dữ liệu và mã lệnh vừa mới được sử dụng gần đây
* Tập trung vào các trường hợp thường xảy ra: các lệnh nạp và giải mã được dùng nhiều hơn các lệnh cộng, nhân -> tập trung vào chúng
* Luật Amdahl
* Công thức tính hiệu năng CPU

7. Phát biểu luật Amdahl.

Hiệu năng tối đa mà một hệ thống có thể đạt được sau khi cải tiến một phần nhất định của nó sẽ bị giới hạn bởi phần tử ít hoạt động nhất trong hệ thống đó

8. CPI là gì?

CPI là một chỉ số đo lường hiệu suất của một bộ xử lý. CPI đo số lần các chu kỳ máy tính cần để thực hiện một lệnh máy tính (instruction cycle) trung bình

9. CPI trung bình là gì?

Do các chỉ lệnh (instruction) là không giống nhau nên có khái niệm CPI trung bình

CPItrung bình = tổng số chu kỳ/tổng số lệnh

Chapter 2

1. Kiến trúc tập lệnh là gì?

* Kiến trúc tập lệnh là giao diện giữa phần cứng và phần mềm
* Một giao diện tốt thỏa mãn các đặc điểm:

- Có tính tương thích và tính khả chuyển

- Có thể dùng theo nhiều cách khác nhau

- Cung cấp đủ chức năng cho các tầng cao

- Cho phép thực hiện một cách hiệu quả ở tầng thấp

2. Kiến trúc load – store là gì. Phân tích ưu điểm, nhược điểm của kiến trúc này với các kiến trúc khác có sử dụng các thanh ghi dùng chung.

* Kiến trúc load-store là một loại kiến trúc trong thiết kế CPU, trong đó các lệnh không thể trực tiếp thao tác trên bộ nhớ. Thay vào đó, chúng chỉ có thể thực hiện trên các thanh ghi (registers). Để thao tác trên bộ nhớ, dữ liệu cần truyền từ thanh ghi vào bộ nhớ, sau đó thực hiện các phép toán, sau khi thực hiện xong thì dữ liệu được lưu trở lại thanh ghi từ bộ nhớ
* So với các kiến trúc khác có sử dụng thanh ghi dùng chung, kiến trúc load-store thường mang lại tính nhất quán và dễ quản lý hơn, nhưng có thể gặp phải overhead và độ trễ cao hơn. Điều này là do việc tải và lưu trữ dữ liệu giữa thanh ghi và bộ nhớ yêu cầu những lệnh phụ trợ.

3. Phân tích ưu điểm, nhược điểm của phương pháp mã hóa lệnh sử dụng độ dài thay đổi và độ dài cố định

* Mã hóa lệnh sử dụng độ dài thay đổi:
* Ưu điểm:

1. Tiết kiệm không gian lưu trữ: Vì độ dài của mỗi lệnh có thể được điều chỉnh tùy thuộc vào yêu cầu của mỗi lệnh nên sẽ tiết kiệm không gian lưu trữ so với việc sử dụng độ dài cố định cho mỗi lệnh.
2. Hiệu quả về mặt băng thông: Có thể tối ưu hóa băng thông hệ thống bằng cách điều chỉnh độ dài của mỗi lệnh dựa trên yêu cầu cụ thể của lệnh đó.

* Nhược điểm:

1. Phức tạp trong giải mã: Việc mã hóa lệnh có độ dài thay đổi có thể phức tạp hơn so với lệnh có độ dài cố định, đặc biệt là trong quá trình xử lý lệnh và điều khiển lệnh.
2. Khó khăn trong việc đồng bộ hóa: Các lệnh có độ dài khác nhau nên có thể gây khó khăn trong quá trình đồng bộ hóa

* Mã hóa lệnh sử dụng độ dài cố định:
* Ưu điểm:

1. Giải mã dễ dàng: Việc giải mã và mã hóa lệnh có độ dài cố định thường dễ dàng hơn so với lệnh có độ dài thay đổi.
2. Đồng bộ hóa dễ dàng: Vì các lệnh có độ dài giống nhau nên việc đồng bộ hóa sẽ dễ dàng hơn.

* Nhược điểm:

1. Tốn không gian lưu trữ: Vì các lệnh có độ dài giống nhau nên sẽ gây lãng phí bộ nhớ, đặc biệt với những lệnh yêu cầu ít hơn độ dài cố định.
2. Không linh hoạt: Việc sử dụng độ dài cố định có thể làm giảm tính linh hoạt trong việc thêm các chức năng mới hoặc mở rộng các tập lệnh.

4. Nêu ví dụ một câu lệnh sử dụng cơ chế địa chỉ thanh ghi (register) và giải thích nội dung câu lệnh đó.

ADD R1, R2, R3

Tính R2 + R3 sau đó lưu giá trị vào R1

5. Nêu ví dụ một câu lệnh sử dụng cơ chế địa chỉ tức thời (immediate) và giải thích nội dung câu lệnh đó.

ADD R1, #10

Giá trị 10 được thêm trực tiếp vào thanh ghi R1

6. Nêu ví dụ một câu lệnh sử dụng cơ chế địa chỉ dịch chuyển (displacement) và giải thích nội dung câu lệnh đó.

ADD R1, R2, 5(R3)

Tính giá trị R2 + giá trị của ô nhớ có địa chỉ [R3 + 5] sau đó lưu vào R1.

7. Nêu vai trò của trình biên dịch

* Chuyển đổi mã nguồn: Trình biên dịch chuyển đổi mã nguồn từ một ngôn ngữ lập trình như C, C++, Java, Python,..) thành mã máy hoặc mã trung gian (như bytecode trong Java) để máy tính có thể hiểu và thực thi.
* Kiểm tra lỗi: kiểm tra lỗi cú pháp (syntax errors) hoặc lỗi logic (logical errors) trong mã nguồn để đảm bảo rằng chương trình được viết đúng cú pháp và hoạt động như mong đợi.
* Tối ưu hóa mã máy
* Tạo tài liệu: Một số trình biên dịch cung cấp tài liệu về mã nguồn, các lỗi phổ biến và cách sửa chúng, giúp người lập trình hiểu hơn về mã nguồn của họ.
* Hỗ trợ nền tảng: Trình biên dịch cung cấp hỗ trợ cho nhiều nền tảng và kiến trúc máy tính khác nhau, cho phép chương trình được biên dịch và chạy trên nhiều loại máy tính khác nhau.
* Phân phối chương trình: tạo ra các tệp thực thi hoặc các tệp bytecode, giúp phân phối chương trình cho người dùng cuối một cách dễ dàng

Chapter 3

1. Nêu khái niệm về đường ống.

Kiến trúc đường ống là một kiểu thiết kế trong kiến trúc máy tính, trong đó các tác vụ của một quá trình xử lý được chia thành các giai đoạn khác nhau và các tác vụ này được thực hiện song song trên các giai đoạn đó. Ý tưởng chính của kiến trúc đường ống là tận dụng thời gian cần thiết để thực hiện một tác vụ bằng cách xử lý tác vụ tiếp theo ngay sau khi tác vụ trước đó đã hoàn thành giai đoạn của mình, thay vì đợi cho đến khi tất cả các tác vụ trước đó hoàn thành.

Các thành phần của một kiến trúc đường ống:

* Giai đoạn (Stages): Mỗi giai đoạn thực hiện một phần nhỏ của quá trình xử lý. Ví dụ, trong một CPU, các giai đoạn có thể bao gồm lấy lệnh, giải mã, thực thi, truy xuất dữ liệu từ bộ nhớ, ghi kết quả.
* Bộ đệm (Buffers): Các bộ đệm dùng để lưu trữ dữ liệu trung gian giữa các giai đoạn để đảm bảo rằng dữ liệu luôn sẵn sàng xử lý khi cần thiết.
* Điều khiển (Control): Điều khiển quá trình xử lý và điều chỉnh việc chuyển đổi giữa các giai đoạn.

2. Nêu 5 bước để thực hiện 1 lệnh RISC

* IF (instruction fetch): Nạp lệnh
* ID (instruction decode/register fetch) : Giải mã lệnh
* EX (Execution): Thực thi
* MEM (Memory access): Truy cập bộ nhớ
* WB (Write back): Ghi kết quả

3. Nêu đặc điểm kỹ thuật đường ống

* Kỹ thuật đường ống không làm giảm thời gian thực hiện một lệnh mà làm giảm thời gian thực hiện toàn bộ khối lượng công việc.
* Nhiều lệnh được xử lý cùng lúc
* Tốc độ đường ống phụ thuộc vào tốc độ của trạng thái chậm nhất.
* Khả năng tốc độ lớn nhất bằng số trạng thái của đường ống.

4. Hazard cấu trúc là gì?

Hazard cấu trúc là một loại xung đột xảy ra khi có hai hoặc nhiều tác vụ cố gắng truy cập cùng một phần cứng của hệ thống trong cùng thời điểm. Điều này có thể xảy ra khi một thành phần cứng không thể chia sẻ một cách hiệu quả giữa các tác vụ đang chạy, dẫn đến sự cản trở và giảm hiệu suất của hệ thống. Vd: cùng một thanh ghi được sử dụng bởi nhiều lệnh trong cùng một thời điểm, bus chỉ có thể truyền tải một lượng dữ liệu hữu hạn mỗi lần,…

5. Kỹ thuật chuyển về phía trước là gì?

* Kết quả của ALU lưu trữ tại thanh ghi pipeline EX/MEM và MEM/WB được gửi lại đầu vào của ALU
* Nếu phần cứng phát hiện thấy kết quả của ALU trước được gửi lại về đầu vào của ALU hiện tại, nó sẽ dùng giữ liệu này mà không dùng dữ liệu từ thanh ghi

Chapter 4

1. Hazard là gì?

Trong kiến trúc máy tính, Hazard là một khái niệm đề cập đến các tình huống gây ra sự trì hoãn (delay) hoặc xung đột trong quá trình thực thi các lệnh, dẫn đến giảm hiệu xuất của hệ thống. Có ba loại Hazard chính:

* Data Hazard (hazard dữ liệu): Xảy ra khi một lệnh cần dùng một giá trị mà lệnh trước đó vẫn chưa tạo ra. Điều này xảy ra trong trường hợp lệnh sau cần dùng kết quả lệnh trước mà lệnh trước vẫn chưa thực thi xong. Do đó, lệnh sau phải đợi cho đến khi kết quả của lệnh trước thực thi xong. Các kỹ thuật như đặt lệnh (instruction scheduling) hoặc sử dụng bộ đệm (buffering) có thể được áp dụng để giảm hazard dữ liệu.
* Control Hazard (hazard kiểm soát): Xảy ra khi quyết định điều kiện của một lệnh nhảy vẫn chưa được xác định tại thời điểm thực thi lệnh. Khi điều này xảy ra, CPU phải chờ đợi cho đến khi kết quả của lệnh nhảy được xác định trước khi có thể thực thi các lệnh tiếp theo. Kỹ thuật như dự đoán lệnh nhảy (branch prediction) được sử dụng để giảm thiểu hazard kiểm soát bằng cách dự đoán kết quả của các lệnh nhảy trước khi chúng thực sự được xác định.
* Structural Hazard (hazard cấu trúc): …Để giải quyết hazard cấu trúc, các kỹ thuật như chia sẻ tài nguyên (resource sharing) có thể được sử dụng.

2. Phân biệt sự khác nhau giữa phụ thuộc tên và phụ thuộc thực sự

1. Phụ thuộc tên:

* Phụ thuộc tên xảy ra khi một lệnh cần sử dụng kết quả của một lệnh trước đó dựa trên tên của thanh ghi hoặc biến mà lệnh trước đã ghi vào.
* Không có sự thay đổi dữ liệu của thanh ghi hoặc biến giữa hai lệnh, chỉ có sự thay đổi về tên của chúng.
* Ví dụ, nếu lệnh A ghi kết quả vào thanh ghi R1 và lệnh B sau đó cần sử dụng kết quả từ R1, ta nói rằng lệnh B phụ thuộc vào lệnh A dựa trên tên

1. Phụ thuộc thực sự:

* Phụ thuộc thực sự xảy ra khi một lệnh cần sử dụng kết quả từ một lệnh trước đó dựa trên dữ liệu thực sự mà lệnh trước đã ghi vào.
* Có sự thay đổi về dữ liệu giữa hai lệnh, nghĩa là kết quả của lệnh trước là đầu vào thực sự của lệnh sau.
* Ví dụ, nếu lệnh A ghi kết quả vào thanh ghi R1 và lệnh B sau đó cần sử dụng kết quả từ R1, và không có lệnh nào khác ghi đè hoặc thay đổi giá trị của R1, ta nói rằng lệnh B phụ thuộc vào lệnh A dựa trên dữ liệu thực sự.

3. Song song hóa mức câu lệnh là gì? Nêu các giải pháp thực hiện song song hóa mức câu lệnh.

* Song song hóa mức câu lệnh là một kỹ thuật trong kiến trúc máy tính, trong đó các câu lệnh độc lập về mặt dữ liệu và không có phụ thuộc giữa chúng được thực thi đồng thời để tăng hiệu xuất xử lý của hệ thống. Mục tiêu của song song hóa mức câu lệnh là tận dụng tất cả tài nguyên và thời gian trống trong máy tính một cách hiệu quả nhất.
* Pipelining là một kỹ thuật song song hóa mức câu lệnh.

4. Nêu các cách để pipeline CPI càng gần pipeline CPI lý tưởng càng tốt

* Giải pháp 1: Sử dụng phần mềm để giải quyết vấn đề ở giai đoạn biên dịch.
* Giải pháp 2: Sử dụng phần cứng để giải quyết vấn đề một cách linh động.

5. Hazard RAW là gì?

* Xảy ra khi một lệnh cố gắng đọc dữ liệu từ một vị trí nhớ mà lệnh trước đó vừa ghi vào.
* Điều này có nghĩa là dữ liệu mà lệnh hiện tại cần chưa được tính toán hoàn thành và được ghi vào bởi lệnh trước đó, dẫn đến việc lệnh hiện tại phải đợi cho đến khi dữ liệu được ghi hoàn thành trước khi có thể đọc nó.

6. Hazard WAW là gì?

* Xảy ra khi hai lệnh cố gắng ghi dữ liệu vào một vị trí nhớ, và lệnh sau ghi trước khi lệnh trước đã hoàn thành ghi của nó.
* Điều này có thể dẫn đến việc dữ liệu được ghi đè và mất đi, hoặc dẫn đến kết quả không chính xác khi lệnh sau ghi trước khi dữ liệu của lệnh trước được sử dụng.

7. Hazard WAR là gì?

* Xảy ra khi một lệnh cố gắng ghi dữ liệu vào một vị trí nhớ mà một lệnh khác đang cố gắng đọc từ nó.
* Điều này có thể dẫn đến việc dữ liệu mà lệnh sau ghi bị ảnh hưởng bởi dữ liệu đọc từ vị trí đó bởi lệnh trước, dẫn đến kết quả không chính xác.

Chapter 5

1. Mâu thuẫn trong thiết kế bộ nhớ là gì?

Mâu thuẫn trong thiết kế bộ nhớ là một khái niệm trong lĩnh vực kiến trúc máy tính và hệ thống phân phối, đề cập đến các tình huống mà các quy tắc về thứ tự thực thi lệnh và truy cập bộ nhớ không nhất quán với nhau. Mâu thuẫn này có thể dẫn đến hiện tượng không đồng nhất về dữ liệu giữa các thành phần trong hệ thống, làm giảm tính tin cậy và hiệu suất của hệ thống.

2. Mô tả khái niệm phân tầng bộ nhớ và giải thích tầm quan trọng của nó.

Phân tầng bộ nhớ (memory hierachy) là một khái niệm trong kiến trúc máy tính, mô tả cách mà dữ liệu được tổ chức và lưu trữ ở nhiều cấp độ khác nhau của bộ nhớ trong một hệ thống máy tính. Mỗi cấp độ của phân tầng bộ nhớ thường có đặc điểm về tốc độ, dung lượng và giá thành khác nhau, với mục tiêu cung cấp hiệu suất và hiệu quả cao nhất cho các ứng dụng và tác vụ cụ thể.

Phân tầng bộ nhớ thường bao gồm các cấp sau đây:

* Bộ nhớ đăng ký (Register memory)
* Bộ nhớ cache (Cache memory)
* Bộ nhớ chính (Main memory)
* Ổ cứng (Hard disk drive – HDD)

Phân tầng bộ nhớ quan trọng vì nó cung cấp một cách tổ chức hiệu quả cho việc quản lý và truy cập dữ liệu trong hệ thống máy tính. Bằng cách tận dụng sự kết hợp của các cấp độ bộ nhớ với các tính chất khác nhau về tốc độ và dung lượng, phân tầng bộ nhớ giúp cải thiện hiệu suất và tối ưu hóa thời gian truy cập dữ liệu, giảm thiểu thời gian chờ đợi và tăng cường khả năng xử lý của hệ thống.

3. Nêu cơ chế hoạt động các tầng bộ nhớ

* CPU truy cập bộ nhớ có địa chỉ A. Nếu A ở trong bộ đệm, chuyển trực tiếp đến CPU.
* Nếu không, chuyển một khối từ nhớ có chứa A từ bộ nhớ vào bộ đệm. Truy cập A trong bộ đệm.
* Nếu A không có trong bộ nhớ, chuyển một trang bộ nhớ chứa A từ đĩa cứng vào bộ nhớ, sau đó chuyển một khối từ nhớ có chứa A từ bộ nhớ vào bộ đệm. Truy cập A trong bộ đệm.

4. Nguyên lý định hướng thời gian trong thiết kế bộ nhớ là gì?

Định hướng thời gian: Nếu một phần tử được truy cập, nó thường sẽ được truy cập lại ngay sau đó.

5. Nguyên lý định hướng không gian trong thiết kế bộ nhớ là gì?

Nếu một phần tử được truy cập, phần tử có địa chỉ gần đó thường sẽ được truy cập ngay sau đó.

6. Nêu 4 câu hỏi với phân tầng bộ nhớ

* Q1: Khối nhớ có thể được đặt ở đâu tại tầng trên?
* Q2: Làm cách nào để tìm khối nhớ tại tầng trên?
* Q3: Khối nhớ nào nên bị thay thế khi có lỡ?
* Q4: Phương thức ghi bộ nhớ là gì?

7. Giải thích hai cơ chế ghi bộ nhớ.

1. Ghi trực tiếp (write-through):

* Mỗi khi dữ liệu được ghi vào bộ nhớ cache, nó cũng được ghi vào bộ nhớ chính ngay lập tức. Điều này đảm bảo dữ liệu trong cache và dữ liệu trong bộ nhớ chính luôn được đồng bộ hóa.
* Khi CPU cần truy cập dữ liệu, nếu dữ liệu không tồn tại trong cache (cache miss), dữ liệu sẽ được đọc từ bộ nhớ chính. Nếu CPU thực hiện một lệnh ghi, dữ liệu sẽ được ghi cả vào cache và bộ nhớ chính.
* Ưu điểm: Đảm bảo tính nhất quán của dữ liệu giữa cache và bộ nhớ chính, giảm thiểu rủi ro mất mát dữ liệu.
* Nhược điểm: Tăng thời gian truy cập vì khi ghi phải cập nhật cả cache và bộ nhớ chính, làm giảm hiệu suất.

1. Ghi lưu trữ tạm thời (write back):

* Dữ liệu chỉ được ghi vào cache khi nó được thay đổi bởi CPU. Dữ liệu không được sao chép ngay lập tức vào bộ nhớ chính, mà được giữ lại trong cache và được ghi vào bộ nhớ chính chỉ khi cần thiết.
* Khi CPU cần truy cập dữ liệu, nếu dữ liệu không tồn tại trong cache, nó sẽ được đọc từ bộ nhớ chính. Khi CPU thực hiện một lệnh ghi, dữ liệu chỉ được ghi vào cache và được đánh dấu là thay đổi. Dữ liệu không được sao chép ngay vào bộ nhớ chính.
* Khi dữ liệu bị loại bỏ khỏi cache, nếu dữ liệu đã được thay đổi, nó sẽ được ghi vào bộ nhớ chính. Điều này đảm bảo chỉ những dữ liệu đã thay đổi mới được ghi lại vào bộ nhớ chính.
* Ưu điểm: Giảm thời gian truy cập bởi vì không cần cập nhật bộ nhớ chính ngay lập tức sau mỗi lệnh ghi.
* Nhược điểm: Có thể dẫn đến sự không nhất quán giữa dữ liệu trong cache và bộ nhớ chính trong thời gian ngắn khi dữ liệu thay đổi được giữ lại trong cache nhưng chưa được ghi vào bộ nhớ chính.

8. Tại sao cần thay thế khối? Thuật toán thay thế khối là gì? Tại sao các thuật toán thay thế khối không thể áp dụng được cho bộ đệm liên kết trực tiếp?

* Khi bộ nhớ cache đã đầy, việc lựa chọn khối nào sẽ bị loại bỏ ra khỏi cache để nhận khối dữ liệu mới là rất quan trọng. Mục đích của thay thế khối là để tối ưu hóa hiệu suất của cache bằng cách loại bỏ những khối dữ liệu ít hoặc không được sử dụng, giữ lại những khối dữ liệu quan trọng để tăng hiệu suất truy cập.
* Thuật toán thay thế khối là một thuật toán được sử dụng trong quản lý bộ nhớ cache để quyết định khối dữ liệu nào sẽ bị loại bỏ khi cache đã đầy và cần chứa thêm khối mới
* Các thuật toán thay thế không thể áp dụng cho bộ đệm liên kết trực tiếp vì trong bộ đệm này, mỗi khối dữ liệu được lưu trữ trong một vị trí cụ thể được xác định bởi hàm băm. Khi cần thay thế một khối, không có sự lựa chọn về việc khối nào sẽ bị loại bỏ, mà khối mới sẽ được đặt vào vị trí cụ thể được xác định bởi hàm băm.

9. Mô tả kiến trúc UMA.

* Hệ thống đa xử lý đơn giản nhất có một bus duy nhất để kết nối ít nhất hai CPU và một bộ nhớ (dùng chung cho tất cả các bộ xử lý)
* Khi CPU muốn truy cập một vị trí bộ nhớ, nó sẽ kiểm tra xem bus đang rảnh rỗi, sau đó sẽ gửi yêu cần đến modul giao diện bộ nhớ và đợi dữ liệu được yêu cầu có sẵn trên bus.
* Bộ xử lý đa lõi là hệ thống đa xử lý UMA nhỏ, nơi bộ đệm được chia sẻ đầu tiên (L2 hoặc L3) thực sự là kênh thông tin liên lạc.
* Bộ nhớ dùng chung có thể nhanh chóng trở thành nút thắt cổ chai cho hiệu năng hệ thống, vì tất cả các bộ xử lý phải đồng bộ hóa truy cập trên một bus và bộ nhớ.
* Các hệ thống đa xử lý lớn hơn (>32 CPU) không thể sử dụng một bus duy nhất kết nối các CPU tới các bộ nhớ, bởi vì tranh chấp bus trở nên không thể điều khiển được.
* Kết nối CPU-bộ nhớ được thực hiện thông qua một mạng kết nối.
* Bộ nhớ cache cục bộ cho mỗi CPU làm giảm bớt vấn đề, hơn nữa mỗi bộ xử lý có thể được trang bị một bộ nhớ riêng để lưu trữ dữ liệu của các tính toán không cần được chia sẻ bởi các bộ xử lý khác. Luồng dữ liệu đến từ bộ nhớ dùng chung có thể giảm đáng kể.

10. Mô tả kiến trúc NC-NUMA.

* Trong hệ thống NC-NUMA, bộ xử lý không có bộ nhớ cache cục bộ.
* Mỗi quyền truy cập bộ nhớ được quản lý bằng một MMU đã sửa đổi, kiểm soát nếu yêu cầu dành cho cục bộ hoặc cho một khối từ xa; bên trong trường hợp thứ hai, yêu cầu được chuyển tiếp đến nút chứa dữ liệu yêu cầu.
* Rõ ràng, các chương trình sử dụng dữ liệu từ xa (đối với CPU yêu cầu chúng) sẽ chạy chậm hơn nhiều so với những gì chúng sẽ làm nếu dữ liệu được lưu trữ trong bộ nhớ cục bộ.
* Trong hệ thống NC-NUMA không có vấn đề về sự chặt chẽ trong bộ nhớ cache, bởi vì không có bộ nhớ đệm nào cả: mỗi mục bộ nhớ nằm trong một địa điểm.
* Tuy nhiên, truy cập bộ nhớ từ xa rất kém hiệu quả. Vì lý do này, hệ thống NC-NUMA có thể sử dụng phần mềm đặc biệt để di chuyển các trang bộ nhớ từ khối này sang khối khác, chỉ để tối đa hóa hiệu năng.
* Một bộ quét trang sẽ kích hoạt sau vài giây, kiểm tra thống kê về việc sử dụng bộ nhớ và di chuyển các trang từ một khối sang khối khác, để tăng hiệu năng.
* Trên thực tế, mỗi bộ xử lý cũng có thể có bộ nhớ riêng và bộ nhớ cache, và chỉ dữ liệu riêng tư (dữ liệu được cấp phát trong bộ nhớ cục bộ riêng) có thể nằm trong bộ nhớ cache.
* Giải pháp này làm tăng hiệu suất của từng bộ xử lý và được sử dụng trong Cray T3D/E.
* Tuy nhiên, thời gian truy cập từ xa vẫn rất cao, bộ xử lý cần 400 chu kỳ xung nhịp trong Cray T3D/E, so với 2 để truy xuất dữ liệu từ cục bộ trong bộ nhớ đệm.

11. Mô tả kiến trúc CC-NUMA.

* Bộ nhớ đệm có thể giảm bớt sự cố do truy cập dữ liệu từ xa, nhưng đưa trở lại vấn đề sự chặt chẽ dữ liệu trong bộ nhớ cache.
* Phương pháp phổ biến trong hệ thống CC-NUMA với nhiều CPU để thực thi sự chặt chẽ trong bộ nhớ cache là giao thức dựa trên thư mục.
* Ý tưởng cơ bản là liên kết mỗi nút trong hệ thống với một thư mục cho các khối RAM của nó: một cơ sở dữ liệu cho biết một khối nằm trong bộ đệm nào, và trạng thái của nó là gì.
* Khi một khối bộ nhớ được định địa chỉ, thư mục trong nút nơi đặt khối được truy vấn, để biết liệu khối nằm trong bất kỳ bộ nhớ cache và nếu có, nếu nó đã được thay đổi so với bản sao trong RAM.
* Vì một thư mục được truy vấn tại mỗi lần truy cập bởi một chỉ dẫn tới khối bộ nhớ tương ứng, nó phải được triển khai với phần cứng rất nhanh, như một phiên bản có bộ nhớ đệm kết hợp, hoặc ít nhất là với RAM tĩnh.

Chapter 6

1. Trình bày hệ thống lưu trữ tại trung tâm dữ liệu Google