

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



TÍNH TOÁN SONG SONG

(Dùng cho sinh viên hệ đào tạo đại học từ xa)

Lưu hành nội bộ

cuu duong than cong . com

HÀ NỘI - 2007

TÍNH TOÁN SONG SONG

Biên soạn : THS. PHẠM VĂN CƯỜNG

cuu duong than cong . com

Bài giảng

TÍNH TOÁN SONG SONG

Biên soạn: Phạm Văn Cường

Khoa CNTT1- Học viện Công nghệ BCVT

Email: pcuongcntt@yahoo.com

cuu duong than cong . com

cuu duong than cong . com

Mục lục

| | |
|--|-----------|
| CHƯƠNG 1 : CÁC KIẾN TRÚC SONG SONG..... | 5 |
| 1.1 Tổng quan về tính toán song song | 5 |
| 1.1.1 Nhu cầu tính toán..... | 5 |
| 1.1.2 Lịch sử phát triển..... | 7 |
| 1.1.3 Các thuật ngữ..... | 9 |
| 1.1.4 Các xu thế xây dựng máy tính | 9 |
| 1.2 Các kiến trúc song song | 10 |
| 1.2.1 Máy tính một dòng lệnh, một dòng dữ liệu (SISD) | 11 |
| 1.2.2 Bộ nhớ chia sẻ (shared memory) và bộ nhớ phân tán (distributed memory). | 13 |
| 1.2.3 Máy tính một dòng lệnh, nhiều dòng dữ liệu (SIMD) | 14 |
| 1.2.4 Máy tính nhiều dòng lệnh, một dòng dữ liệu (MISD) | 17 |
| 1.2.5 Máy tính nhiều dòng lệnh, nhiều dòng dữ liệu (MIMD) | 19 |
| 1.2.6 Hiệu suất của Máy tính song song | 20 |
| 1.3 Tổ chức các bộ vi xử lý | 21 |
| 1.3.1 Mạng hình lưới (Mesh)..... | 21 |
| 1.3.2 Mạng hình cây nhị phân (Binary Tree Networks) | 22 |
| 1.3.3 Mạng hình siêu cây (Hypertree networks)..... | 22 |
| 1.3.4 Mạng hình tháp (Pyramid networks) | 23 |
| 1.3.5 Mạng hình bướm (Butterfly networks)..... | 24 |
| 1.3.6 Mạng hình siêu khối (Hypercube networks)..... | 25 |
| 1.3.7 Mạng các chu trình hướng kết nối khối (Cube-Connected Cycles networks) | 26 |
| 1.3.8 Mạng hoán vị di chuyển (Shuffle-exchange networks) | 27 |
| 1.3.9 Mạng de Bruijn..... | 29 |
| 1.3.10 Tổng kết về tổ chức các bộ vi xử lý | 29 |
| 1.4 Các hệ thống mảng bộ xử lý, đa bộ xử lý, và đa máy tính..... | 30 |
| 1.4.1 Hệ thống mảng bộ vi xử lý (processor arrays)..... | 30 |
| 1.4.2 Máy tính đa bộ xử lý (Multiprocessors) | 35 |
| 1.4.3 Hệ thống đa máy tính (Multicomputers) | 39 |
| 1.5 Kết chương..... | 41 |
| 1.6 Câu hỏi và bài tập | 42 |
| 1.6.1 Câu hỏi..... | 42 |
| 1.6.2 Bài tập..... | 44 |
| CHƯƠNG 2 : CÁC THUẬT TOÁN SONG SONG | 45 |
| 2.1 Mô hình PRAM | 45 |
| 2.1.1 Mô hình xử lý tuần tự | 46 |
| 2.1.2 Mô hình tính toán song song PRAM | 46 |
| 2.1.3 Một số thuật toán PRAM | 48 |
| 2.2 Các thuật toán song song nhân hai ma trận | 56 |
| 2.2.1 Thuật toán nhân ma trận tuần tự | 57 |
| 2.2.2 Thuật toán nhân ma trận trên máy SIMD với các bộ xử lý được tổ chức theo mạng hình lưới hai chiều (2-D Mesh SIMD). | 57 |
| 2.2.3 Thuật toán nhân ma trận trên máy SIMD với các bộ xử lý được tổ chức theo mạng hình siêu khối (Hypercube SIMD). | 61 |
| 2.2.4 Thuật toán nhân ma trận trên máy đa bộ xử lý | 64 |
| 2.3 Các thuật toán sắp xếp song song..... | 67 |

| | |
|---|------------|
| 2.3.1 Sắp xếp bằng liệt kê (enumeration sort) và cận dưới (lower bounds) của sắp xếp song song..... | 67 |
| 2.3.2 Sắp xếp song song đổi chỗ chẵn lẻ (odd-even transposition) | 69 |
| 2.3.3 Sắp xếp song song trộn bitonic (bitonic merge) | 71 |
| 2.3.4 Sắp xếp song song tựa trên Quicksort | 83 |
| 2.4 Thuật toán tìm kiếm song song trên danh bạ..... | 88 |
| 2.4.1 Độ phức tạp của tìm kiếm song song. | 88 |
| 2.4.2 Tìm kiếm song song trên máy tính đa bộ xử lý..... | 89 |
| 2.5 Thuật toán song song trên đồ thị..... | 97 |
| 2.5.1 Thuật toán song song tìm đường đi ngắn nhất | 97 |
| 2.5.2 Thuật toán song song tìm cây khung bé nhất | 102 |
| 2.7 Kết chương | 107 |
| 2.8 Câu hỏi và bài tập | 108 |
| 2.8.1 Câu hỏi..... | 108 |
| 2.8.2 Bài tập..... | 109 |

cuu duong than cong . com

cuu duong than cong . com

Lời nói đầu (chưa viết)

Chương 2: Các vấn đề của hệ thống tính toán song song

LT8/BT2

- 2.1 Hiệu suất của hệ thống xử lý song song.
- 2.2 Tốc độ (speedup) và hiệu quả (efficiency) của xử lý song song
 - 2.2.1 Tốc độ (speedup) của xử lý song song
 - 2.2.2 Hiệu quả (efficiency) của xử lý song song
 - 2.2.3 Định luật Amdhal và Gustafson-Barsis về tốc độ và hiệu quả của xử lý song song.
- 2.3 Ánh xạ dữ liệu trên máy tính song song
 - 2.3.1 Ánh xạ dữ liệu lên các mảng bộ vi xử lý (processor arrays).
 - 2.3.2 Ánh xạ dữ liệu lên hệ thống nhiều máy tính (multicomputers).
- 2.4 Vấn đề cân bằng tải động trên hệ thống nhiều máy tính (multicomputers)
- 2.5 Vấn đề lập lịch biểu trên hệ thống nhiều máy tính (multicomputers)
 - 2.5.1 Giải thuật Graham 's List Scheduling
 - 2.5.2 Giải thuật Coffman-Graham Scheduling
 - 2.5.3 Các mô hình đơn định và không đơn định.
- 2.6 Vấn đề deadlocks

Chương 3: Lập trình song song LT9/TH4/KT1

- 3.1 Cơ bản về giao tiếp bằng phương pháp trao đổi thông điệp (message passing)
 - 3.1.1 Trao đổi thông điệp như một mô hình lập trình
 - 3.1.2 Cơ chế trao đổi thông điệp
 - 3.1.3 Tiếp cận đến một ngôn ngữ cho lập trình song song
- 3.2 Thư viện giao diện trao đổi thông điệp (Message Passing Interface – MPI)
 - 3.2.1 Giới thiệu về MPI
 - 3.2.2 Lập trình song song bằng ngôn ngữ C và thư viện MPI
 - 3.2.3 Một số kỹ thuật truyền thông: broadcast, scatter, gather, blocking message passing...
- 3.3 Máy ảo song song (Parallel Virtual Machine-PVM)
- 4.4 Thiết kế và xây dựng một chương trình (giải một bài toán (NP-complete) sử dụng MPI và C.

Thực hành: Xây dựng và chạy chương trình sử dụng C và MPI

CHƯƠNG 1 : CÁC KIẾN TRÚC SONG SONG

Nội dung chương này trình bày các vấn đề sau:

- Tổng quan về tính toán song song: phần này trình bày về nhu cầu tính toán trên mọi lĩnh vực: thương mại, khoa học...; lịch sử phát triển của máy tính song song, các xu thế thiết kế máy tính.
- Các kiến trúc song song: phần này trình bày về 4 loại kiến trúc máy tính được phân loại theo thuật ngữ Flynn; máy tính một dòng lệnh, một dòng dữ liệu (SISD), máy tính một dòng lệnh, nhiều dòng dữ liệu (SIMD), máy tính nhiều dòng lệnh, một dòng dữ liệu (MISD) và máy tính nhiều dòng lệnh, nhiều dòng dữ liệu (MIMD).
- Tổ chức các bộ vi xử lý trong các máy tính song song theo 9 cách khác nhau, bao gồm: tổ chức các bộ vi xử lý theo hình mạng lưới, theo hình cây, theo hình siêu cây, hình tháp, hình siêu khối, các chu trình hướng khối, hoán vị-đổi chỗ, và de Bruijn. Mỗi cách thức tổ chức được đánh giá ưu điểm, nhược điểm qua các tiêu chí: đường kính, độ rộng phân đôi và số nút/cạnh.
- Một số máy tính song song thực tế: máy tính mang các bộ vi xử lý, máy tính đa bộ xử lý và hệ thống đa máy tính.
- Phần cuối cùng là câu hỏi và bài tập dành cho sinh viên.

1.1 Tổng quan về tính toán song song

1.1.1 Nhu cầu tính toán

Khoa học kinh điển dựa vào các quan sát, phát triển thành các lý thuyết và tiến hành thực nghiệm. Sự quan sát một hiện tượng một hiện tượng dẫn đến một giả thuyết nào đó. Nhà khoa học sẽ phát triển một lý thuyết để giải thích hiện tượng đó và thiết kế các thực nghiệm để chứng minh (hoặc bác bỏ) lý thuyết đó. Các kết quả từ thực nghiệm sẽ giúp các nhà khoa học hoàn chỉnh lý thuyết của mình.

Một điều không may là không phải lúc nào ta cũng có thể sử dụng các thực nghiệm để kiểm chứng lý thuyết, đó là vì việc tiến hành các thực nghiệm đó tốn quá nhiều thời gian và tiền của. Các máy tính tốc độ cao sẽ cho phép các nhà khoa học kiểm chứng các giả thuyết của họ theo phương pháp mô phỏng các hiện tượng (numerical simulation). Nhà khoa học so sánh các kết quả của chương trình mô phỏng lý thuyết và quan sát các hiện tượng trong “thế giới thực” bằng chương trình mô phỏng. Các nhà khoa học sẽ hiệu chỉnh lý thuyết hoặc tiếp tục quan sát nếu có sự khác biệt.

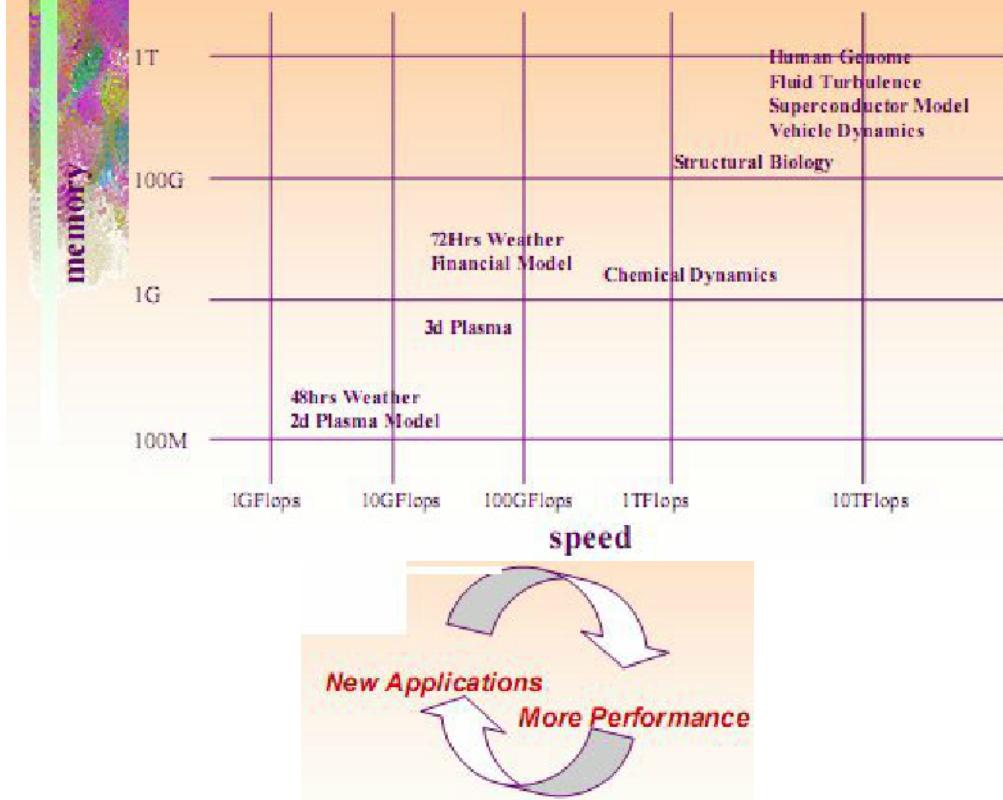
Chính vì vậy, khoa học hiện đại được mô tả bằng sự quan sát, lý thuyết, thực nghiệm và mô phỏng. Trong đó, mô phỏng ngày càng đóng vai trò quan trọng đối với các nhà khoa học. Nhiều bài toán khoa học phức tạp khi được mô phỏng yêu cầu độ phức tạp là hàm số mũ. A.

a. *Nhu cầu tính toán cho các ứng dụng khoa học.*

Các ứng dụng mới ngày càng có nhu cầu tiêu tốn tài nguyên phần cứng hơn.

cuu duong than cong . com

Grand Challenges



Hình 1.1. Các ứng dụng mới ngày càng yêu cầu số lượng tính toán lớn

Điển hình là các loại ứng dụng :

- Mô phỏng, mô hình hóa như các bài toán sau đây; những bài toán này vẫn còn là thách thức lớn (grand challenges) đối với khoa học, đó là:

1. Hóa học lượng tử (quantum), phương pháp thống kê, và vật lý tương đối.
2. Vũ trụ (cosmology) và vật lý thiên văn (astrophysics).
3. Động lực học (fluid dynamics)
4. Thiết kế vật liệu và siêu dẫn
5. Sinh học, dược học, so sánh DNA, công nghệ gen & protein...
6. Y học và mô hình hóa sự hoạt động hệ xương và các cơ quan nội tạng
7. Mô hình hóa khí hậu và sự biến đổi môi trường

Hình 1.2. Yêu cầu về phân cứng của các bài toán khoa học

Một ví dụ thực tế về mô hình hóa bài toán lưu lượng của dòng chảy đại dương [được thực hiện bởi các nhà khoa học từ đại học bang Oregon, Hoa Kỳ]. Để đạt được kết quả chính xác, các nhà khoa học chia đại dương thành 4096 vùng từ đông sang tây và 1024 vùng từ bắc đến nam. Đồng thời, đại dương được chia thành 12 lớp; với mô hình này thì đại dương có khoảng 50 triệu

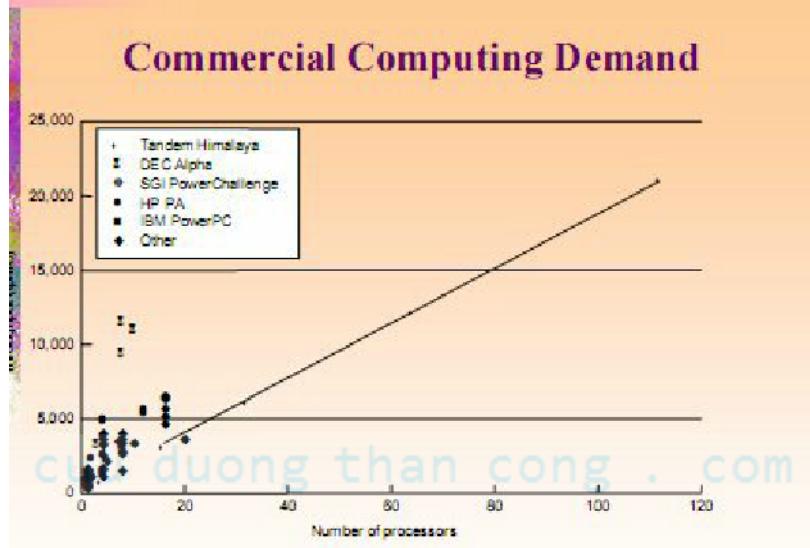
khối 3 chiều. Để mô phỏng cho một khối 3 chiều như vậy với một chu kỳ là 10 phút thì cần khoảng 30 tỉ phép tính. Trong khi đó các nhà nghiên cứu mô phỏng chu kỳ của đại dương trong vòng hàng trăm năm.

b. Nhu cầu tính toán cho các ứng dụng và dịch vụ thương mại

Các ứng dụng và dịch vụ thương mại ngày càng đa dạng, điển hình là:

- Các ứng dụng đa phương tiện như: video servers, multimedia databases, video on demand...
- Các ứng dụng về data mining và phân tích, xử lý dữ liệu trực tuyến (OLAP)
- Các ứng dụng máy chủ thời gian thực, xử lý đồ họa...

Đó là các dịch vụ yêu cầu khối lượng tính toán lớn:

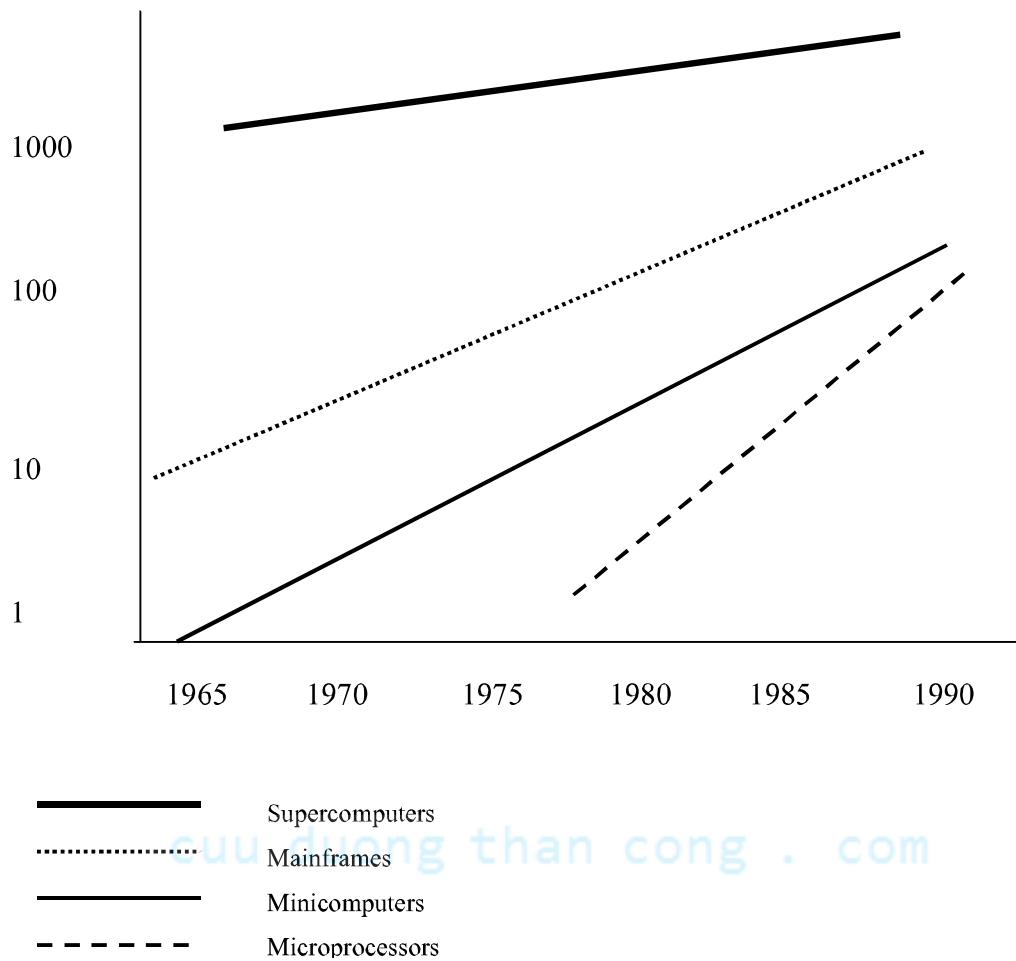


Hình 1.3. Nhu cầu tính toán cho các ứng dụng thương mại

1.1.2 Lịch sử phát triển

Phải mất hơn 20 năm để các máy tính song song đi từ các phòng thí nghiệm ra thị trường. Daniel Slotnick tại đại học Illinois đã thiết kế hai máy tính song song sớm nhất là: Solomon được xây dựng bởi công ty Điện tử Westinghouse vào những năm 1960 và ILLIAC IV được lắp ráp bởi công ty Burroughs vào những năm 1970. Sau đó, trong suốt thập kỷ 70s, trường Đại học Carnegie Mellon xây dựng hai máy tính song song C.mmp và CM*. Vào năm 1980, các nhà khoa học tại Học viện kỹ thuật CalTech xây dựng máy tính Cosmic Cube, tiền thân của đa máy tính ngày nay và được hiện thực bởi các công ty Ametek, Intel và nCUBE.

Cho đến giữa thập kỷ 80, các máy tính song song với nhiều bộ vi xử lý mới được đưa ra thị trường. Một nghiên cứu về hiệu suất máy tính đối với các loại máy tính khác nhau đã chỉ ra lý do các máy tính song song dựa trên đa bộ xử lý trở thành hiện thực.



Hình 1.4 : Hiệu suất của 4 loại máy tính song song thông dụng

Trong hình 1.4, tỉ lệ gia tăng hiệu suất đối với các loại máy tính minicomputers, mainframes và supercomputers hàng năm chỉ dưới 20%. Trong khi đó, tỉ lệ gia tăng hiệu suất đối với các bộ xử lý (microprocessors) trung bình khoảng 35% mỗi năm.

Tại sao hiệu suất của máy tính nhiều bộ vi xử lý tăng nhanh hơn các loại máy tính song song khác?. Hiệu suất của một bộ vi xử lý đơn có thể được cải tiến thông qua sự cải tiến kiến trúc hoặc cải tiến về công nghệ. Sự cải tiến về kiến trúc có thể làm tăng khối lượng công việc trong một chu kỳ lệnh. Sự cải tiến về công nghệ có thể làm giảm thời gian thực hiện chu kỳ lệnh. Những năm 1970s, các kiến trúc cơ bản: bộ nhớ có các bit song song (bit-parallel memory), bộ tính toán bit song song (bit-parallel arithmetic), bộ nhớ Cache, các kênh truyền dữ liệu, bộ nhớ xen kẽ (interleave memory), tiền xử lý lệnh (instruction lookahead), xử lý xen kẽ (pipelining), đa chức năng và các bộ trợ giúp xử lý xen kẽ đồng mã lệnh (pipelined functional unit)... đã được tích hợp vào thiết kế các siêu máy tính. Tuy nhiên, sự cải tiến hiệu suất của các bộ xử lý riêng lẻ (làm

giảm thời gian thực hiện chu kỳ lệnh) rất khó khăn vì điều này bị giới hạn bởi tốc xử lý của vi mạch điện tử (bé hơn tốc độ ánh sáng).

Ngược lại, máy tính nhiều bộ vi xử lý đạt được các bước tiến về cải tiến hiệu suất rất ấn tượng. Mặc dù, ban đầu những máy tính nhiều bộ xử lý không kết hợp tất cả các cải tiến kiến trúc như trong các siêu máy tính và tốc độ đồng hồ của chúng khá chậm.

Sự hội tụ (convergence) giữa microcomputers và siêu máy tính truyền thống đã kéo theo sự thương mại hóa những máy tính song song với hàng trăm bộ xử lý. Điểm điểm là các máy tính song song dựa trên đa bộ vi xử lý có thể kể đến như: Intel's Paragon XP/STM, MP-2TM, và Thinking Machine -5TM đã vượt qua tốc độ của các siêu máy tính đơn bộ xử lý như: Cray Y/MPTM và NEC SX -3TM.

1.1.3 Các thuật ngữ

Hầu hết các máy tính hiệu suất cao hiện đại đều hỗ trợ xử lý nhiều lệnh một cách đồng thời (concurrency). Chẳng hạn, đa xử lý (multiprocessing) là một phương pháp được sử dụng để đạt được xử lý đồng thời nhiều lệnh ở mức công việc hoặc chương trình. Trong khi đó, xử lý xen kẽ dòng lệnh (pipeline) là phương pháp nhằm xử lý đồng thời các lệnh ở mức lệnh (instruction). Tuy nhiên, không thể gọi các máy tính hỗ trợ xử lý các lệnh đồng thời là máy tính song song. Dưới đây là các thuật ngữ cơ bản:

Lập trình song song (parallel programming) là việc lập trình sử dụng một ngôn ngữ có hỗ trợ xử lý song song các lệnh trong một chương trình.

Máy tính song song (Parallel computer) là một máy tính có nhiều bộ xử lý (multiple-processor computer) có khả năng phối hợp với nhau để giải quyết các bài toán.

Xử lý song song (parallel computing) là quá trình sử dụng máy tính song song để giải quyết các bài toán đơn (single problem) nhanh hơn.

Siêu máy tính (supercomputer) là một máy tính đa năng có khả năng giải các bài toán đơn với tốc độ tính toán cao (cỡ hàng nghìn tỉ phép tính trong một giây). So với các máy tính được chế tạo cùng thời thì siêu máy tính có tốc độ xử lý lớn hơn hàng nghìn lần. Các siêu máy tính hiện đại là các máy tính song song. Một vài siêu máy tính có số lượng ít bộ xử lý nhưng rất mạnh; đa số siêu máy tính có số lượng bộ xử lý rất lớn.

Thông lượng (throughput) của một thiết bị là lưu lượng dữ liệu được truyền tải qua thiết bị trong một đơn vị thời gian. Có nhiều cách để cải tiến thông lượng của một thiết bị như: tăng tốc độ, tăng số thao tác tại một thời điểm...

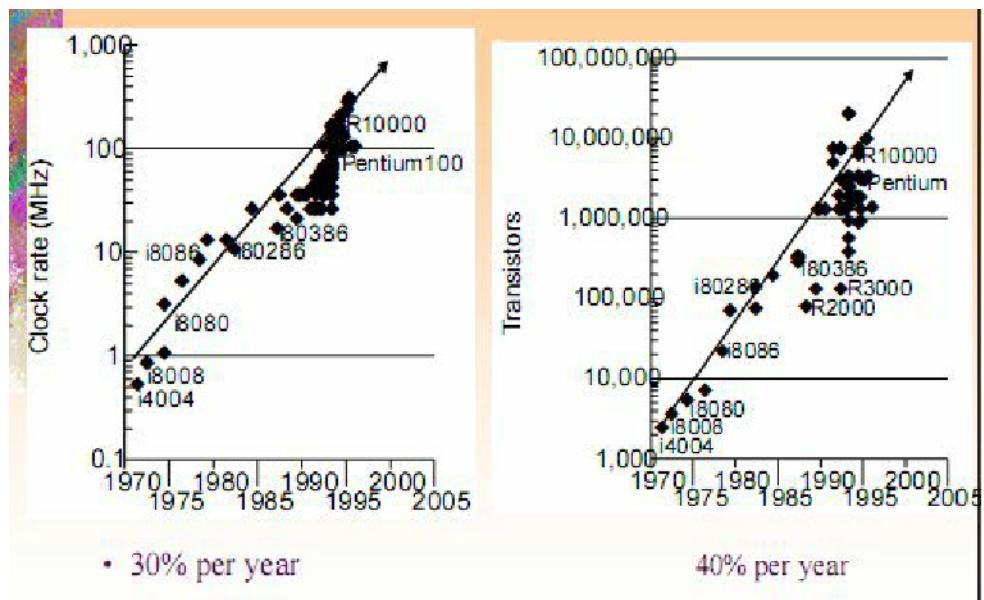
Song song hóa dữ liệu (data parallelism) là việc sử dụng nhiều bộ chức năng (functional units) để xử lý cùng một thao tác đồng thời cho các phần tử của một tập dữ liệu.

Tốc độ (speedup) là tỉ số giữa thời gian cần thiết xử lý một thuật toán tuân tự tốt nhất và thời gian cần thiết để xử lý pipeline hoặc song song trên cùng một máy tính.

1.1.4 Các xu thế xây dựng máy tính

a. Xu thế phát triển phân cứng:

- Hiệu suất bộ vi xử lý tăng 50 -100% mỗi năm.
- Cứ 3 năm, số lượng các transistors tăng gấp đôi trên mỗi vi mạch.
- Cứ 3 năm kích thước bộ nhớ RAM tăng 4 lần.



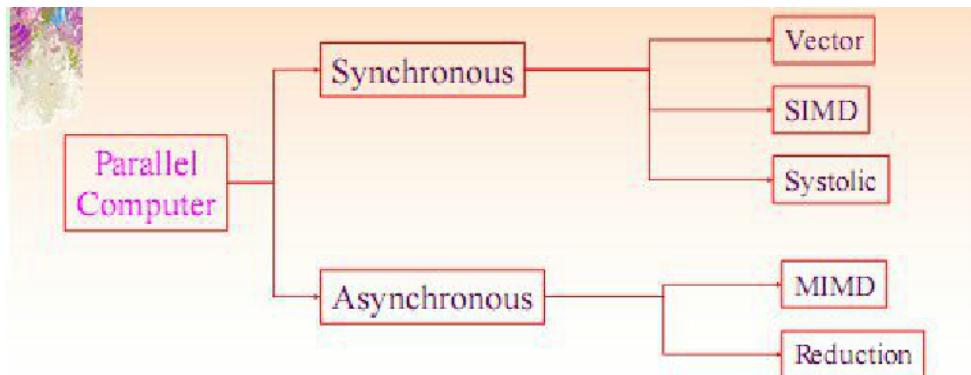
Hình 1.5 Xu thế phát triển phần cứng

b. Xu hướng thiết kế kiến trúc máy tính:

- Giảm thời gian thực hiện chu kỳ lệnh máy; tuy nhiên, xu hướng này bị giới hạn bởi công nghệ điện tử.
- Tăng số lệnh máy được thực hiện đồng thời (xử lý xen kẽ các dòng lệnh, siêu vô hướng ...)
- Tăng số bít truyền dữ liệu song song 4 bít, 8 bít, 16 bít, 32 bít, 64 bít và 128 bít.
- Song song ở mức luồng (multithread)
- Nhiều bộ xử lý được tích hợp trên cùng một chip (dual core, quad. Core ...).

1.2 Các kiến trúc song song

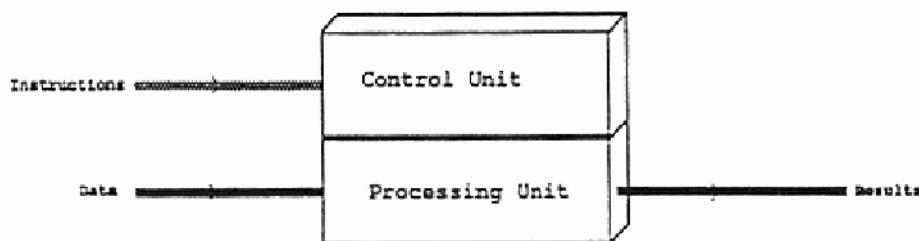
Kiến trúc của các máy tính song song có thể được chia làm hai loại (như hình dưới đây): kiến trúc đồng bộ, bao gồm: máy tính véc tơ, máy tính SIMD hoặc máy tính Systolic và kiến trúc không đồng bộ, bao gồm: máy tính MIMD, Reduction. Trong kiến trúc song song kiểu đồng bộ thì các bộ vi xử lý thực hiện đồng thời cùng một lệnh nào đó trên các bộ vi xử lý khác nhau (với dữ liệu có thể khác nhau) và kết thúc trong cùng một chu kỳ lệnh. Ngược lại, kiến trúc song song kiểu không đồng bộ thì các bộ vi xử lý có thể thực hiện các lệnh khác nhau, hoặc các đoạn chương trình khác nhau và có thể kết thúc việc xử lý lệnh trong các thời điểm khác nhau.



Hình 1.6. Hai loại kiến trúc song song: đồng bộ và không đồng bộ

1.2.1 Máy tính một dòng lệnh, một dòng dữ liệu (SISD)

a. Máy tính SISD

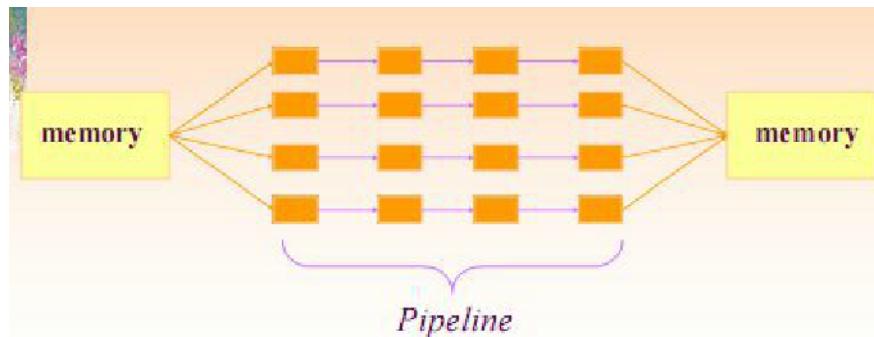


Hình 1.7. Mô hình máy tính SISD cỗ diễn
cuuduongthancong . com

Hình trên là kiến trúc tổng quan của máy tính SISD (Single Instruction stream Single Data stream). Nó bao gồm hai đầu vào: dòng lệnh và dòng dữ liệu; một bộ điều khiển chứa bộ giải mã lệnh và bộ tạo xung điều khiển; và một bộ xử lý lệnh và dữ liệu. Các máy tính SISD là các máy tính xử lý tuần tự. Tại một thời điểm, chỉ một dòng lệnh và một dòng dữ liệu được xử lý. Chính vì vậy mà máy tính SISD còn được gọi là máy tính có von Neumann.

Mặc dù dòng lệnh (instruction stream) được xử lý một cách tuần tự, nhưng các lệnh có thể được xử lý đồng thời theo cách xen kẽ dòng mã lệnh (pipeline). Nhưng tại một thời điểm, chỉ có duy nhất một lệnh được giải mã (decode) hoặc được đọc (fetch). Máy tính SISD có thể có nhiều bộ chức năng (multiple functional units) như: bộ đồng xử lý toán học (mathematics co-processor), bộ vector (vector units), các bộ xử lý đồ họa (graphics processors), và các bộ xử lý vào/ra (I/O processors). Trong mô hình kiến trúc máy tính SISD, tất cả các bộ chức năng đều được điều khiển bởi một bộ xử lý (single processing unit) đơn.

Một số biến thể của máy tính SISD như Systolic có thể tăng năng lực xử lý nhờ vào tích hợp nhiều bộ xử lý lệnh và dữ liệu như hình dưới đây:



Hình 1.8. Mô hình máy tính SISD với mảng bộ xử lý

Một số máy tính SISD hiện đại như Cray-1 còn hỗ trợ nhiều bộ xử lý kiểu vector (như hình trên). Dòng dữ liệu được “lọc” qua một mảng các bộ xử lý (processing units) để xử lý. Do đó, người ta còn gọi các máy tính SISD với mảng các bộ xử lý là máy tính Systolic.

b. Ví dụ về thực hiện bài toán Banking trên máy tính song song Systolic:

Số bộ vi xử lý = số máy ATM

Số công việc = số khách hàng

Có m khách hàng, thời gian phục vụ mỗi khách hàng là t_i .

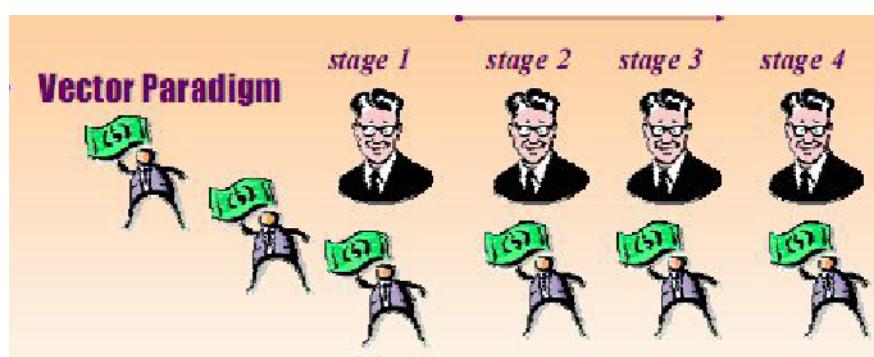
Có n máy ATM,

Với $n=1$: xử lý tuần tự; thời gian thực hiện sẽ là tổng thời gian phục vụ từng khách hàng:

$$T = t_1 + t_2 + \dots + t_n$$

Với $m=n$, trường hợp tốt nhất; thời gian thực hiện sẽ là $T=\max\{t_1, t_2, \dots, t_n\}$

Với $n < m$, mục tiêu là xử lý công việc sao cho tại mỗi thời điểm, số máy ATM không làm việc là ít nhất. Trong trường hợp này máy tính Systolic giải bài toán trên như sau:



Hình 1.9. Giải bài toán Banking trên máy Systolic

Bước 1: đọc số tài khoản

Bước 2: kiểm tra tính hợp lệ và số dư trong tài khoản

Bước 3: cập nhật tài khoản với giao dịch mới

Bước 4: nhận tiền từ khách hàng (deposit) hoặc trả tiền rút (withdraw) cho khách hàng.

Thời gian thực hiện:

$$(1/n) * (t_1 + t_2 + \dots + t_n) \sim (t * m)/n.$$

c. Một số máy tính SISD thực tế :

- CDC 6600 : không hỗ trợ xử lý pipeline, nhưng được tích hợp các bộ đa chức năng.
- CDC 7600: là thế hệ sau của CDC 6600, có bộ xử lý toán học và logic (ALU) có hỗ trợ xử lý pipeline.
- Amdahl 470/6: hỗ trợ xử lý pipeline.
- Cray-1: hỗ trợ vector các bộ xử lý.

1.2.2 Bộ nhớ chia sẻ (shared memory) và bộ nhớ phân tán (distributed memory).

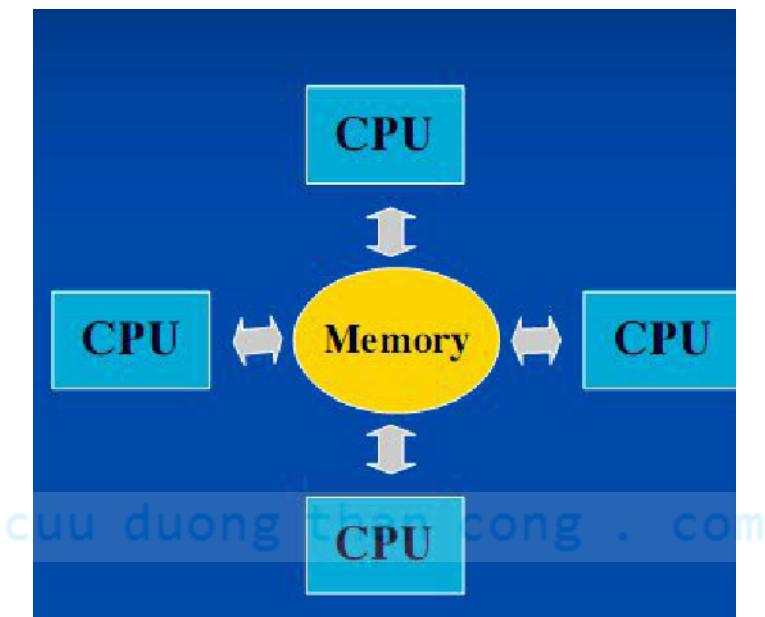
a. Khái niệm về chia sẻ phương tiện (shared medium).

- Cho phép nhận một thông điệp tại một thời điểm.
- Các thông điệp có thể truyền rộng rãi (broadcast).
- Mỗi bộ xử lý “lắng nghe” (listens) các thông điệp chuyên đến.
- Cơ chế trọng tài là tập trung.
- Yêu cầu truyền lại thông điệp nếu có đụng độ.
- Ethernet, Bus là các ví dụ điển hình về chia sẻ phương tiện.

Các kiến trúc song song sẽ hỗ trợ truyền thông điệp kiểu điểm-điểm giữa các cặp bộ vi xử lý, các bộ xử lý được kết nối với nhau theo một hình trạng (topology) nào đó (star, ring..)

Hai ưu điểm của chia sẻ phương tiện là có thể truyền đồng thời nhiều thông điệp (broadcast) tại một thời điểm và dễ dàng mở rộng (khi bổ sung thêm CPU).

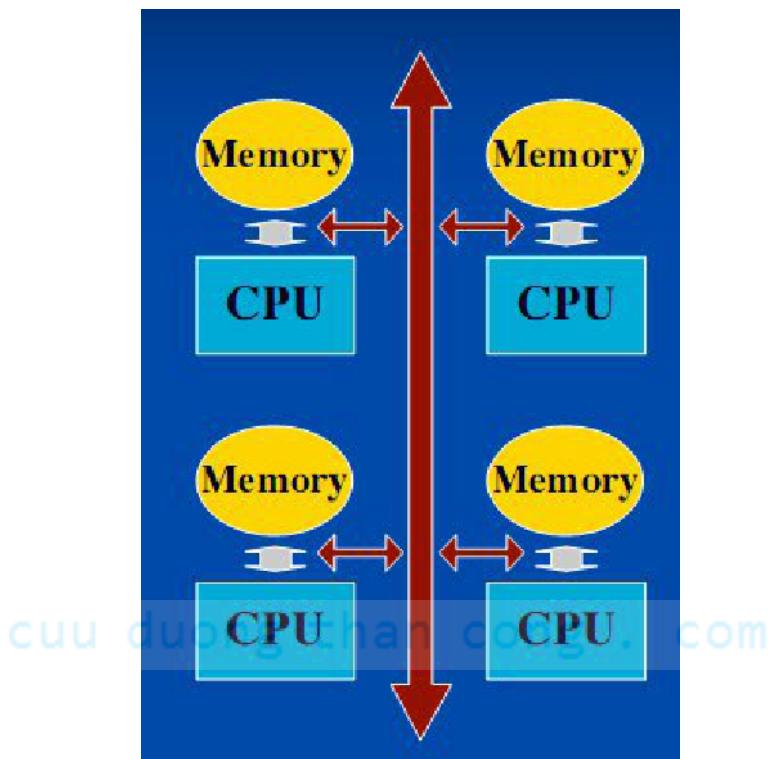
b. Bộ nhớ chia sẻ :



Hình 1.10. Bộ nhớ chia sẻ

- Mỗi bộ xử lý có thể thao tác trên nhiều dữ liệu khác nhau (thậm chí tất cả) trên bộ nhớ chia sẻ.
- Kết nối các bộ xử lý (qua bus, mạng...)
- Có bộ nhớ chia sẻ toàn cục
- Giao tiếp với bộ nhớ toàn cục thông qua READ/WRITE

c. Bộ nhớ phân tán :



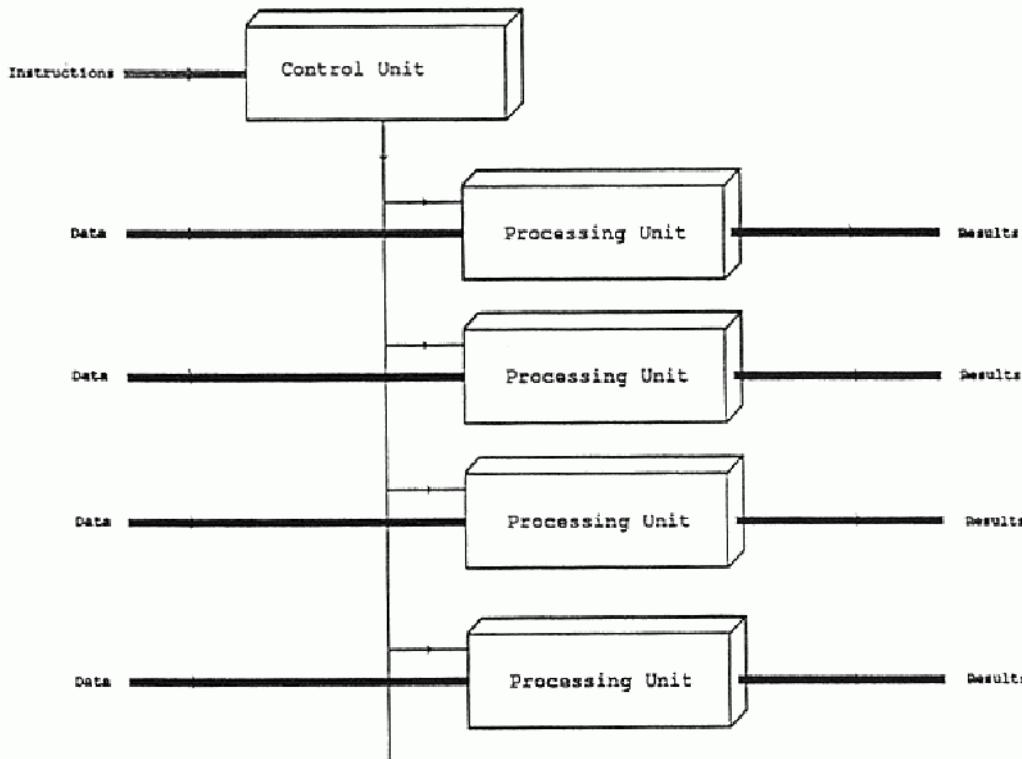
Hình 1.11. Bộ nhớ phân tán

- Kết nối giữa các cặp CPU + bộ nhớ cục bộ
- Giao tiếp qua thông điệp (message) được truyền qua mạng.
- Dễ dàng mở rộng.

1.2.3 Máy tính một dòng lệnh, nhiều dòng dữ liệu (SIMD)

Tất cả các bộ vi xử lý trên một máy SIMD thực hiện cùng một lệnh một cách đồng bộ trên các dữ liệu khác nhau.

Trong một máy SIMD, nhiều thành phần xử lý được giám sát bởi một đơn vị điều khiển. Tất cả những thành phần xử lý đều nhận cùng mệnh lệnh từ đơn vị điều khiển nhưng lại thực hiện trên những tập dữ liệu khác nhau và đến từ những luồng dữ liệu khác nhau. Một máy SIMD biểu diễn ở hình 1.12 có những đặc điểm sau: xử lý phân tán trên một số lượng lớn phần cứng, thực hiện đồng thời trên nhiều thành phần dữ liệu khác nhau và thực hiện cùng một câu lệnh trên các thành phần dữ liệu.



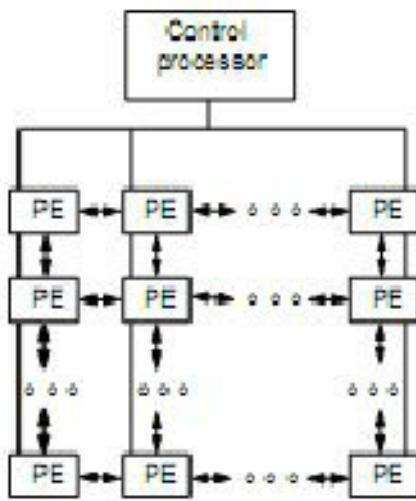
Hình 1.12. Kiến trúc máy tính SIMD

Máy tính SIMD đóng vai trò quan trọng trong xử lý song song. Chúng có thể thao tác trên các dữ liệu biểu diễn dưới dạng vector hay ma trận; trên thực tế các dữ liệu về thời tiết hoặc các nghiên cứu về bức xạ gây ung thư thường được biểu diễn dưới dạng véc tơ. Máy SIMD có thể xử lý các bài toán này trong một khoảng thời gian ngắn hơn so với các mô hình khác. Trong trường hợp kích thước của vector đúng bằng số lượng bộ vi xử lý thì hiệu suất của máy SIMD sẽ đạt tối ưu. Trong trường hợp mà số bộ vi xử lý và kích thước vector khác nhau, thì tốc độ xử lý của máy tính SIMD cũng tốt hơn nhiều so với máy tính tuần tự.

Dưới đây ta tìm hiểu về 3 loại máy tính SIMD :

a. *Máy tính SIMD với bộ nhớ phân tán:*

SIMD với bộ nhớ phân tán gồm một bộ điều khiển (Control Unit) với nhiều bộ xử lý (Processing Elements). Các bộ xử lý hoạt động giống như các bộ tính toán số học (Arithmetic Unit). Các bộ tính toán số học là thợ (slaver) được điều khiển bởi bộ điều khiển là chủ (Master). Các bộ tính toán số học không thể đọc hoặc giả mã lệnh, chúng chỉ là các bộ tính toán thuần túy có thể thực hiện được các phép tính cộng, trừ, nhân, và chia. Mỗi bộ tính toán số học có thể truy cập tới bộ nhớ cục bộ của chúng. Trong trường hợp, bộ tính toán số học này cần thông tin chứa trong bộ tính toán số học khác, nó phải gửi yêu cầu đến bộ điều khiển và bộ điều khiển sẽ làm nhiệm vụ đọc thông tin của bộ tính toán số học đó và gửi đến bộ tính toán số học đã yêu cầu.



Hình 1.13 Máy tính SIMD với bộ nhớ phân tán

Ưu điểm của kiến trúc này là ta có thể rất dễ dàng mở rộng bộ nhớ toàn cục cũng như cục bộ trên mỗi bộ tính toán số học.

Nhược điểm dễ nhận thấy là mất nhiều thời gian khi bộ điều khiển phải quản lý tất cả các trao đổi thông tin giữa các bộ tính toán số học. Trong trường hợp phải thực hiện một chương trình mà có nhu cầu trao đổi dữ liệu lớn giữa các bộ xử lý thì thời gian chờ sẽ làm hạn chế hiệu suất của hệ thống tính toán.

b. Máy tính SIMD với bộ nhớ chia sẻ:

Một kiến trúc khác của SIMD được thiết kế với sự kết hợp giữa các bộ xử lý (Processing Element) với các mô đun bộ nhớ. Trong kiến trúc này, bộ nhớ cục bộ của mỗi bộ tính toán số học ở trên được thay thế bằng các mô đun nhớ. Các mô đun nhớ này được chia sẻ cho tất cả các bộ xử lý thông qua mạng hoặc các thiết bị chuyển mạch. Điều này cho phép các bộ xử lý truy cập đến bộ nhớ chia sẻ mà không phải truy cập qua bộ điều khiển như ở trên; đây cũng là ưu điểm của kiến trúc này. Nhược điểm của kiến trúc này là khó khăn khi ta muốn mở rộng các mô đun nhớ vì bị giới hạn của không gian địa chỉ của máy tính.

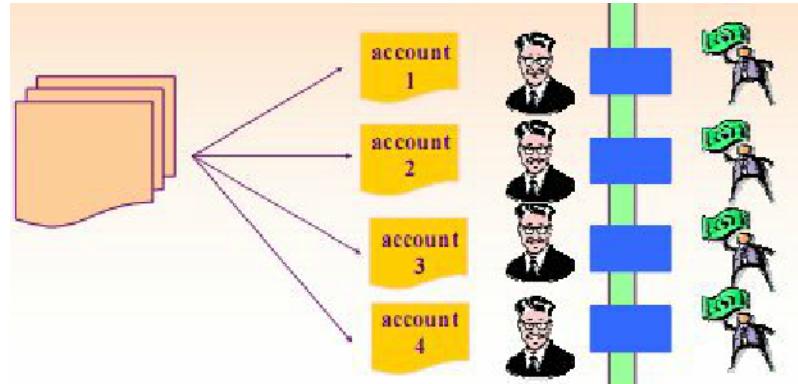
c. Máy tính SIMD có sự hỗ trợ của pipeline

Kiến trúc SIMD có sự hỗ trợ của pipeline bao gồm một bộ tính toán số học có hỗ trợ pipeline với bộ nhớ chia sẻ. Xử lý pipeline có thể đọc các lệnh từ các dòng mã lệnh (instruction streams) khác nhau và thực hiện chúng cùng một bộ tính toán số học. Pipeline là một thủ tục xử lý theo kiểu vào trước ra trước (First In First Out) và hiệu xuất pipeline là tương đối. Để tăng hiệu quả của xử lý pipeline thì dữ liệu được lưu trong bộ nhớ chia sẻ để bộ xử lý có thể đọc chúng càng nhanh càng tốt. Ưu điểm có thể nhận thấy là tốc độ và hiệu quả của xử lý dữ liệu với yêu cầu trên được thỏa mãn..

d. Một số ví dụ

Ví dụ 1: Bài toán banking

Bài toán banking (đã đề cập ở phần 1.2.1) thực hiện trên máy SIMD như sau:



Hình 1.14 Giải bài toán Banking trên Máy tính SIMD

Tất cả các bộ xử lý sẽ cùng thực hiện một lệnh (hoặc chờ) tại cùng một thời điểm trên các dữ liệu khác nhau.

Bước 1: phân chia (partition) dữ liệu cho các bộ vi xử lý (máy ATM)

Bước 2: thực hiện các giao dịch một cách song song.

Hiệu suất phụ thuộc vào có bao nhiêu giao dịch được thực hiện song song.

Giả sử khối lượng công việc được phân chia đều (balanced) thì:

Gọi t là thời gian để phân phát dữ liệu và gửi trả về kết quả thì:

Thời gian tốt nhất là: $T = t + \max \{t_1, t_2, \dots, t_n\}$

Ví dụ 2: Bài toán quản lý nhân viên

Thông tin của mỗi nhân viên là một bản ghi, và có trường salary chứa thông tin về hệ số lương của từng nhân viên.

Bước 1: Phân phát mỗi một bộ xử lý (PE) lưu trữ một bản ghi về nhân viên.

Bước 2: Đoạn chương trình sau được thực hiện đồng thời:

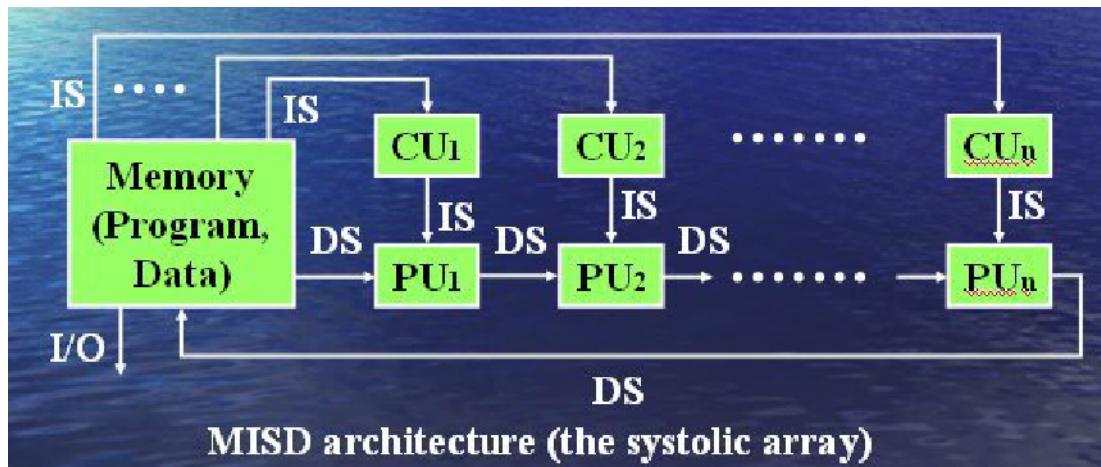
```
If salary >100K then  
    Salary=salary * 1.05  
Else  
    Salary=salary * 1.10
```

Toàn bộ lương của nhân viên được tính trong 1 bước. Sẽ có một số bộ xử lý được thực hiện, còn lại sẽ không được kích hoạt (disabled).

1.2.4 Máy tính nhiều dòng lệnh, một dòng dữ liệu (MISD)

MISD là một kiến trúc song song mà trong đó tại mỗi thời điểm các bộ xử lý thực hiện các thao tác khác nhau trên cùng một dữ liệu. Máy tính MISD có ý nghĩa về mặt lý thuyết nhiều hơn thực tế. Trên thực tế, chưa có máy tính MISD nào được thương mại hóa.

Máy tính MISD còn được gọi là máy tính với các mảng Systolic (Systolic arrays machine), chúng đóng vai trò đối với các bài toán “làm mịn” dữ liệu bằng cách “bơm” (pump) dữ liệu từ bộ xử lý này đến bộ xử lý khác. Mỗi bộ vi xử lý có thể thay đổi dữ liệu trước khi chuyển dữ liệu sang bộ xử lý khác. Các thao tác thay đổi dữ liệu trên các bộ vi xử lý có thể khác nhau.



Hình 1.15. Kiến trúc MISD

IS : Instruction Stream – dòng lệnh

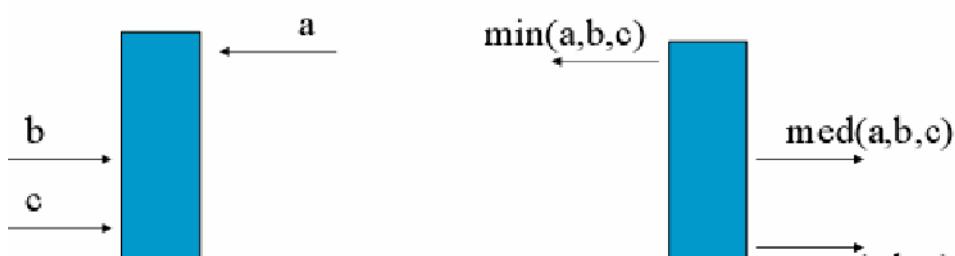
DS: Data Stream – dòng dữ liệu

CU : Control Unit – Bộ điều khiển

PU : Processing Unit – Bộ xử lý.

Hai ứng dụng trong thực tế mà máy tính MISD có thể nhắm tới là hệ thống hỗ trợ xử lý đa pipeline (hyper pipeline) và hệ thống dung thứ lỗi (fault tolerance).

Ví dụ minh họa: chương trình tìm max, min và medium của 3 số.



Hình 1.16. Một ví dụ thực hiện trên máy tính MISD

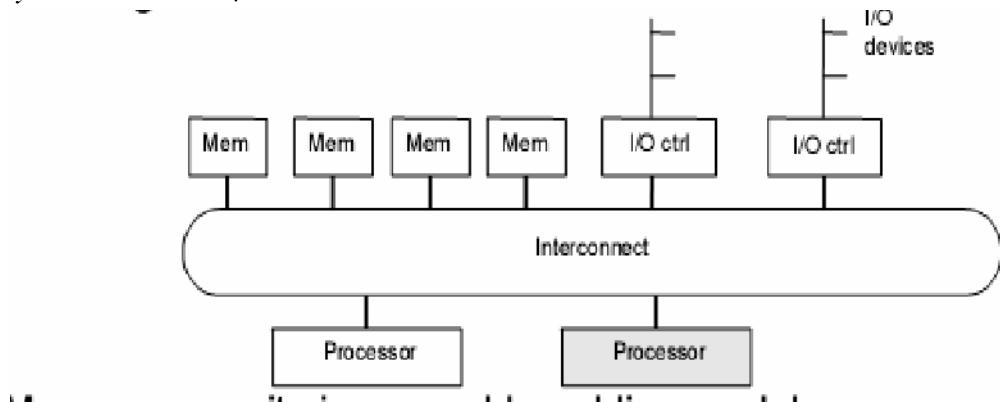
Với một dòng dữ liệu đầu vào gồm các số a,b và c. Dòng dữ liệu này đi qua máy tính MISD là được máy tính MISD thực hiện đồng thời 3 thao tác: tìm giá trị lớn nhất, bé nhất và trung bình của ba số trên.

1.2.5 Máy tính nhiều dòng lệnh, nhiều dòng dữ liệu (MIMD)

Máy tính MIMD còn được gọi là máy tính đa bộ xử lý (multiprocessors) trong đó các bộ xử lý có các chức năng độc lập nhau. Tại bất cứ thời điểm nào, các bộ xử lý của máy tính MIMD có thể thực hiện các lệnh khác nhau trên các dữ liệu khác nhau một cách *không đồng bộ*. Máy tính MIMD đã và đang được ứng dụng rất rộng rãi trên nhiều lĩnh vực như: thiết kế và sản xuất có sự trợ giúp của máy tính (computer-aided design/ computer-aided manufacturing), mô phỏng, mô hình hóa, chuyển mạch ... Các máy tính MIMD có thể có bộ nhớ chia sẻ hoặc phân tán tùy thuộc vào cách thực truy cập bộ nhớ của các bộ vi xử lý.

Dưới đây là một số loại máy tính MIMD thông dụng.

a. *Máy tính MIMD với bộ nhớ chia sẻ*



Hình 1.17. Kiến trúc máy tính MIMD

Máy tính MIMD với bộ nhớ chia sẻ bao gồm các bộ vi xử lý cùng chia sẻ một bộ nhớ toàn cục. Bộ nhớ toàn cục là một bộ nhớ được quản lý tập trung mà các bộ vi xử lý đều có thể truy cập được địa chỉ bất kỳ. Bộ nhớ toàn cục bao gồm nhiều mô đun nhớ.

Máy tính MIMD với bộ nhớ chia sẻ có thể dễ dàng nâng cấp (scalability) để đáp ứng nhu cầu tính toán: dễ bổ sung các mô đun bộ nhớ và bộ vi xử lý. Việc quản lý trao đổi vào/ ra được thực hiện bởi các bộ điều khiển (Control unit) hoặc thiết bị vào ra (I/O devices).

Trên thực tế có 3 loại máy tính MIMD với bộ nhớ chia sẻ:

- *Máy tính MIMD với bộ nhớ chia sẻ dựa trên bus (bus-based shared memory MIMD machines):* Tất cả các bộ xử lý được kết nối với nhau thông qua một hệ thống Bus và hệ thống Bus này cũng được kết nối tới các bộ xử lý. Tùy theo thiết kế hoặc nhu cầu thực tế mà loại máy tính này có thể có các loại bus khác nhau hoặc cho phép các bộ vi xử lý giao tiếp trực tiếp với nhau. Thông thường thì các loại bus này cũng được sử dụng để đồng bộ giữa các bộ xử lý. Máy tính MIMD dựa trên bus có nhược điểm là tại một thời điểm, không có nhiều bộ vi xử lý được hỗ trợ truy cập bộ nhớ vì khả năng tranh chấp đường truyền (contention) khi sử dụng bus để truy cập bộ nhớ toàn cục là rất lớn. Chúng phải tuân theo cơ chế trọng tài bus trong khi sử dụng các bus chung để truyền tải dữ liệu. Khi nhu cầu về trao đổi dữ liệu càng lớn, thì khả năng tranh chấp đường truyền và thời gian chờ giữa các bộ xử lý càng cao. Điều này làm ảnh hưởng đến hiệu suất chung của toàn hệ thống.

- *Máy tính MIMD với bộ nhớ chia sẻ mở rộng (extended shared memory MIMD machines)*: Máy tính MIMD với bộ nhớ chia sẻ mở rộng được sinh ra để khắc phục tình trạng tranh chấp đường truyền khi sử dụng bus để truy cập bộ nhớ toàn cục. Bằng cách chia nhỏ bộ nhớ toàn cục thành các đơn vị bộ nhớ nhỏ hơn nhưng độc lập với nhau. Những đơn vị bộ nhớ này sẽ được kết nối với các bộ vi xử lý bằng một liên mạng (interconnection network). Các đơn vị bộ nhớ được xem như một bộ nhớ tập trung hợp nhất. Liên mạng được sử dụng trong kiến trúc này có thể là một mạng chuyển mạch chéo (crossbar switching network). Trong liên mạng này, nếu có N bộ vi xử lý được kết nối đến M đơn vị bộ nhớ và cần $N \times M$ chuyển mạch. Số chuyển mạch là khá lớn khi mở rộng kiến trúc; gây tốn kém về mặt kinh tế nếu phải kết nối một số lượng lớn bộ xử lý.

- *Máy tính MIMD với bộ nhớ chia sẻ phân cấp (hierarchical shared memory MIMD machines)*: Máy tính MIMD với bộ nhớ chia sẻ phân cấp sử dụng một hệ thống bus phân cấp cho phép các bộ vi xử lý có thể truy cập bộ nhớ toàn cục. Các bus nội bộ được sử dụng để giao tiếp giữa các bộ vi xử lý trên cùng một bo mạch (board). Các bus ngoài cho phép các bộ vi xử lý nằm trên các bo mạch khác nhau có thể giao tiếp được với nhau. Với kiến trúc này máy tính có thể hỗ trợ mở rộng đến hàng ngàn bộ vi xử lý.

b. Máy tính MIMD với bộ nhớ phân tán

Trong máy tính MIMD với bộ nhớ phân tán, mỗi bộ vi xử lý có riêng một bộ nhớ cục bộ. Các dữ liệu cần chia sẻ được truyền từ bộ xử lý này đến bộ xử lý khác như các thông điệp (message). Vì không có bộ nhớ chia sẻ nên việc tranh chấp đường truyền (contention) không còn là hạn chế đối với loại máy này. Tuy nhiên, mô hình này đòi hỏi chi phí cao trong trường hợp có nhiều bộ vi xử lý vì chúng được kết nối trực tiếp với nhau. Một cách để giảm bớt chi phí là hạn chế số lượng bộ xử lý kết nối trực tiếp: một bộ vi xử lý có thể kết nối với 1 số bộ vi xử lý khác thay vì kết nối với tất cả các bộ vi xử lý còn lại. Cách thiết kế này trên thực tế lại kém hiệu quả về thời gian yêu cầu để truyền thông điệp giữa hai bộ vi xử lý vì các thông điệp phải đi qua các bộ vi xử lý trung gian. Có hai cách tổ chức bộ vi xử lý thông dụng có thể giảm được thời gian truyền tải các thông điệp giữa các bộ vi xử lý đó là: tổ chức theo mạng hình siêu khối và hình lưới mà ta sẽ thảo luận ở các phần sau.

1.2.6 Hiệu suất của Máy tính song song

Hiệu suất (performance) là số lượng công việc được hoàn thành trong một đơn vị thời gian (work done per time unit).

Ví dụ: về tính hiệu suất của một máy tính SIMD

Tính tổng hai vector có kích thước 1024 A và B,

Biết rằng : thời gian để thực hiện 1 phép cộng là 1 μ s.

Bước 1: bộ vi xử lý thứ i sẽ nhận các phần tử A[i] và B[i] với $i = 1..1024$.

Bước 2: thực hiện cộng chúng một cách song song.

Tính hiệu Suất trong 2 trường hợp :

Trường hợp 1: Số bộ vi xử lý là 1024.

$$P = (1024 \text{ phép tính}) / (1 \text{ chu kỳ} * 1 \mu\text{s}) = 1024 * 10^9 \text{ ops/s}$$

Trường hợp 2: Số bộ vi xử lý là 1000.

$$P = (1024 \text{ phép tính}) / (2 \text{ chu kỳ} * 1 \mu\text{s}) = 512 * 10^9 \text{ ops/s}$$

Chỉ có 24 bộ xử lý thực hiện trong chu kỳ thứ 2, 1000 bộ xử lý khác idle. Điều kiện để một bộ vi xử lý có thể idle là do không có dữ liệu hoặc là do câu lệnh có điều kiện.

1.3 Tổ chức các bộ vi xử lý

Trong phần này ta sẽ thảo luận 9 mô hình tổ chức các bộ vi xử lý quan trọng của sự kết nối các bộ vi xử lý trong các máy tính song song. Một mô hình tổ chức các bộ vi xử lý có thể được biểu diễn bởi một đồ thị trong đó các đỉnh (vertices) hoặc nút (node) biểu diễn các bộ vi xử lý và các cạnh (edges) biểu diễn các đường truyền thông giữa các cặp bộ vi xử lý. Dưới đây là các tiêu chuẩn để đánh giá tính hiệu quả của các mô hình tổ chức các bộ vi xử lý đối với các thuật toán song song thực hiện trên phân cứng.

Đường kính (diameter) : của một mạng các bộ vi xử lý là khoảng cách lớn nhất giữa hai nút.

Đường kính càng nhỏ thì càng tốt vì đường kính là cận dưới của độ phức tạp về truyền thông của các thuật toán song song. Nó chỉ số bộ vi xử lý trung gian ít nhất mà thông điệp phải đi qua giữa hai bộ vi xử lý bất kỳ.

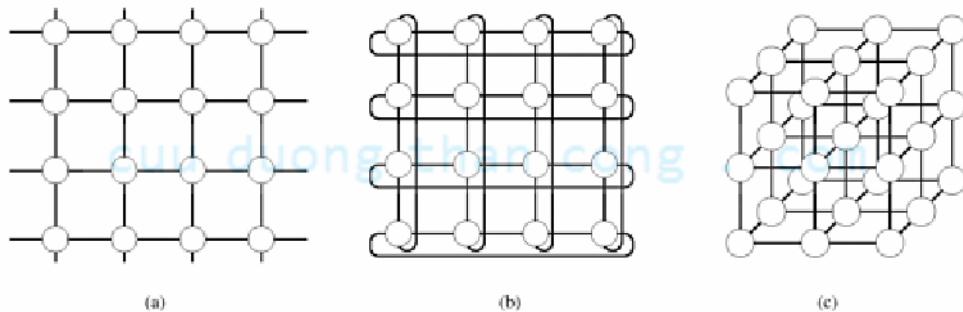
Độ rộng phân đôi (Bisection width): của một mạng các bộ vi xử lý là số cạnh nhỏ nhất có thể bỏ đi để chia mạng thành hai nửa (hơn kém nhau không quá 1 nút). Độ rộng phân đôi càng lớn thì càng tốt bởi vì trong các thuật toán yêu cầu một lượng lớn dữ liệu được di chuyển thì kích thước của tập dữ liệu chia cho độ rộng phân đôi sẽ là cận dưới của độ phức tạp về truyền thông của một thuật toán song song.

Số cạnh của một nút (number of edges per node): Trường hợp tốt nhất là số cạnh của một nút là một hằng số độc lập với kích thước của mạng; Bởi vì, nếu vậy ta sẽ dễ dàng mở rộng mạng với một số lượng nút lớn.

1.3.1 Mạng hình lưới (Mesh)

Trong một mạng hình lưới, các nút được sắp xếp trong một lưới q-chieu. Chỉ các nút lân cận có thể giao tiếp được với nhau vì vậy một nút trong có thể giao tiếp được với $2q$ nút.

Một số biến thể của mạng hình lưới cho phép các kết nối vòng (wrap-around) giữa các bộ vi xử lý trên các cạnh của hình lưới. Các kết nối này có thể kết nối các bộ vi xử lý trên cùng một hàng hoặc cột hoặc các cạnh bên của hàng hoặc cột.



Hình 1.18. Mạng hình lưới

Ta hãy đánh giá mạng hình lưới theo các tiêu chuẩn ở trên:

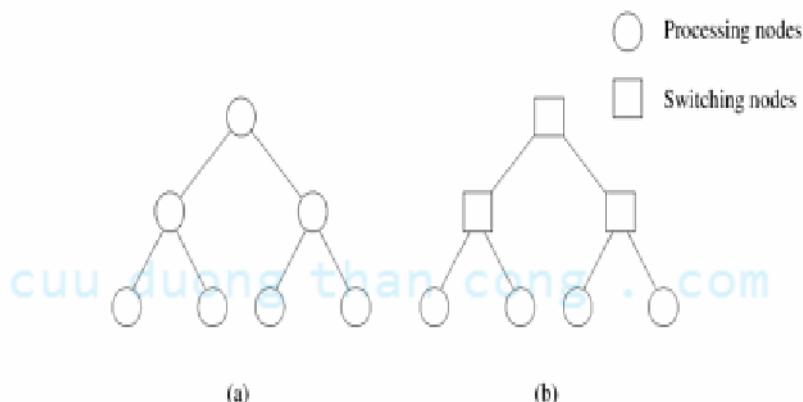
Giả sử một mạng hình lưới không chứa các kết nối vòng. Đường kính của một mạng hình lưới q chiều với k^q nút là $q(k-1)$. Vì vậy, theo quan điểm trên, mạng hình lưới sẽ có nhược điểm là sự di chuyển dữ liệu sẽ làm tăng độ phức tạp về truyền thông của thuật toán song song thực hiện trên hình lưới. Tuy nhiên trong thực tế, các máy tính song song tổ chức các bộ xử lý theo mạng hình lưới thực thi một số kết nối nhanh để làm giảm độ phức tạp về truyền thông.

Độ rộng phân đôi của một mạng hình lưới q chiều là k^{q-1} và số cạnh của một nút là $2q$.

Tổ chức CPU theo mạng Mesh 2 chiều được sử dụng rộng rãi các máy tính mảng bộ vi xử lý như: Goodyear Aerospace 's MPPTM, the AMT DAPTM, Intel Paragon và MP-1TM của MasPar.

1.3.2 Mạng hình cây nhị phân (Binary Tree Networks)

Trong cây nhị phân, $2^k - 1$ nút được sắp xếp trong một cây nhị phân hoàn chỉnh với chiều cao $k-1$. Mỗi nút có nhiều nhất là 3 liên kết. Mỗi nút bên trong (nếu không là nút gốc) có thể giao tiếp được với 2 nút con và 1 nút cha của nó.



Hình 1.19. Mạng hình cây nhị phân

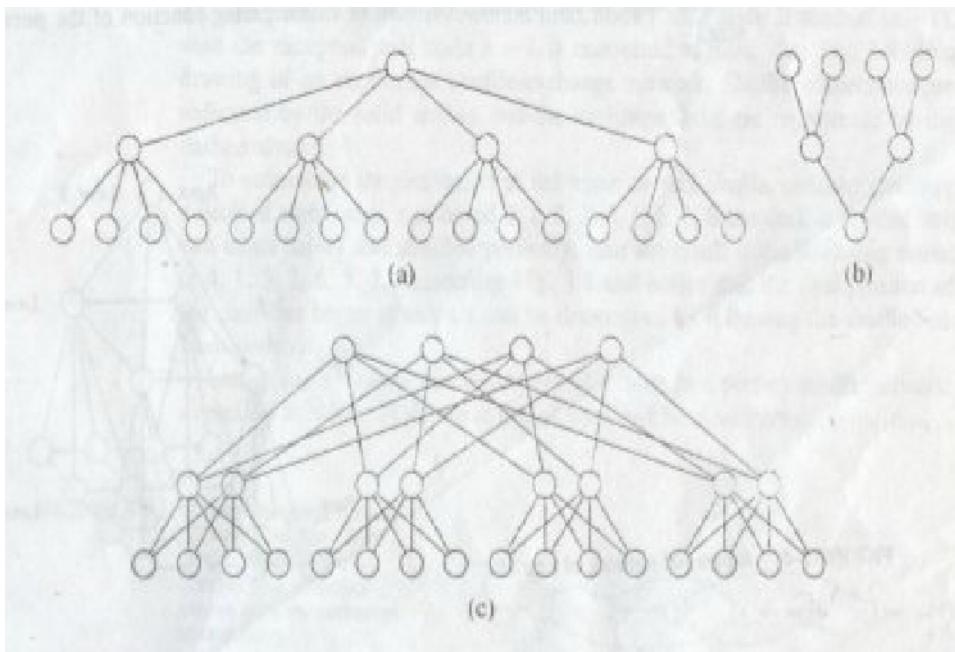
Mạng hình cây nhị phân có 2 dạng: a) mạng tĩnh trong đó mỗi nút đều là các phần tử xử lý và b) là mạng động trong đó chỉ có nút là là các phần tử xử lý; các nút trong là các nút chuyển mạch (cho việc di chuyển dữ liệu giữa các nút).

Mạng hình cây nhị phân có đường kính là $2(k-1)$, độ rộng phân đôi là 1.

Như vậy, mạng hình cây nhị phân có đường kính tốt (nhỏ), nhưng độ rộng phân đôi không tốt.

1.3.3 Mạng hình siêu cây (Hypertree networks)

Mạng hình siêu cây thể hiện một cách tiếp cận để xây dựng một mạng với đường kính nhỏ của mạng cây nhị phân và cải tiến độ rộng phân đôi. Một mạng siêu cây cấp k và độ sâu d được nhìn từ hai phía: nhìn phía mặt trước (front view) như một k -cây với chiều cao d . Nhìn từ phía bên cạnh (side view) siêu cây như một cây nhị phân chiều cao d từ dưới lên trên. Kết hợp giữa nhìn phía trước và bên cạnh là mạng hoàn chỉnh. Hình 1.20c là một mạng với cấp 4 chiều cao 2.



Hình 1.20. Một mạng hình siêu cây; a. nhìn từ mặt trước; b. nhìn từ bên cạnh; c. nhìn đầy đủ

Xét một 4-siêu cây với độ cao d có 4^d lá và $2^d(2^{d+1}-1)$ nút. Đường kính của mạng này là $2d$, độ rộng phân đôi là 2^{d+1} . Số cạnh của một nút không vượt quá 6.

Một ví dụ về máy tính song song có cách tổ chức bộ xử lý theo siêu cây (4-siêu cây) là Connection Machine CM-5 multicomputer.

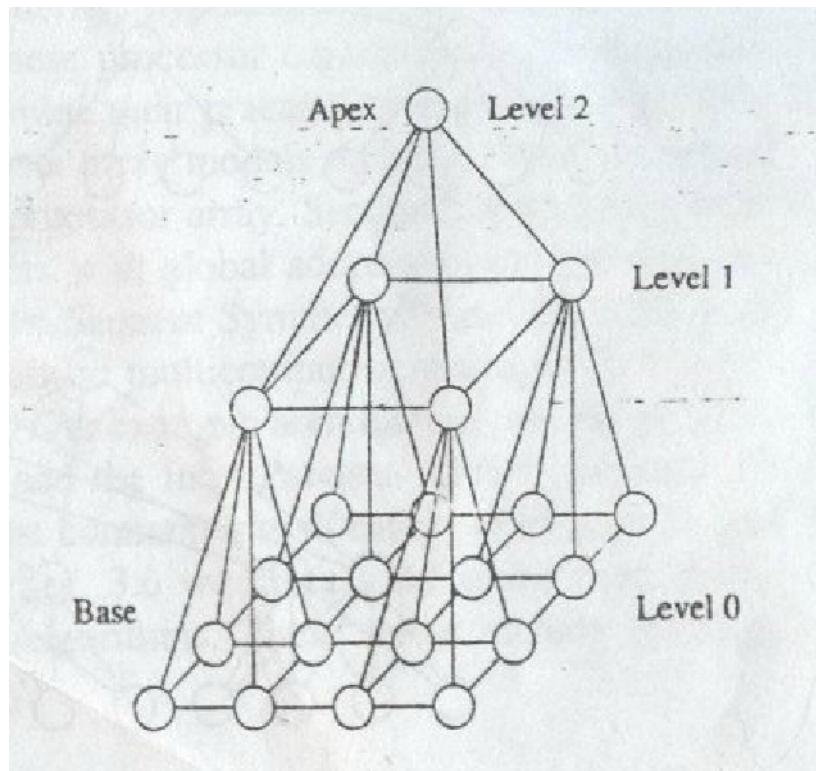
www.cuuduongthancong.com

1.3.4 Mạng hình tháp (Pyramid networks)

Mạng hình tháp là sự kết hợp các ưu điểm của mạng hình lưới và mạng hình cây. Một mạng hình tháp kích thước k^2 là một 4-cây với độ cao $\log_2 k$ và bổ sung các liên kết giữa các bộ vi xử lý sao cho các bộ vi xử lý trong mỗi mức của mạng hình thành một mạng 2-D mesh (Miller & Stout 1987).

Một mạng hình tháp kích thước k^2 có nền là một 2-D mesh với k^2 bộ vi xử lý. Tổng số bộ vi xử lý trong mạng hình tháp kích thước k^2 là $(4/3)k^2 - (1/3)$. Các mức của mạng hình tháp được đánh số theo chiều tăng và mạng Mesh nền có mức 0, và có một bộ vi xử lý đơn ở đỉnh có mức $\log_2 k$. Mỗi bộ vi xử lý phía trong được kết nối với 9 bộ vi xử lý khác; trong đó, 1 nút cha, 4 nút trong cùng mức và 4 nút con. Hình 1.21 là một mạng hình tháp có kích thước 16.

www.cuuduongthancong.com



Hình 1.21. Một mạng hình tháp có kích thước 16.

Một ưu điểm của mạng hình tháp là trên các mạng hình lưới 2 chiều thì đường kính sẽ giảm. Chẳng hạn, khi một thông điệp được trao đổi từ hai bộ vi xử lý bất kỳ trong mạng thì sẽ di chuyển từ mức này đến mức khác sẽ phải qua các bộ vi xử lý trung gian ít hơn là di chuyển bên trong mạng hình lưới.

Đường kính của một mạng hình tháp kích thước k^2 là $2 \log_2 k$.

Khi bổ sung thêm các liên kết không làm tăng đáng kể độ rộng phân đôi. Độ rộng phân đôi đối với một mạng hình tháp kích thước k^2 là $2k$.

Số cạnh trên mỗi nút không vượt quá 9.

1.3.5 Mạng hình bướm (Butterfly networks)

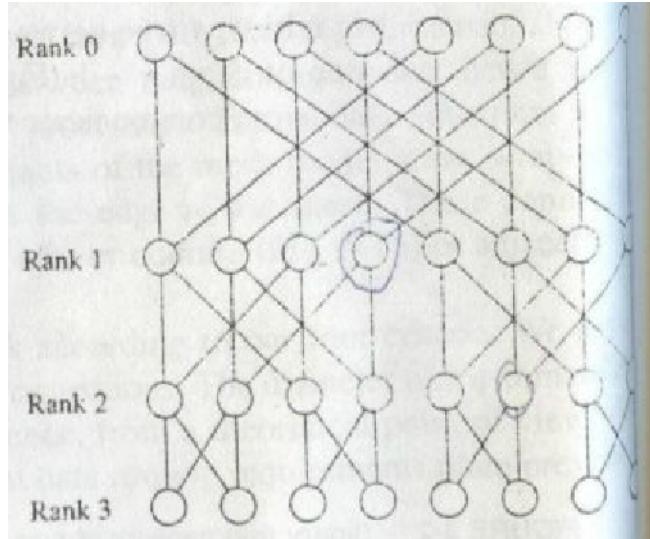
Một mạng hình bướm bao gồm $(k+1)2^k$ nút được chia vào $k+1$ hàng hay bậc (ranks); mỗi hàng chứa $n=2^k$ nút. Các hàng được gán nhãn từ 0 đến k , hàng 0 và k đôi khi được kết hợp lại để cho mỗi nút được kết hợp với 4 nút khác.

Giả sử nút (i,j) là nút thứ j trên hàng i , với $0 \leq i \leq k$ và $0 \leq j \leq n$. Thì nút (i,j) trên hàng $i > 0$ được kết nối tới 2 nút ở hàng $i-1$ là $(i-1,j)$ và $(i-1,m)$ với m là số nguyên khi ta đảo ngược bit thứ i bên trái nhất (the i th most significant bit) trong biểu diễn nhị phân của j .

Ví dụ: hình 1.22, nút $(2,5)$ ở hàng 2 cột 5 sẽ kết nối tới 2 nút là $(1,5)$ và $(1,7)$ vì:

$5=0101$, bit thứ 2 ($i=1$) là tính từ bên trái nhất là 0 ($0,1,2$) sau khi đảo sẽ là $0111=7$.

Chú ý rằng, nếu nút (i,j) được kết nối tới nút $(i-1,m)$ thì nút (i,m) được kết nối tới nút $(i-1,j)$. Vì vậy, toàn bộ mạng được tổ chức giống như hình bướm nên có tên như vậy.
Khi số hàng tăng, thì độ rộng của “cánh” bướm sẽ tăng theo hàm số mũ (exponentially).



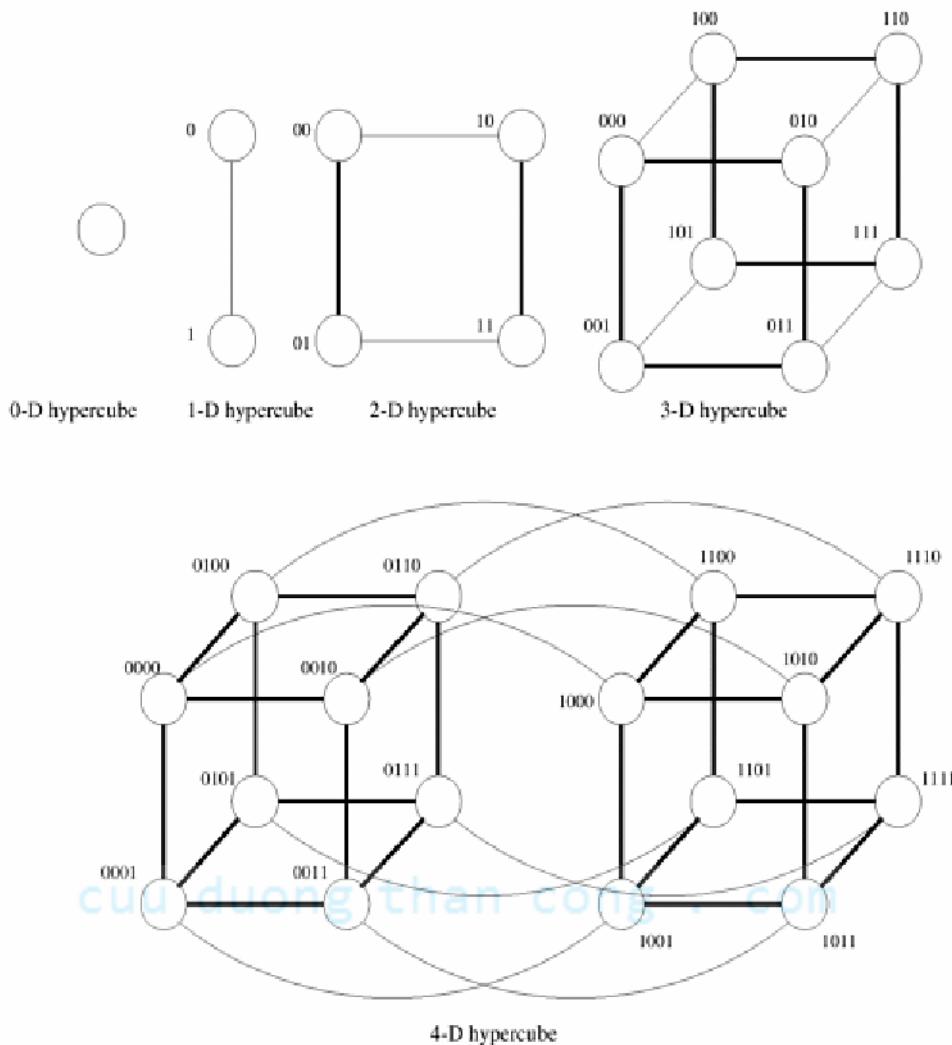
Hình 1.22. Một mạng hình bướm

Đường kính của một mạng hình bướm với $(k+1)2^k$ nút là $2k$ và độ rộng phân đôi là 2^{k-1} .
Một số máy tính song song dựa trên mô hình này như máy tính đa bộ xử lý BBN TC 2000.

1.3.6 Mạng hình siêu khối (Hypercube networks)

Một mạng kết nối kiểu siêu khối còn được gọi là một n-lập phương nhị phân. Bao gồm 2^k nút hình thành lên một siêu khối k-chiều. Các nút được gán nhãn $0, 1, 2, \dots, 2^k - 1$. Hai nút cạnh nhau nếu chúng khác nhau chính xác 1 vị trí bít. Hình 1.23 là ví dụ về các mạng siêu khối 0, 1, 2, 3, 4 chiều.

cuu duong than cong . com



Hình 1.23. Các mạng siêu khối 0,1,2,3,4 chiều

Đường kính của một mạng siêu khối với 2^k nút là k , độ rộng phân đôi là 2^{k-1} . So với các mạng khác thì mạng siêu khối có đường kính nhỏ so với kích thước của mạng.

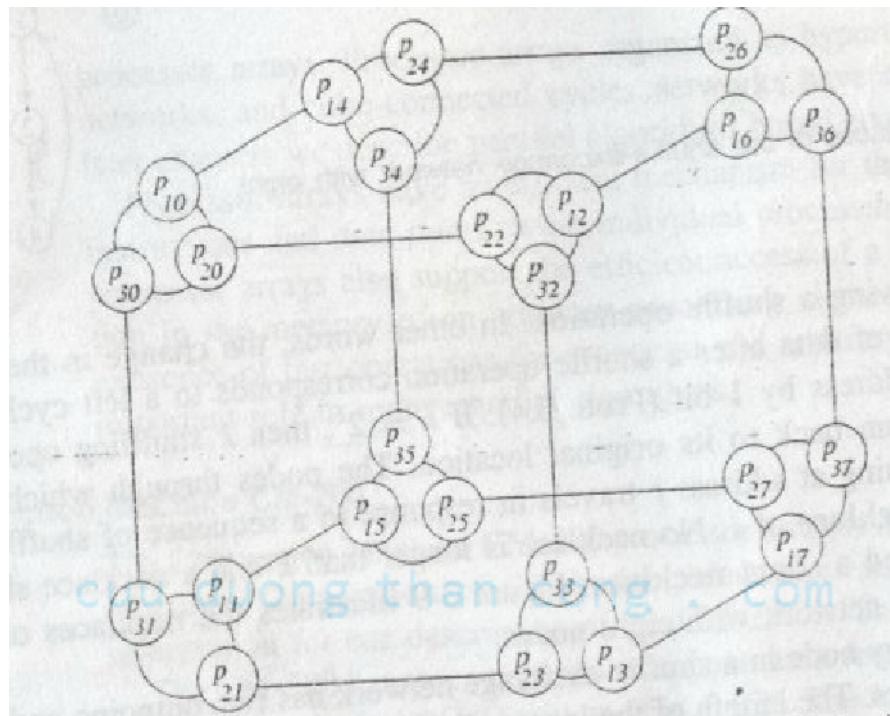
Số cạnh trên nút là k , chính bằng logarit của số nút trên mạng.

Mạng hình siêu khối đã từng trở nên thông dụng nhất đối với cách tổ chức các bộ vi xử lý cho các máy tính đa bộ vi xử lý thế hệ thứ nhất và thứ hai. Một số máy tính song song dựa trên mô hình này là: nCUBE, máy tính mảng các bộ xử lý Connection Machine CM-200.

1.3.7 Mạng các chu trình hướng kết nối khối (Cube-Connected Cycles networks)

Mạng các chu trình hướng kết nối khối là mạng siêu khối k chiều với 2^k đỉnh. Các chu trình gồm k nút, được hình thành bởi các cột của mạng hình bướm hàng 0 và k kết hợp lại. Với mỗi chiều, mọi chu trình có 1 nút được kết nối tới 1 nút trong chu trình bên cạnh của chiều đó.

Nút (i,j) được kết nối tới nút (i,m) nếu và chỉ nếu m là kết quả khi ta đảo ngược bit thứ i bên trái nhất (the i th most significant bit) trong biểu diễn nhị phân của j . Chú ý rằng, các kết nối khác với mạng hình bướm 1 chút ở chỗ: nếu nút (i,j) được kết nối tới nút $(i-1,m)$ trong mạng hình bướm khi $j \neq m$, thì nút (i,j) được kết nối tới nút (i,m) trong mạng các chu trình hướng kết nối khói. Tuy nhiên, trong mạng các chu trình hướng kết nối khói, nút (i,j) có thể giao tiếp với nút $(i-1,m)$ theo 2 cách vì có một kết nối trực tiếp từ nút (i,m) tới $(i-1,m)$.



Hình 1.24. Mạng các chu trình hướng kết nối khói 3 chiều

So với mạng siêu khói, tổ chức các bộ vi xử lý theo mạng các chu trình hướng kết nối khói có ưu điểm là: số cạnh trên nút bằng 3 và là hằng số không phụ thuộc vào kích thước của mạng. Tuy nhiên, mạng các chu trình hướng kết nối khói có nhược điểm là: đường kính của mạng gấp 2 lần đường kính của mạng hình siêu khói và độ rộng phân đôi thấp hơn.

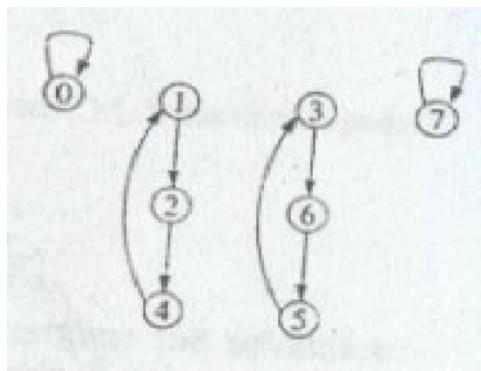
Một mạng các chu trình hướng kết nối khói kích thước $k2^k$ có đường kính là $2k$ và độ rộng phân đôi là 2^{k-1} .

1.3.8 Mạng hoán vị di chuyển (Shuffle-exchange networks)

Một mạng hoán vị di chuyển bao gồm $n=2^k$ nút, được đánh số từ $0,1,2,\dots,n-1$, và có hai loại kết nối là di chuyển (shuffle) và hoán vị (exchange). Các kết nối hoán vị liên kết các cặp nút thứ i với nút thứ $2i$ modulo $(n-1)$; trừ nút thứ $n-1$ được kết nối đến chính nó. Hình 1.26 là một mạng hoán vị di chuyển với 8 nút.

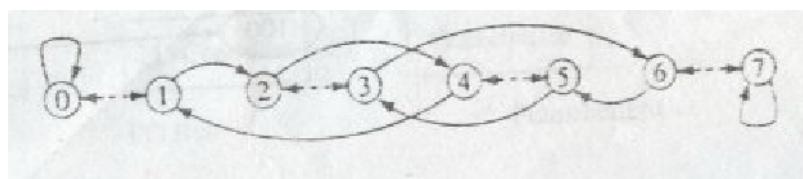
Mạng hoán vị di chuyển còn có một tên khác đó là mạng di chuyển hoàn hảo (perfect shuffle). Để dễ hình dung ta xem sự di chuyển của 8 quân bài được đánh số 0,1,2,3,4,5,6,7 trên một chiếc bàn. Nếu chiếc bàn được chia làm hai nửa và sự di chuyển là hoàn hảo thì kết quả cho ta thứ tự sau: 0,4,1,5,2,6,3,7. Ta có thể kiểm nghiệm điều này qua hình 1.25 và để ý rằng vị trí cuối cùng của quân bài bắt đầu với chỉ số i có thể được xác định theo sau sự di chuyển liên kết từ nút i.

Giả sử, $a_{k-1} a_{k-2} \dots a_1 a_0$ là địa chỉ dưới dạng nhị phân của một nút trong một mạng di chuyển hoàn hảo. Dữ liệu này có thể liên kết tới địa chỉ $a_{k-2} \dots a_1 a_0 a_{k-1}$, theo sau là một thao tác hoán chuyển. Nói một cách khác, sự thay đổi chỉ của một mẩu dữ liệu sau một thao tác di chuyển tương đương với việc quay trái (left cyclic rotate) địa chỉ 1 bit. Nếu $n=2^k$, thi cần k thao tác di chuyển để chuyển một mục dữ liệu về vị trí ban đầu. Những nút trung gian mà mục dữ liệu bắt đầu ở địa chỉ i di chuyển theo yêu cầu của một dãy các thao tác di chuyển được gọi là chuỗi di chuyển của i (necklace). Nếu một chuỗi di chuyển của i nhỏ hơn k thì được gọi là một chuỗi di chuyển ngắn.



cuu duong than cong . com

Hình 1.25. Thao tác hoán chuyển trên một mạng hoán vị di chuyển.



Hình 1.26. Một mạng hoán vị di chuyển với 8 nút

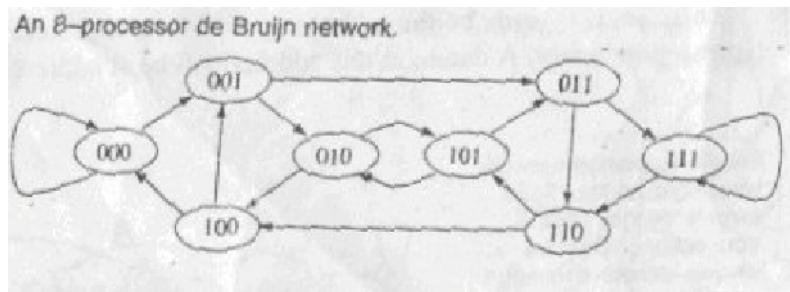
Mỗi nút trong mạng hoán vị di chuyển đều có hai nút vào và hai nút ra. Độ dài của liên kết dài nhất tăng theo một hàm của kích thước mạng. Số liên kết dài sẽ cải tiến đường kính và độ rộng phân đôi trong mạng hoán vị-di chuyển. Trong mạng hoán vị-di chuyển thì đường kính là lô gô rít đối với số nút. Một mạng với 2^k nút thì đường kính là $2k-1$ và độ rộng phân đôi bé nhất là $(2^{k-1}/k)$.

Siegel (1979) đã phát triển một cải tiến của mạng hoán vị-di chuyển là mạng Omega, tương đương với mạng hình siêu khối cấp độ k.

1.3.9 Mạng de Bruijn

Mạng de Bruijn bao gồm $n=2^k$ nút. Giả sử $a_{k-1} a_{k-2} \dots a_1 a_0$ là địa chỉ của một nút trong mạng de Bruijn thì hai nút có liên kết trực tiếp với nút đó sẽ là: $a_{k-2} \dots a_1 a_0 0$ và $a_{k-2} \dots a_1 a_0 1$.

Số cạnh của một nút là hằng số và độc lập với kích thước mạng. Độ rộng phân đôi của một mạng de Bruijn với 2^k nút là $2^k/k$. Tương tự mạng hoán vị-di chuyển, mạng de Bruijn có chứa các kết nối di chuyển (shuffle).



Hình 1.27. Một mạng de Bruijn với 8 bộ xử lý.

Đường kính của mạng de Bruijn với 2^k nút là k .

Một số máy tính song song dựa trên mạng de Bruijn như máy Triton/1™. Đây là một máy tính song song lai giữa hai mô hình SIMD và MIMD được thiết kế bởi trường Đại học Karlsruhe năm 1992.

1.3.10 Tổng kết về tổ chức các bộ vi xử lý

Trong 9 kiểu tổ chức ta đã xét ở trên được tóm tắt trong bảng sau:

| Mạng | Số nút | Đường kính | Độ rộng phân đôi | Số cạnh là hằng số? |
|---------------------------------|------------------|------------|------------------|---------------------|
| 1-D hình lưới | K | $k-1$ | 1 | đúng |
| 2-D hình lưới | k^2 | $2(k-1)$ | K | Đúng |
| 3-D hình lưới | k^3 | $3(k-1)$ | k^2 | Đúng |
| Cây nhị phân | $2^k - 1$ | $2(k-1)$ | 1 | Đúng |
| Siêu cây 4 chiều | $2^k(2^{k+1}-1)$ | $2k$ | 2^{k+1} | Đúng |
| Hình tháp | $(4k^2-1)/3$ | $2\log k$ | $2k$ | Đúng |
| Hình bướm | $(k+1)2^k$ | $2k$ | 2^k | Đúng |
| Siêu khối | 2^k | K | 2^{k-1} | Không |
| Hướng kết nối khối có chu trình | $k2^k$ | $2k$ | 2^{k-1} | Đúng |
| Hoán vị-di chuyển | 2^k | $2k-1$ | $>= 2^{k-1}/k$ | Đúng |

| | | | | |
|-----------|-------|---|---------|------|
| de Bruijn | 2^k | K | $2^k/k$ | Đúng |
|-----------|-------|---|---------|------|

1.4 Các hệ thống mảng bộ xử lý, đa bộ xử lý, và đa máy tính

1.4.1 Hệ thống mảng bộ vi xử lý (processor arrays)

a. Máy tính vector

Một máy tính vector là một máy tính mà tập lệnh của nó chứa các thao tác trên các vector có hướng hoặc vô hướng (scalar). Nói chung, có hai loại máy tính vector đó là:

Bộ xử lý vector kiểu pipeline (pipelined vector processor) có khả năng xử lý các dòng vector (lệnh) từ bộ nhớ đến CPU mà các bộ xử lý toán học có thể thao tác với chúng bằng cơ chế pipeline. Một số máy tính song song có kiểu tổ chức này là Cray-1™ và Cyber-205™.

Mảng bộ xử lý (processor array) là máy tính vector bao gồm một máy tính tuần tự được kết nối đến một tập các phần tử xử lý (processing elements) đồng bộ, riêng biệt có khả năng thực hiện cùng một thao tác trên các dữ liệu khác nhau một cách đồng thời. Máy tính tuần tự thường được gọi là front-end.

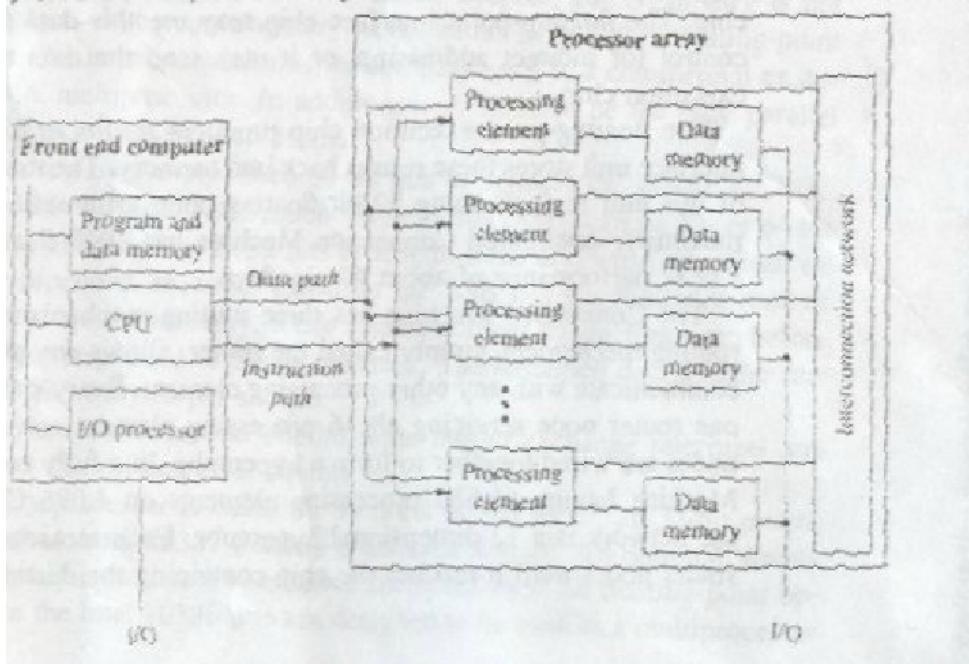
Front-end là một CPU đa năng chứa chương trình và các dữ liệu không thể xử lý song song. Đồng thời, front-end có thể thực hiện các phần tuần tự của chương trình (trong một chương trình, có thể có nhóm lệnh không thể thực hiện song song được, nhóm lệnh đó phải được thực hiện tuần tự).

Mỗi phần tử xử lý có một bộ nhớ cục bộ nhỏ mà nó có thể truy cập trực tiếp. Tập hợp lại, các bộ nhớ cục bộ của các phần tử xử lý lưu trữ vector dữ liệu sẽ được thao tác một cách song song. Khi máy tính front-end gặp một lệnh mà toán hạng của lệnh đó là một vector, nó sẽ phát ra một lệnh yêu cầu các phần tử xử lý thực hiện lệnh đó một cách song song. Mặc dù các phần tử xử lý thực hiện song song, các bộ xử lý này có thể được lập trình để lờ đi bất cứ lệnh không cần thiết nào. Chẳng hạn, trong trường hợp kích thước vector nhỏ hơn số phần tử xử lý, một vài phần tử xử lý sẽ không có dữ liệu để thực hiện lệnh do front-end yêu cầu, chúng sẽ bị idle và được lập trình để lờ đi lệnh này để tránh gây ra lỗi (bug) hoặc ngoại lệ (exception). Một trường hợp khác là khi thực hiện các câu lệnh điều kiện như *if...then* hoặc *if...then...else*, một số bộ vi xử lý sẽ bị che (masked) trong quá trình thực hiện do quá trình thực hiện trên máy tính mảng các phần tử xử lý là đồng bộ.

Thông thường, các luồng dữ liệu có thể di chuyển bằng một trong ba cách sau: đi từ front-end đến mảng các phần tử xử lý, giữa các phần tử xử lý trong mảng, và đi từ mảng các phần tử xử lý đến front-end. Các phần tử xử lý trao đổi dữ liệu với nhau thông qua một mạng kết nối (interconnection network).

cuu duong than cong . com

A realistic processor array model. Each processor has its own private memory, and processors can pass data only via a limited interconnection network.



Hình 1.28. Một máy tính mảng các bộ xử lý.

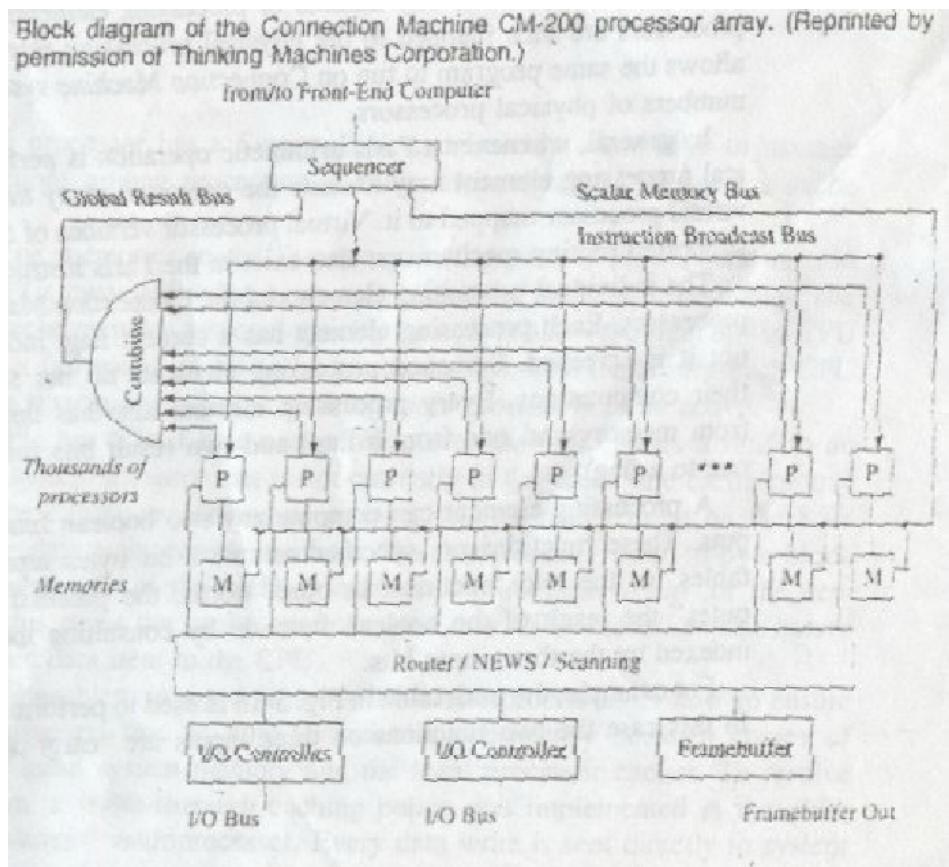
Chính vì vậy, cách tổ chức các bộ vi xử lý theo mô hình mạng lưới 2-D là phù hợp nhất. Một số cách tổ chức khác như: hình siêu khối, hình hướng khối có chu trình, và hoán vị-di chuyển cũng được giới thiệu cho máy tính với mảng các bộ xử lý.

Máy tính mảng các bộ vi xử lý có một cơ chế rất hiệu quả đối với front-end là phát tán (broadcast) các lệnh và dữ liệu đến các phần tử xử lý. Hơn nữa, máy tính với các mảng bộ vi xử lý cũng hỗ trợ front-end truy cập bộ nhớ cục bộ của bất kỳ phần tử xử lý nào rất hiệu quả. Phân phát các thao tác nhanh và truy cập bộ nhớ hiệu quả đóng vai trò quan trọng trong nhiều thuật toán song song.

b. Máy tính CM-200

Một ví dụ về máy tính với mảng các bộ vi xử lý là máy tính Connection Machine CM-200 được thiết kế và chế tạo bởi tập đoàn Thinking Machine.

Máy CM-200 bao gồm ba thành phần cơ bản: một máy tính front-end, một bộ xử lý song song (parallel processing unit) và một hệ thống vào/ra. Máy tính front-end thông thường là một máy trạm của công ty máy tính Sun. Nó lưu trữ dữ liệu tuần tự và thực hiện phần tuần tự của chương trình. Các dữ liệu song song được lưu trữ trong bộ xử lý song song. Máy tính front-end phân phát (broadcast) các lệnh song song tới bộ xử lý song song để thực hiện. Hệ thống vào/ra tốc độ cao cho phép dữ liệu được di chuyển giữa bộ xử lý song song và các thiết bị vào/ra như các khung vùng đệm (frame buffer) và các ổ đĩa song song (parallel disk devices).



[cuu duong than cong . com](http://cuuduongthancong.com)

Hình 1.29. Máy tính CM-200.

Máy tính front-end có thể trao đổi dữ liệu với các phần tử xử lý theo ba cách. Nó có thể phân phát (broadcast) một giá trị dữ liệu tới tất cả các phần tử xử lý. Thông qua các tính toán toàn cục, ta có thể tính được tổng, giá trị lớn nhất, OR toàn cục... của một giá trị từ mỗi phần tử xử lý. Bằng việc sử dụng bus bộ nhớ vô hướng (scalar memory), bộ xử lý front-end có thể đọc hoặc ghi các giá trị 32-bit được lưu trữ trên bất kỳ phần tử xử lý nào.

Bộ xử lý song song chứa khoảng 2048 đến 65536 phần tử xử lý, một bộ xử lý lệnh tuần tự và các mạng truyền thông giữa các bộ vi xử lý, bộ điều khiển vào / ra và/ hoặc các mô đun vùng đệm khung truyền.

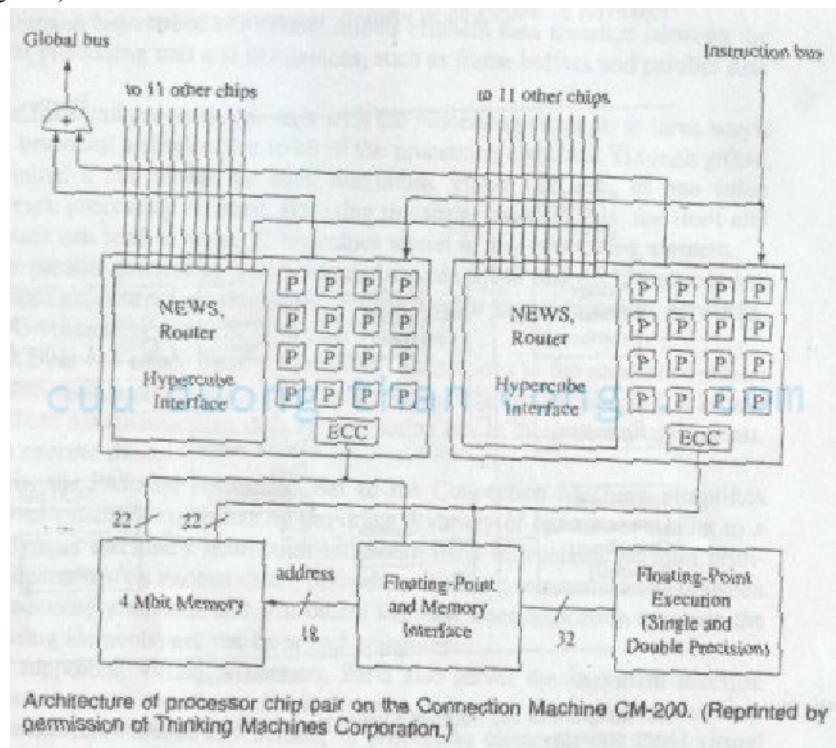
Front-end phân phát các lệnh xử lý song song tới bộ xử lý tuần tự, nó dịch từng lệnh và sinh ra một tập vi lệnh (nanoinstructions). Nó phân phát (broadcast) các vi lệnh này qua một bus truyền tải lệnh (instruction bus) tới các phần tử xử lý, sau đó các phần tử xử lý sẽ thực hiện vi lệnh.

PARIS (PARallel Instruction Set) là tập lệnh của máy tính CM-200, làm đơn giản hóa chương trình dịch bằng cách cung cấp các thao tác tương tự như một tập lệnh máy tính điển hình. Một số lệnh PARIS thực hiện các thao tác tính toán trên nhiều loại dữ liệu, những lệnh khác làm thuận tiện việc truyền thông giữa các phần tử xử lý và máy tính front-end.

Bằng việc hỗ trợ các máy ảo, PARIS cũng tạo ra sự ngăn cách an toàn (insulating) giữa người sử dụng với mảng bộ vi xử lý bên dưới. Một chương trình có thể giả sử sự tồn tại của một số lượng các phần tử xử lý bất kỳ, và những máy ảo này được ánh xạ lên các phần tử xử lý vật lý. Tính năng này cho phép cùng một chương trình chạy trên các hệ thống Connection Machine với các số lượng bộ vi xử lý khác nhau.

Nhìn chung, bất kỳ khi nào một thao tác (operation) PARIS được thực hiện, mỗi phần tử xử lý có thể thực hiện thao tác nhiều lần, mỗi lần sẽ được bộ xử lý ảo ánh xạ vào. Các phiên bản của bộ nhớ ảo của ba cơ chế hỗ trợ truyền thông cũng tồn tại trong tập lệnh PARIS.

Các phần tử xử lý của máy tính CM là các bộ xử lý bít tuần tự (bit-serial processors). Mỗi phần tử xử lý có một cờ trạng thái (context flag) chỉ ra rằng phần tử xử lý đó được che (screened) hay không. Các phần tử xử lý được che không lưu trữ các kết quả tính toán. Mỗi phần tử xử lý có 3 bít đầu vào (2 bít từ bộ nhớ và 1 bít từ cờ trạng thái) và 2 bít kết quả (1 bít đến bộ nhớ và 1 bít đến cờ trạng thái).



Hình 1.30. Kiến trúc bộ xử lý đôi của máy CM-200.

Một phần tử xử lý có thể tính toán hai hàm Boolean bất kỳ từ 3 đầu vào. Các hàm này được mô tả bằng 2 byte được biểu diễn trong bảng chân lý (Truth table). Nói cách khác, phần tử xử lý tính toán kết quả của hàm Boolean bằng việc tra cứu bảng chân lý từ ba bít đầu vào.

Chẳng hạn, bảng chân lý sau được sử dụng để thực hiện phép cộng bít tuần tự (bit-serial addition). Trong trường hợp này, hai hàm trên ba đầu vào sẽ được thực hiện (carry-out) và cộng (sum). Để cộng hai số nguyên k-bit được lưu trữ trong bộ nhớ cục bộ của nó, một phần tử xử lý tái cờ trạng thái (context flag) của bộ xử lý ảo (virtual processor) vào một thanh ghi cờ của phần cứng (vật lý). Tất cả các kết quả của phép toán trên ALU sẽ phản ánh vào trạng thái của cờ này.

Tiếp theo, bộ xử lý sẽ xóa cờ thứ 2 (cờ này có chức năng như một bít nhớ (carry bit). Sau đó, bộ xử lý lặp k lần qua một chu kỳ trong đó nó đọc bít nhớ và một bít của mỗi toán hạng (operand) và tính bít tổng và bít nhớ. Việc tính toán bắt đầu với bít có trọng số bé nhất (least significant bits) của các toán hạng và kết thúc với bít có trọng số cao nhất (most significant bits).

| INPUT BITS | | | OUTPUT BITS | |
|------------|--------|----|-------------|----|
| Bộ nhớ | Bộ nhớ | Cờ | Bộ nhớ | Cờ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Một chip VLSI đơn chứa 16 phần tử xử lý cùng với phần cứng hỗ trợ truyền thông (routing hardware). Mỗi cặp chip xử lý chia sẻ một nhóm chip bộ nhớ. Một chip làm giao diện dấu phẩy động (floating point interface chip) và một chip thực hiện các phép tính liên quan đến dấu phẩy động (floating-point execution chip) như hình (3-14, pg 66).

Các chip bộ nhớ cung cấp một đường dữ liệu 44 bit được chuyển đổi thành một đường dữ liệu 32 bit và một đường 12 bit mã hóa các mã khắc phục lỗi (error-correcting code). 32 bit dữ liệu có thể được trực tiếp gửi đến 32 phần tử xử lý bit tuần tự. Mỗi bit cho một phần tử xử lý. Tiếp đến, các bit dữ liệu 32 bit sẽ được gửi trực tiếp đến chip giao diện xử lý dấu phẩy động. Chip giao diện xử lý dấu phẩy động có thể sẽ sử dụng dữ liệu này để điều khiển cho các địa chỉ gián tiếp, hoặc gửi các dữ liệu này đến chip thực hiện các phép tính liên quan đến dấu phẩy động.

Chip thực hiện các phép tính liên quan đến dấu phẩy động trả lại kết quả dạng 32-bit. Bộ giao diện sẽ lưu lại những kết quả này vào bộ nhớ. Hiệu suất lớn nhất của bộ tính toán dấu phẩy động khi thực hiện các phép toán 32 bit là 20 megaflops. Máy CM có thể có đến 2048 chip thực hiện các phép tính liên quan đến dấu phẩy động và hiệu suất đạt tối đa là 40 Gigaflops.

Máy tính CM có ba cơ chế giao tiếp (routing mechanisms). Phổ biến và đơn giản nhất là dung bộ định tuyến (router). Cơ chế này cho phép bất kỳ phần tử xử lý nào cũng có thể giao tiếp được với nhau. Mỗi chip xử lý có một nút định tuyến có thể phục vụ 16 phần tử xử lý trên chip. Các nút định tuyến được nối dẫn (wired) với nhau để hình thành nên 1 siêu khối. Trong cấu hình tổng quát của máy tính CM bao gồm 65,536 phần tử xử lý được chứa trên 4096 (2^{12}) chip xử lý. Vì vậy các bộ xử lý của CM được tổ chức theo một mạng 12 siêu khối 12 chiều. Mỗi thông điệp đi qua các nút định tuyến cho đến khi nó đến được chip chứa bộ vi xử lý đích. Các nút định tuyến (router nodes) tự động gửi các thông điệp và thực hiện việc cân bằng tải động (dynamic load balancing). Chẳng hạn, đường đi của thông điệp có thể biến đổi, phụ thuộc vào những đường đi đang rảnh rỗi.

Mỗi nút định hướng có một ALU có khả năng thực hiện vài thao tác toán học. Bộ định tuyến kiểm tra xem các thông điệp có cùng đến 1 đích hay không?. Nếu điều đó xảy ra nó sẽ kết hợp các thông điệp dựa trên ngữ nghĩa của các lệnh sẽ được thực hiện. Chẳng hạn như có thể tính tổng, tính giá trị lớn nhất, tính phép tính logic của các giá trị hay đơn giản chỉ là bỏ đi tất cả các giá trị khác và chỉ giữ lại 1 giá trị duy nhất.

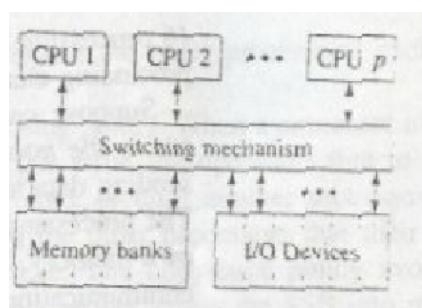
Có chế định tuyến thứ hai được gọi là NEWS grid. Một mạng hình lưới j chiều có thể được nhúng trong một mạng siêu khói k chiều nếu $j \leq k$. Vì vậy, một tập con của các dây nối hỗ trợ bộ định tuyến có thể hình thành một mạng lưới Cartesian, mà Thinking Machine gọi là NEWS grid. Nếu các giao tiếp giữa các phần tử xử lý là lân cận (neighbors) trong một NEWS grid của bất kỳ chiều nào nhỏ hơn hoặc bằng số chiều của siêu khói, thì tốc độ truyền tải thông điệp (message passing speed) sẽ cao hơn nhiều nếu các thông điệp đi qua bộ định tuyến. Hệ thống gửi các thông điệp như vậy sử dụng ba phương pháp truyền tải đặc biệt, hai trong ba phương pháp đó được thực hiện bởi phần cứng.

1.4.2 Máy tính đa bộ xử lý (Multiprocessors)

Máy tính nhiều bộ xử lý (multiple-CPU computers) có một số bộ vi xử lý có thể lập trình được, mỗi bộ vi xử lý có khả năng thực hiện chương trình của nó. Đa bộ vi xử lý có thể là các máy tính có nhiều bộ vi xử lý với một bộ nhớ chia sẻ. Có hai phương thức tổ chức máy tính đa bộ vi xử lý là: chia sẻ bộ nhớ tập trung (Uniform Memory Access- UMA) và chia sẻ bộ nhớ phân tán (Non-uniform Memory Access- NUMA).

a. Máy tính đa bộ xử lý chia sẻ bộ nhớ tập trung (UMA multiprocessors)

Cách thức bộ vi xử lý liên lạc đơn giản nhất là giả sử rằng tất cả các bộ vi xử lý làm việc qua cơ chế chuyển mạch tập trung tới bộ nhớ chia sẻ tập trung như hình (?).

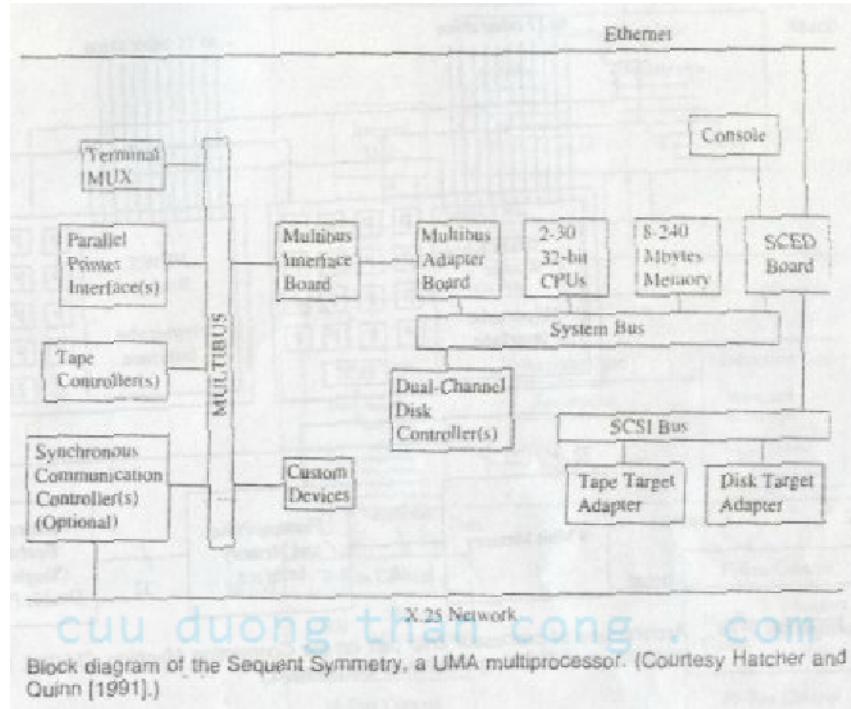


Hình 1.31. Máy tính đa bộ xử lý chia sẻ bộ nhớ tập trung truy cập bộ nhớ toàn cục và thiết bị vào/ra thông qua cơ chế chuyển mạch tập trung

Có nhiều cách thực thi bộ nhớ chuyển mạch này, bao gồm một bus chung tới bộ nhớ toàn cục, một bộ chuyển mạch chéo (crossbar switch) và một mạng chuyển mạch gói. Máy tính Encore Multimax™ và Sequent Symmetry S81™ là các ví dụ của máy tính đa bộ xử lý chia sẻ bộ nhớ tập trung thương mại.

Các hệ thống sử dụng một bus như Multimax hay Symmetry được giới hạn bằng kích thước do nhiều bộ vi xử lý chia sẻ một bus trước khi nó bị bão hòa (saturated). Trong trường hợp các hệ

thống sử dụng một bộ chuyển mạch chéo, chi phí của chuyển mạch sẽ trở thành mău số, điều này giới hạn số lượng bộ xử lý được kết nối. Máy tính đa bộ xử lý chia sẻ bộ nhớ tập trung dựa trên các mạng chuyển mạch có thể có một số lượng lớn các bộ xử lý. Mặc dù chưa có máy tính thương mại nào có kiến trúc như vậy xuất hiện, một máy tính thử nghiệm dựa trên máy tính đa bộ xử lý chia sẻ bộ nhớ tập trung có tên là NYU Ultracomputer dựa trên mạng chuyển mạch omega (một biến thể của mạng hoán vị-di chuyển). Chi phí cho một hệ thống với p bộ xử lý dựa trên mạng omega là O(plogp) thấp hơn so với chi phí O(p²) cho hệ thống dựa trên chuyển mạch chéo.

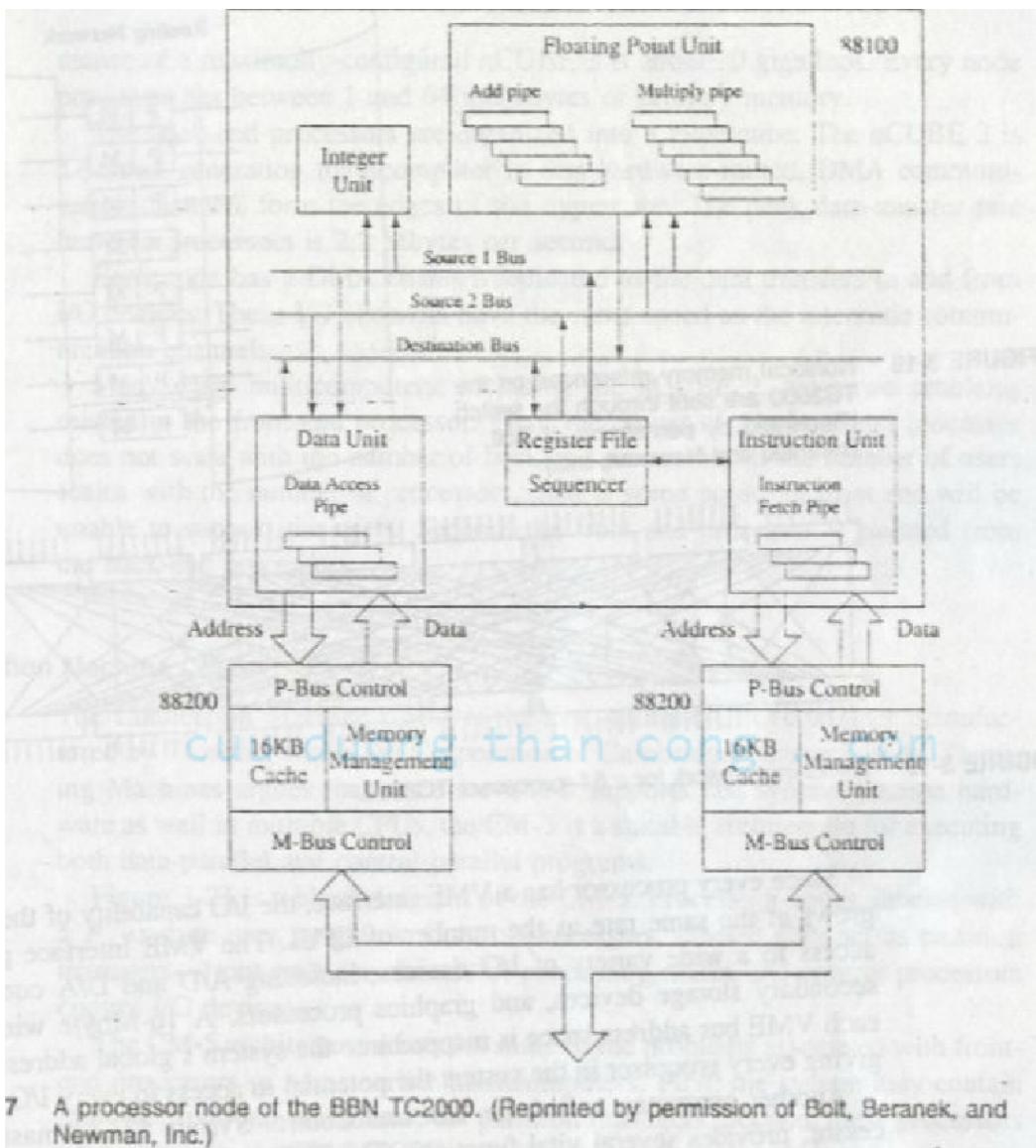


Hình 1.32. Một máy tính đa bộ xử lý UMA.

b. Máy tính đa bộ xử lý chia sẻ bộ nhớ phân tán (NUMA multiprocessors)

Máy tính đa bộ xử lý chia sẻ bộ nhớ phân tán giống máy tính đa bộ xử lý chia sẻ bộ nhớ tập trung ở điểm là chúng cùng chia sẻ không gian địa chỉ nhô. Nhưng chúng khác nhau bởi điểm cơ bản: Máy tính đa bộ xử lý chia sẻ bộ nhớ phân tán sử dụng bộ nhớ phân tán. Mỗi bộ xử lý có riêng một bộ nhớ cục bộ và không gian địa chỉ nhô chia sẻ là sự kết hợp của các bộ nhớ cục bộ này. Thời gian cần thiết để truy cập từng bộ nhớ cục bộ trên máy tính đa bộ xử lý chia sẻ bộ nhớ phân tán phụ thuộc vào vị trí cục bộ đối với bộ xử lý.

Một ví dụ về Máy tính đa bộ xử lý chia sẻ bộ nhớ phân tán là máy tính TC2000 được thiết kế và sản xuất bởi công ty BBN Systems and Technologies ở thành phố Cambridge, bang Massachusetts- Hoa Kỳ. Máy tính TC 2000 có 128 nút xử lý. Mỗi nút xử lý gồm một CPU Motorola 88100, ba chip Motorola 88200 cung cấp 2 loại bộ nhớ cache là bộ nhớ cache lệnh và cache dữ liệu với kích thước từ 4 đến 16 MB, và giao diện chuyển mạch dựa trên mạng hình bướm và bus VME (hình (?)). Các giao dịch dựa trên bus kết nối các hệ thống bộ xử lý con.



Hình 1.33. Một nút xử lý của máy tính BBN TC2000

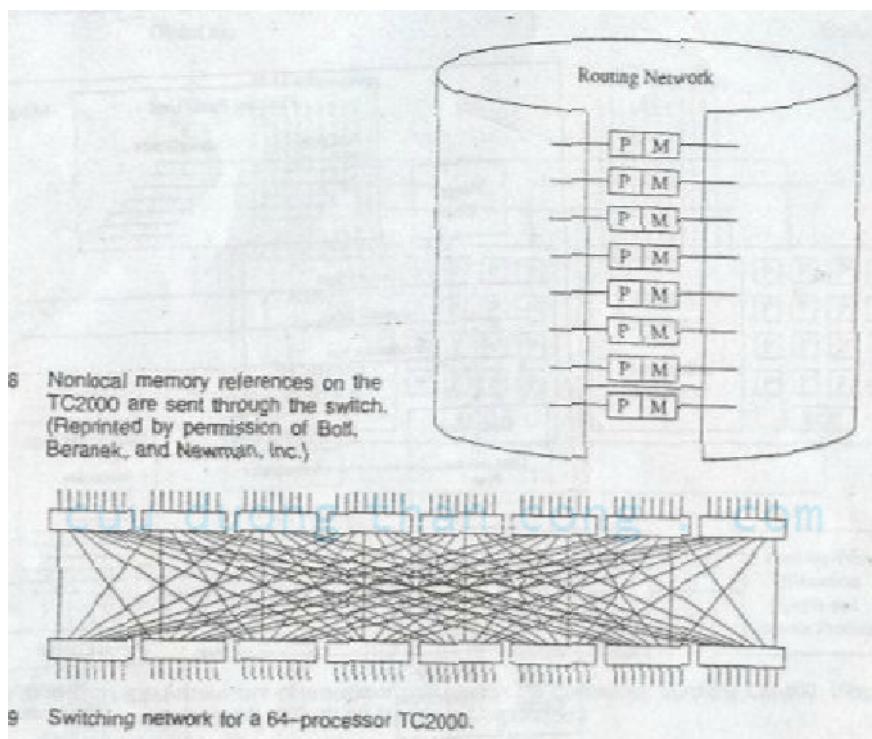
Hiệu suất lớn nhất của một bộ xử lý đơn là 20 megaflop. Một máy tính TC2000 với 128 bộ xử lý có thể đạt tới hiệu xuất khoảng 2.5 gigaflop.

Phần cứng trên mỗi nút xử lý biến đổi địa chỉ áô 32 bit thành địa chỉ vật lý 34 bit. Địa chỉ vật lý có thể trong bộ nhớ cache, cùng bộ nhớ cục bộ trên bộ xử lý hoặc có thể là bộ nhớ cục bộ trên một xử lý khác. Tốc độ truy nhập bộ nhớ cache là nhanh nhất. Dữ liệu cục bộ (private data) và dữ liệu chia sẻ chỉ đọc (read-only shared data), chẳng hạn như mã nguồn chương trình, có thể được

lưu trong bộ nhớ cache. Dữ liệu từ bộ nhớ vật lý của bộ xử lý có thể được đọc lên bus. Dữ liệu từ bộ nhớ của bộ xử lý khác có thể được lấy bằng cách gửi một yêu cầu qua mạng chuyển mạch hình bướm.

Máy TC 2000 không phải duy trì sự nhất quán dữ liệu trên các bộ nhớ cache của các bộ xử lý. Đó là lý do tại sao mà các hệ điều hành chỉ đưa một số lượng hạn chế dữ liệu vào bộ nhớ cache.

Hình (3-18 và 3-19) minh họa một chuyển mạch kiểu hình bướm. Mỗi nút của chuyển mạch là một phần tử chuyển mạch VLSI với 8 đầu vào và 8 đầu ra. Vì vậy, số phần tử chuyển mạch trong một máy tính với p bộ xử lý là $p \times 8^2$. Đường dữ liệu qua các chuyển mạch được lưu trong các gói; và các bit địa chỉ định tuyến từ nơi phát đến nơi nhận.



Hình 1.34. Chuyển mạch trên máy TC-2000.

Do mỗi bộ xử lý có một giao diện VME, dung tích I/O của hệ thống được mở rộng theo số lượng CPU. Các giao diện VME cung cấp khả năng truy nhập tới các thiết bị vào/ra khác nhau, bao gồm cả các bộ biến đổi A/D, D/A, bộ nhớ ngoài, và các bộ xử lý đồ họa. Một cửa sổ 16 MB trong mỗi không gian địa chỉ bus VME được ánh xạ vào địa chỉ toàn cục của hệ thống. Nên mỗi bộ xử lý đều có tiềm năng truy cập vào mọi thiết bị vào/ra.

Một máy tính khác, hệ thống xử lý chủ TCS (Test and Control System) cung cấp các chức năng quan trọng. Bộ xử lý TCS chủ cung cấp một giao diện cho TCS. Nó cũng tải các chương trình khởi động vào bộ nhớ của bộ xử lý tại thời điểm hệ thống khởi động, kết hợp với các hoạt động của bộ xử lý thợ (slaver processor), và nhận các thông điệp được gửi bởi các bộ xử lý thợ. Bộ xử lý chủ TCS được kết nối tới bộ xử lý thợ qua một bus chẩn đoán (diagnostic bus), nó được định vị trong chip của bộ xử lý và trong mọi module chuyển mạch, và nó tiếp tục theo dõi các lỗi của hệ thống.

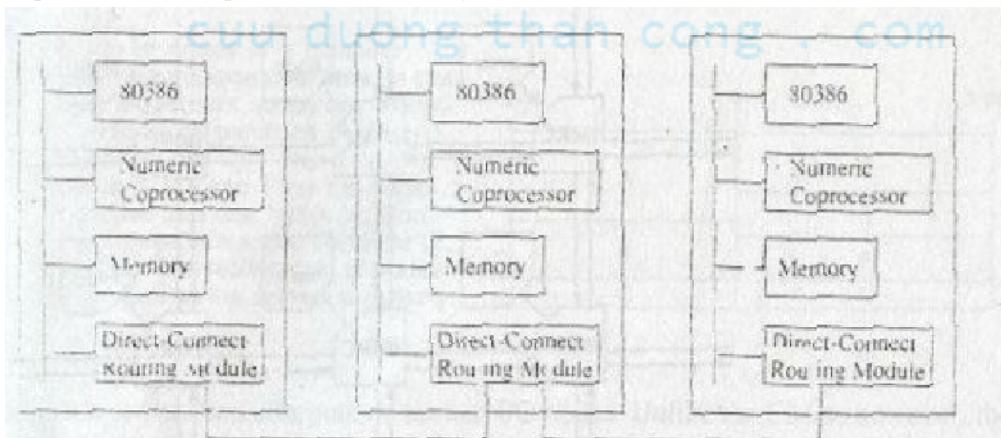
1.4.3 Hệ thống đa máy tính (Multicomputers)

Một kiến trúc khác của đa bộ xử lý là hệ thống đa máy tính. Hệ thống đa máy tính không có bộ nhớ chia sẻ. Mỗi bộ xử lý có riêng một bộ nhớ cục bộ (private memory) và tương tác với nhau qua trao đổi thông điệp (message passing). Hiện nay đã có nhiều hệ thống đa máy tính thương mại như Intel ‘s Paragon XP/S, Meiko Computing Surface™, nCUBE ‘s nCUBE, Parsytec ‘s SuperCluster™, và Thinking Machine ‘s CM-5.

Một đặc điểm nổi trội của hệ thống đa máy tính mới là cách thức bộ vi xử lý giao tiếp với nhau. Các thế hệ đa máy tính đầu tiên như Intel iPSC™, nCUBE/10™ và các hệ thống dựa trên T800 Transputer™, được đặc trưng bởi phần mềm hỗ trợ lưu trữ và chuyển tiếp thông điệp (Fig. 3-20).

Để gửi một thông điệp từ một bộ xử lý đến một bộ xử lý không liền kề (nonadjacent processor), các bộ xử lý trung gian nằm trên đường đi của thông điệp sẽ có nhiệm vụ lưu trữ chuyển thông điệp đó đến bộ xử lý kế tiếp. Thậm chí nếu sự truyền tải dữ liệu được hoàn thành qua các kênh DMA, CPU sẽ bị treo vào thời điểm truyền tải dữ liệu bằng DMA.

Ngược lại, hệ đa máy tính thế hệ thứ hai như Intel iPSC/2™, Intel iPSC/850™ và nCUBE 2 có định tuyến chuyển mạch thông điệp (circuit-switched message routing). Chẳng hạn, Intel các nút trong máy tính iPSC/2™ và Intel iPSC/850™ có card logic định tuyến được gọi là mô đun kết nối trực tiếp (Direct-Connect Module-DCM). Các mô đun kết nối trực tiếp thiết lập nên một mạch điện (circuit) kết nối từ nút nguồn đến nút đích. Mỗi khi một mạch được thiết lập, các luồng thông điệp trong các đường dẫn xen kẽ (pipelined) sẽ đi từ nút nguồn đến nút đích và thông điệp không được lưu trong các nút trung gian. Một thông điệp được truyền từ nút này đến nút khác (không ở cạnh nhau) sẽ không yêu cầu treo CPU mà chỉ cần kích hoạt các mô đun kết nối trực tiếp để điều khiển quá trình truyền thông.



The Direct-Connect Modules on the Intel iPSC/2 multicomputer support circuit-switched message routing which allows a connection to be established between the sending and receiving processors and keeps the CPUs of intermediate nodes from being interrupted when a message passes through. (Courtesy Hatcher and Quinn [1991].)

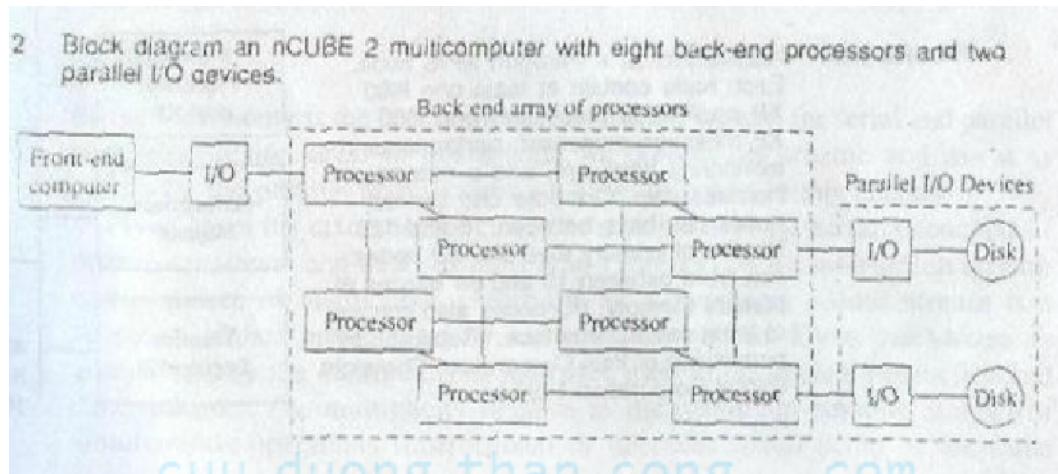
Hình 1.35. Mô đun kết nối trực tiếp trên đa máy tính iPSC/2.

Trong mô hình lưu trữ và chuyển tiếp (store-and-forward), thời gian cần thiết để gửi một thông điệp từ bộ xử lý này đến bộ xử lý khác sẽ tăng tuyến tính đối với “bước nhảy” (hop) của thông điệp để tới được đích của nó. Ngược lại với mô hình lưu trữ và chuyển tiếp, thời gian cần thiết để

gửi một thông điệp từ bộ xử lý này đến bộ xử lý khác trong một chuyên mạch sẽ không phụ thuộc vào khoảng cách giữa các bộ vi xử lý.

a. Máy tính nCUBE 2

Máy tính nCUBE 2 được chế tạo bởi công ty nCUBE tại thành phố Foster, Hoa Kỳ. Máy tính nCUBE 2 có ba thành phần chính: một máy tính front-end, một mảng các bộ xử lý back-end và các thiết bị vào/ra song song (hình 3-22). Máy tính nCUBE 2 tương tự như máy tính mảng các bộ vi xử lý. Tuy nhiên, hai kiến trúc này có một điểm khác nhau cơ bản là: máy tính front-end của mảng các bộ xử lý điều khiển các hoạt động của các phần tử xử lý. Nó thực hiện các phép tính đơn giản một cách đồng bộ. Ngược lại, các bộ xử lý trong phần back-end của máy tính nCUBE 2 là các CPU với đầy đủ tính năng (full-fledged CPUs) và những CPU này thực hiện các dòng lệnh của chúng một cách song song.



Hình 1.36. Sơ đồ khối của đa máy tính nCUBE2.

Máy tính nCUBE 2 có tới 8192 nút xử lý. Mỗi nút xử lý có hiệu xuất tối đa 2.5 megaflop. Vì vậy, theo lý thuyết thì hiệu xuất tối đa của máy tính nCUBE 2 là 20 gigaflops. Mỗi nút xử lý có một bộ nhớ chính với kích thước từ 1 đến 64 MB.

Các bộ xử lý back-end được tổ chức trong một siêu khối. Máy tính nCUBE 2 là thế hệ thứ hai của hệ thống đa máy tính nên kỹ thuật truyền thông giữa các bộ xử lý là định tuyến cứng (hardware-routed). Các kênh truyền thông DMA hình thành nên các cạnh của siêu khối. Tốc độ truyền dữ liệu lớn nhất giữa các bộ xử lý là 2.2 MB/giây.

Mỗi nút có một kênh DMA dùng cho việc truyền tải dữ liệu từ/đến các thiết bị vào/ra. Các kênh vào/ra có cùng tốc độ truyền với các kênh truyền của các nút trong mạng.

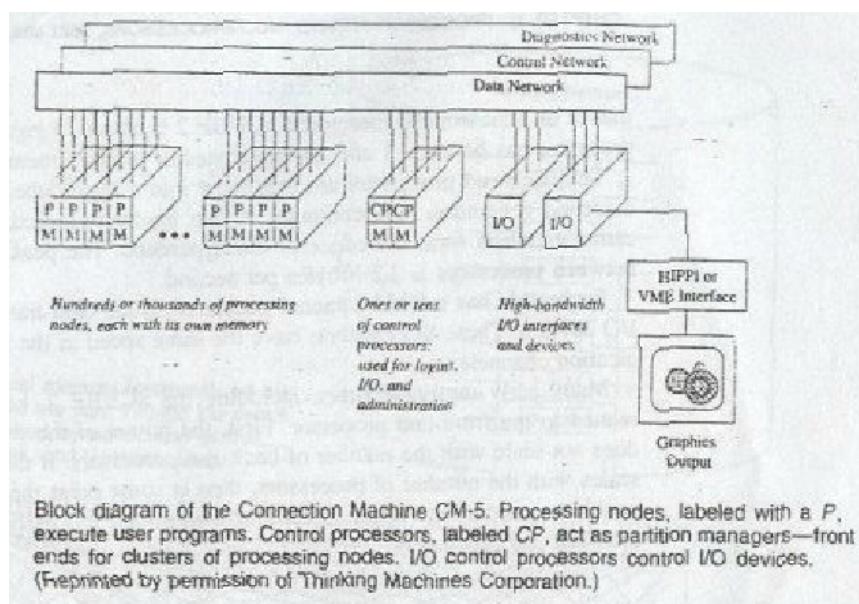
Các hệ đa máy tính đầu tiên có hai vấn đề liên quan đến bộ xử lý front-end. Thứ nhất, công suất (power) của bộ xử lý front-end không tỉ lệ (scale) với các bộ xử lý back-end. Nếu số người dùng tỉ lệ với số bộ xử lý, thì tại một số thời điểm, front-end sẽ không thể hỗ trợ người sử dụng. Thứ hai, bộ xử lý front-end cách biệt với các bộ xử lý back-end.

b. Máy tính Connection Machine CM-5

Máy tính CM-5 là hệ thống đa máy tính đầu tiên được chế tạo bởi công ty Thinking Machine tại Cambridge, bang Massachusetts - Hoa Kỳ. Thinking Machine cho rằng vì máy tính CM-5 hỗ trợ

phần cứng đồng bộ nhanh và đa bộ xử lý. Máy tính CM-5 là kiến trúc phù hợp cho việc thực hiện cả dữ liệu song song và chương trình điều khiển song song.

Hình (3-23) là một sơ đồ khối của CM-5. Các nút xử lý được gán nhãn P, thực hiện chương trình người dùng. Các bộ xử lý điều khiển (Control Processors) có nhãn là CP thực hiện như bộ quản lý phân hoạch (partition managers) front-ends cho các cụm của các nút xử lý. Các bộ xử lý điều khiển vào/ra làm nhiệm vụ điều khiển các thiết bị vào/ra.



Hình 1.37. Sơ đồ khối của máy tính CM-5

Kiến trúc của máy tính CM-5 cũng khắc phục được một số vấn đề gắn với bộ xử lý front-end của các hệ thống đa máy tính trước đây. Trước hết, hệ thống có thể có nhiều bộ xử lý front-end và các bộ quản lý phân hoạch. Thứ hai, các bộ xử lý này dựa trên kiến trúc SPARC, giống với các bộ xử lý back-end. Thứ ba, các bộ phân hoạch là một phần của mạng định tuyến có cùng dữ liệu với các bộ xử lý back-end.

Tổ chức của bộ xử lý trong máy tính CM-5 là một 4 siêu cây (hình 3-3). Các bộ xử lý là các lá của cây và các nút bên trong định tuyến dữ liệu giữa các lá. Dải thông (bandwidth) tối thiểu của mạng là 5MB/giây.

Mỗi nút CM-5 bao gồm một bộ xử lý SPARC, phần cứng giao diện mạng, một bộ nhớ chính 32MB và 4 bộ xử lý vector kiểu pipeline, mỗi bộ xử lý vector có thể xử lý được 32 triệu phép tính dấu phảy động trong một giây. Máy tính CM-5 với cấu hình đầy đủ (fully-configured) có tới 16,384 nút, 512 GB bộ nhớ chính và hiệu xuất lớn nhất có thể đạt tới 2 Teraflops (về mặt lý thuyết).

1.5 Kết chương

Trong chương này, ta đã thảo luận hai vấn đề chính: kiến trúc và tổ chức các bộ vi xử lý trong máy tính song song. Phần đầu của chương là phần tổng quan về tính toán song song (parallel

computing). Bắt nguồn từ các yêu cầu khối lượng tính toán lớn trong hầu hết các ứng dụng trong đời sống: khoa học, thương mại... là các bài toán đòi hỏi khả năng tính toán vượt xa so với hệ thống tính toán hiện tại. Trong khi đó, tốc độ nhanh nhất của máy tính với một bộ xử lý đơn ngày càng tăng nhưng còn kém tốc độ của máy tính với nhiều bộ xử lý. Các thuật ngữ cơ bản và lịch sử phát triển của máy tính song song cũng được đề cập trong phần đầu của chương.

Theo Flynn, tất cả các máy tính được chia thành 4 loại dựa trên xử lý song song hoặc tuần tự của dòng lệnh và dòng dữ liệu. Trong đó, các máy tính tuần tự thuộc loại máy tính một dòng lệnh, một dòng dữ liệu (SISD); bao gồm cả các máy tính tích hợp nhiều bộ chức năng hay xử lý nhưng chỉ thực hiện một dòng lệnh được gọi là systolic. Các máy tính với một mảng các bộ xử lý thuộc loại máy tính một dòng lệnh, nhiều dòng dữ liệu (SIMD). Các máy tính systolic với mảng các bộ vi xử lý thuộc loại nhiều dòng lệnh, một dòng dữ liệu (MISD). Cuối cùng, là các máy tính nhiều CPU (multiple-CPU machines) như máy tính đa bộ xử lý (multiprocessors), đa máy tính (multicomputers) thuộc loại máy tính nhiều dòng lệnh, nhiều dòng dữ liệu (MIMD).

Tiếp đến, có 9 cách thức tổ chức các bộ vi xử lý trong các máy tính song song được thảo luận. Các mô hình tổ chức đó là: mạng hình lưới, mạng hình cây nhị phân, mạng hình siêu cây, mạng hình tháp, mạng hình cánh bướm, mạng hình siêu khối, mạng các chu trình hướng kết nối khối, và mạng de Bruijn. Đồng thời, ta cũng đánh giá cách thức tổ chức của từng mạng theo 3 tiêu chuẩn: đường kính, độ rộng phân đôi và số cạnh trên một nút. Các tiêu chuẩn này giúp chúng ta lựa chọn cách thức tổ chức các bộ vi xử lý để thực hiện các giải thuật song song một cách hiệu quả và phù hợp nhất theo quan điểm của nhà thiết kế và chế tạo máy tính song song một cách thực tế.

Máy tính mảng các bộ vi xử lý được thực thi từ mô hình kiến trúc SIMD, trong đó xử lý song song đạt được thông qua sự áp dụng một thao tác đơn trên một tập dữ liệu. Sẽ là không thực tế nếu ta xây dựng một máy tính mảng các bộ xử lý trong đó các bộ xử lý chia sẻ một bộ nhớ toàn cục. Trong thực tế, mỗi bộ xử lý trong máy tính này có một bộ nhớ cục bộ. Cách thức tổ chức các bộ vi xử lý quyết định cách thức trao đổi dữ liệu giữa các bộ xử lý. Máy tính CM-200 là một máy tính với các mảng bộ vi xử lý theo mô hình SIMD đã được thương mại hóa và ứng dụng rộng rãi trên thực tế.

Các máy tính MIMD là các máy tính đa năng của tính toán song song. Chúng bao gồm nhiều CPU mà mỗi CPU thực hiện một dòng lệnh độc lập và không đồng bộ. Các máy tính MIMD có thể được phân loại bằng cách truy cập bộ nhớ. Máy tính đa bộ xử lý UMA có một không gian địa chỉ chia sẻ và khoảng cách từ một CPU tới một ô nhớ bất kỳ là hằng số. Máy tính đa bộ xử lý NUMA cũng có bộ nhớ chia sẻ. Nhưng khoảng cách mỗi ô nhớ đến các bộ xử lý là khác nhau. Hệ thống đa máy tính không có bộ nhớ chia sẻ mà mỗi CPU có một bộ nhớ cục bộ riêng và các bộ xử lý giao tiếp với nhau thông qua cơ chế truyền thông điệp (message passing).

1.6 Câu hỏi và bài tập

1.6.1 Câu hỏi

- Trình bày về xu hướng thiết kế máy tính hiện đại ?

2. Trình bày phân loại máy tính theo Flynn?. Theo cách phân loại này thì một máy tính cá nhân với chip Dual-core (hai nhân) hiện nay thuộc loại nào?
3. Trình bày về máy tính SIMD ?. Hãy so sánh ưu và nhược điểm của máy tính SIMD với bộ nhớ phân tán và bộ nhớ chia sẻ?.
4. Hãy tìm một ứng dụng thực tế có thể thực hiện trên máy SIMD với bộ nhớ chia sẻ.
5. Trình bày về máy tính MIMD ?. Máy tính MIMD với bộ nhớ chia sẻ và phân tán.
6. Tìm một ứng dụng cụ thể trên máy tính MISD?.
7. Đánh giá các cách thức tổ chức bộ vi xử lý thông qua các tiêu chuẩn gì ?. Ý nghĩa của từng tiêu chuẩn.
8. Trình bày cách thức tổ chức các bộ vi xử lý theo mạng hình lưới. Ưu và nhược điểm của cách thức tổ chức này?.
9. Trình bày cách thức tổ chức các bộ vi xử lý theo mạng hình cây. Ưu và nhược điểm của cách thức tổ chức này?.
10. Trình bày cách thức tổ chức các bộ vi xử lý theo mạng hình siêu cây. Ưu và nhược điểm của cách thức tổ chức này?.
CuuDuongThanCong . com
11. Trình bày cách thức tổ chức các bộ vi xử lý theo mạng hình tháp. Ưu và nhược điểm của cách thức tổ chức này?.
12. Trình bày cách thức tổ chức các bộ vi xử lý theo mạng hình siêu khối. Ưu và nhược điểm của cách thức tổ chức này?.
13. Trình bày cách thức tổ chức các bộ vi xử lý theo mạng hình bướm. Ưu và nhược điểm của cách thức tổ chức này?.
14. Trình bày cách thức tổ chức các bộ vi xử lý theo mạng chu trình hướng kết nối khối. Ưu và nhược điểm của cách thức tổ chức này?.
15. Trình bày cách thức tổ chức các bộ vi xử lý theo mạng di chuyển hoán vị. Ưu và nhược điểm của cách thức tổ chức này?.
CuuDuongThanCong . com
16. Trình bày về máy tính mảng các bộ vi xử lý?.
17. Trình bày về máy tính đa bộ xử lý UMA Sequent Symmetry?

18. Trình bày về máy tính đa bộ vi xử lý NUMA TC 2000?

19. Trình bày về hệ thống đa máy tính?

20. So sánh kiến trúc của máy tính nCUBE2 và CM-5?

1.6.2 Bài tập

1. Người ta cần tính tổng của hai véc tơ u và v trên một máy tính SIMD với 100 bộ vi xử lý; Hãy tính hiệu suất của máy tính SIMD khi thực hiện chương trình này biết rằng:

- Kích thước (số phần tử) của mỗi véc tơ u hoặc v là 350.
- Mỗi lệnh cộng thực hiện trên một bộ xử lý mất 10^{-9} giây.
- Chương trình tính tổng trên đã được viết tối ưu cho máy SIMD

2. Một chương trình tính tổng các phần tử của một véc tơ v trên một máy tính SIMD với 1000 bộ vi xử lý; Hãy tính hiệu suất của máy tính SIMD khi thực hiện chương trình này biết rằng:

- Kích thước (số phần tử) của véc tơ v là 2000.
- Mỗi lệnh cộng thực hiện trên một bộ xử lý mất 10^{-8} giây.
- Chương trình trên đã được viết tối ưu cho máy SIMD

3. Trình bày về cách thức tổ chức 1024 bộ vi xử lý theo mạng hình bướm? Bộ vi xử lý (7,9) sẽ được kết nối với hai bộ vi xử lý nào ở hàng thứ 6.?

4. Trình bày sự khác nhau giữa một mạng siêu khối 4 chiều và một mạng chu trình hướng kết nối khối cấp 4.

5. Hãy so sánh cách thức tổ chức bộ vi xử lý theo hai mạng: hình lưới kích thước 64 và hình tháp kích thước 64.

6. Trình bày về cách thức tổ chức 1024 bộ vi xử lý theo mạng hình siêu khối? Bộ vi xử lý 0011101010 sẽ được kết nối với các bộ vi xử lý nào trong mạng?

7. Trình bày về cách thức tổ chức 64 bộ vi xử lý theo mạng chu trình hướng kết nối khối. Bộ vi xử lý $P_{1,2,3}$ sẽ được kết nối với các bộ vi xử lý nào trong mạng?

8. Một chương trình tính tích vô hướng của hai véc tơ u và v trên một máy tính SIMD với 1000 bộ vi xử lý; Hãy tính hiệu suất của máy tính SIMD khi thực hiện chương trình này biết rằng:

- Kích thước (số phần tử) của mỗi véc tơ u hoặc v là 1000.
- Mỗi lệnh cộng hoặc nhân thực hiện trên một bộ xử lý mất 10^{-10} giây.
- Chương trình trên đã được viết tối ưu cho máy SIMD

9. Cho một mạng hoán vị- di chuyển, chứng minh rằng nếu một kết nối di chuyển (shuffle) nối nút i với nút j thì j là kết quả của phép quay trái một bít trong biểu diễn nhị phân của i.

10. Tính hiệu suất tốt nhất của một chương trình tính tổng $S=1+2+\dots+100$ trên một máy tính CM-200 với 50 bộ vi xử lý. Biết rằng, tại một thời điểm, mỗi bộ vi xử lý có thể thực hiện được một phép cộng cho hai số và thời gian thực hiện một phép cộng trên một bộ xử lý là 10^{-8} .

CHƯƠNG 2 : CÁC THUẬT TOÁN SONG SONG

Chương này trình bày một số vấn đề sau:

- *Mô hình PRAM* là mô hình làm đơn giản việc thiết kế các giải thuật song song. Trong mô hình PRAM độ phức tạp về truyền thông được bỏ qua và số lượng bộ vi xử lý tham gia thực hiện các thuật toán là vô hạn. Cùng với mô hình PRAM, ta tìm hiểu một số thuật toán song song đơn giản thiết kế cho máy PRAM.
- *Các thuật toán song song nhân hai ma trận*; trong phần này ta trình bày 3 thuật toán song song để giải quyết bài toán nhân hai ma trận. Thuật toán thứ nhất là song song hóa từ thuật toán nhân hai ma trận tuần tự. Thuật toán thứ hai là nhân hai ma trận theo khối, và cuối cùng là thuật toán nhân hai ma trận trên hệ thống nhiều máy tính tổ chức theo hình siêu khối.
- *Thuật toán sắp xếp song song*: Ba thuật toán song song giải quyết bài toán sắp xếp được thảo luận trong phần này. Thuật toán sắp xếp song song hoán chuyển chẵn lẻ trên máy tính có các bộ xử lý được tổ chức theo mạng hình lưới một chiều; Thuật toán sắp xếp song song trộn Bitonic và thuật toán song song dựa trên Quicksort.
- *Thuật toán tìm kiếm song song trên danh bạ*: được song song hóa từ giải thuật tuần tự Ellis. Đồng thời, ta cũng tìm hiểu về một giải thuật tìm kiếm song song rất hiệu quả với các tiến trình thực hiện các thao tác trên cây tìm kiếm nhị phân được chạy song song do Manber và Ladner đề xuất.
- *Thuật toán song song trên đồ thị*: trình bày về hai giải thuật được song song hóa từ giải thuật tuần tự của Moore để giải bài toán tìm đường đi ngắn nhất trên một đồ thị có trọng số; và song song hóa từ giải thuật Kruskal để giải bài toán tìm cây khung nhỏ nhất trên đồ thị vô hướng, có trọng số.

Phần cuối cùng của chương là phần kết chương và bài tập dành cho người học.

2.1 Mô hình PRAM

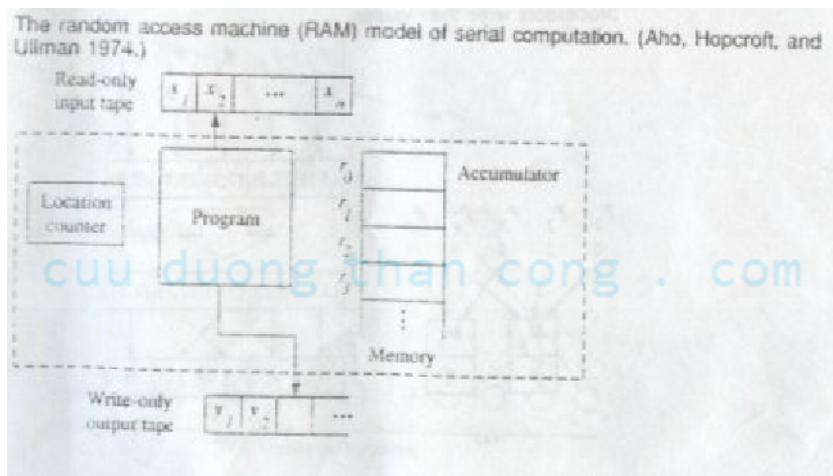
Phần này trình bày về mô hình máy truy cập ngẫu nhiên song song (parallel random access machine) PRAM trong tính toán song song. Trong khi các thuật toán tuần tự dựa trên mô hình máy tính von Neumann hay Turing thì mô hình PRAM cho phép các nhà thiết kế các thuật toán song song xem năng lực xử lý (processing power) như nguồn tài nguyên vô hạn như các nhà lập trình máy tính xem bộ nhớ ảo là tài nguyên không hạn chế. Mô hình PRAM làm đơn giản quá trình thiết kế thuật toán song song nhưng không thực tế vì ta bỏ qua độ phức tạp truyền thông giữa các bộ xử lý.

2.1.1 Mô hình xử lý tuần tự

Máy tính truy cập ngẫu nhiên (random access machine) RAM bao gồm một bộ nhớ, một băng từ chỉ đọc chứa đầu vào, một băng từ (tape) chỉ ghi chứa đầu ra, và một chương trình (hình 2-1). Chương trình không được lưu trong bộ nhớ và không thể thay đổi. Băng đầu vào chứa một dãy các số nguyên. Tại mỗi thời điểm một giá trị đầu vào được đọc, đầu đọc sẽ dịch chuyển một ô vuông. Tương tự, khi một giá trị đầu ra được ghi vào băng thì đầu ghi được dịch chuyển một lần. Bộ nhớ là một dãy vô hạn các thanh ghi kí hiệu là r_0, r_1, \dots mỗi thanh ghi có thể chứa một số nguyên.

Các lệnh có thể tương tự như các lệnh trên một máy tính thực. Vì vậy, một máy tính RAM có thể có các lệnh load, store, write, add, subtract, multiply, divide, jump ...

Độ phức tạp về thời gian trong trường hợp xấu nhất (worst-case time complexity) của một chương trình RAM là một hàm $f(n)$ chỉ thời gian lớn nhất thực hiện bởi chương trình cho mọi đầu vào có kích thước n . Độ phức tạp về thời gian kỳ vọng trung bình (expected time complexity) của chương trình là thời gian trung bình thực hiện của chương trình đối với mọi đầu vào kích thước n . Các khái niệm tương tự đối với độ phức tạp về không gian trong trường hợp xấu nhất và trung bình.



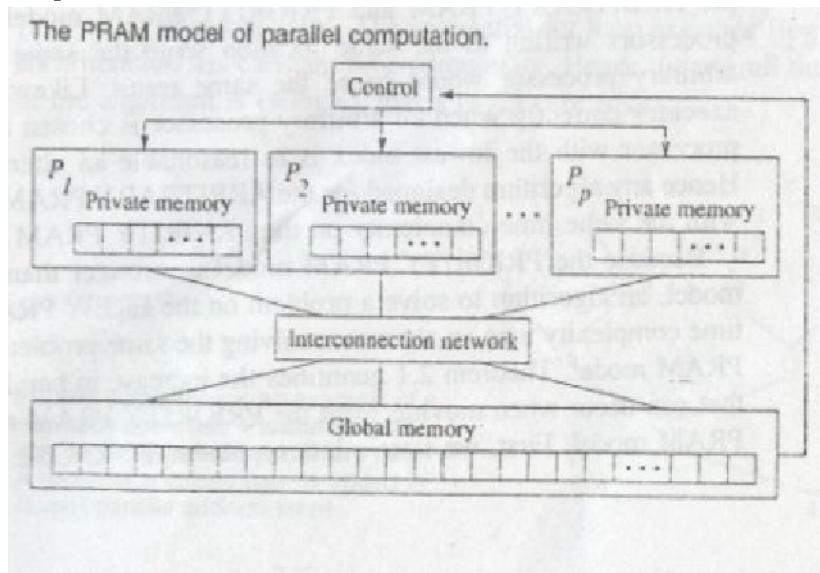
Hình 2.1. Mô hình xử lý tuần tự RAM

Có hai cách để đo độ phức tạp về thời gian và không gian dựa trên mô hình RAM. *Dánh giá theo tiêu chuẩn đồng nhất chi phí* (uniform cost criterion) phát biểu rằng mỗi lệnh của RAM đòi hỏi tiêu tốn một đơn vị thời gian để thực hiện và mỗi thanh ghi tương đương với một đơn vị không gian (bộ nhớ). *Dánh giá theo tiêu chuẩn chi phí lô ga* (logarithmic cost criterion) thì cho rằng bộ nhớ thực tế bị giới hạn về khả năng lưu trữ (kích thước). Dánh giá theo tiêu chuẩn đồng nhất chi phí là phù hợp nếu các giá trị được thao tác bởi chương trình có thể lưu trữ trong một từ máy.

2.1.2 Mô hình tính toán song song PRAM

Một mô hình PRAM gồm một bộ điều khiển, một bộ nhớ toàn cục, và một tập vô hạn các bộ xử lý; mỗi bộ xử lý có bộ nhớ cục bộ riêng. Mặc dù các bộ xử lý đang hoạt động thực hiện các lệnh giống nhau; nhưng mỗi bộ xử lý có một chỉ số duy nhất và chỉ số này có thể được sử dụng để

kích hoạt (enable) hoặc dừng hoạt động (disabled). Chỉ số này cũng được gắn với vị trí bộ nhớ mà bộ xử lý truy nhập đến.



Hình 2.2. Mô hình xử lý song song PRAM

Xử lý PRAM bắt đầu với đầu vào được lưu trong bộ nhớ toàn cục và một bộ vi xử lý được kích hoạt. Tại mỗi bước, một bộ xử lý đang hoạt động có thể thực hiện một trong các thao tác: đọc dữ liệu từ bộ nhớ riêng cục bộ hay bộ nhớ toàn cục, thực hiện một thao tác xử lý RAM, ghi dữ liệu vào bộ nhớ riêng cục bộ hoặc toàn cục, hoặc kích hoạt bộ xử lý khác. Tất cả các bộ xử lý đang hoạt động phải thực hiện cùng một lệnh, có thể trên các địa chỉ khác nhau. Xử lý PRAM kết thúc khi bộ xử lý cuối cùng dừng hoạt động.

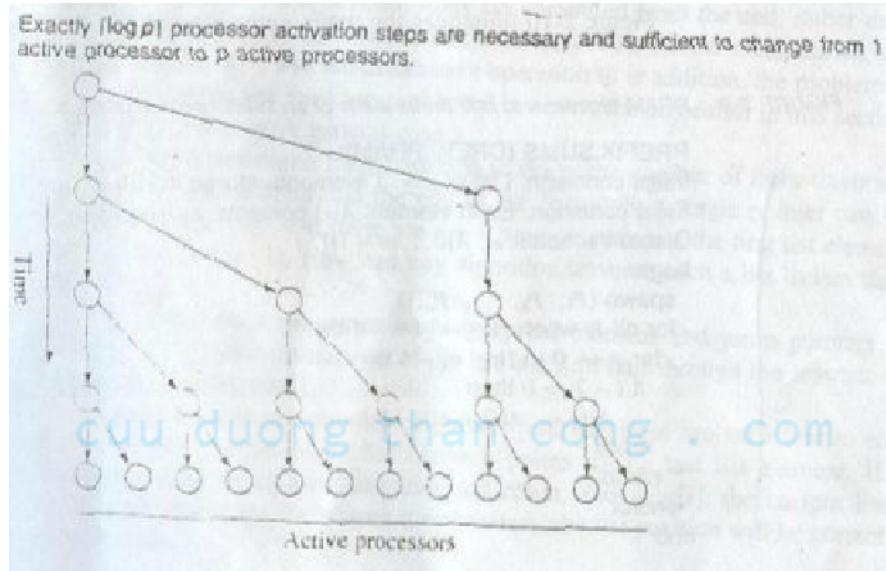
Các mô hình PRAM khác nhau ở chỗ làm thế nào để giải quyết (handle) được các xung đột từ thao tác ghi và đọc bộ nhớ toàn cục. Chẳng hạn, khi có nhiều hơn một bộ xử lý đọc hoặc ghi vào cùng một địa chỉ trong bộ nhớ. Các kết quả nghiên cứu về các thuật toán PRAM dựa vào một trong các cơ chế sau sau đây:

1. EREW (Exclusive Read Exclusive Write): Không cho phép xung đột Đọc và Ghi.
2. CREW (Concurrent Read Exclusive Write): Cho phép xung đột Đọc và không cho phép xung đột Ghi. Tại cùng một thời điểm, nhiều bộ xử lý có thể đọc đến cùng một địa chỉ trong bộ nhớ toàn cục nhưng không có quá một bộ xử lý được phép ghi.
3. CRCW (Concurrent Read Concurrent Write): Cho phép xung đột Đọc và cho phép xung đột Ghi. Tại cùng một thời điểm, nhiều bộ xử lý có thể đọc hoặc ghi cùng một địa chỉ trong bộ nhớ toàn cục theo một trong ba mô hình dưới đây:
 - *Thông thường (common)*: tại cùng một thời điểm, tất cả các bộ xử lý phải ghi cùng một giá trị vào cùng một địa chỉ trong bộ nhớ toàn cục.
 - *Ngẫu nhiên (arbitrary)*: tại cùng một thời điểm, nếu có nhiều bộ xử lý đồng thời ghi vào một địa chỉ trong bộ nhớ toàn cục, thì một trong số các bộ xử lý đó được chọn ngẫu nhiên sẽ được ghi giá trị vào ô nhớ đó.

- *Ưu tiên (priority)*: tại cùng một thời điểm, nếu có nhiều bộ xử lý đồng thời ghi vào một địa chỉ trong bộ nhớ toàn cục, thì bộ xử lý có chỉ số nhỏ nhất sẽ được ghi giá trị vào ô nhớ đó.

2.1.3 Một số thuật toán PRAM

Khi một thuật toán PRAM có độ phức tạp về thời gian tốt hơn (thấp hơn) so với một thuật toán RAM đã được tối ưu tương ứng, thì đó là do trong thuật toán PRAM đã có các lệnh được xử lý song song. Do thuật toán PRAM bắt đầu với một bộ xử lý hoạt động nên mọi thuật toán PRAM luôn có hai pha. Trong pha đầu, một số lượng đủ lớn bộ xử lý tham gia thực hiện thuật toán được kích hoạt. Trong pha thứ hai, các bộ xử lý đã được kích hoạt thực hiện xử lý thuật toán song song. Với một bộ xử lý ban đầu được kích hoạt, dễ nhận thấy rằng ta phải cần đến $\lceil \log p \rceil$ bước để kích hoạt p bộ xử lý.



Hình 2.3. Thời gian kích hoạt các bộ xử lý

Để kích hoạt các bộ vi xử lý hoạt động từ một bộ xử lý, các thuật toán PRAM thực hiện một câu lệnh sau:

Spawn(<tên các bộ vi xử lý>)

Để dễ dàng hiểu về pha thứ 2 của thuật toán PRAM, ta giả sử khi tham chiếu (reference) đến các thanh ghi toàn cục như một dãy các tham chiếu. Nghĩa là, có một ánh xạ từ dãy các tham chiếu này đến các thanh ghi toàn cục tương ứng.

Cấu trúc điều khiển

For all <danh sách các bộ vi xử lý> do <danh sách các câu lệnh> endfor

Mô tả đoạn mã được thực hiện song song bởi các bộ xử lý cụ thể.

Bên cạnh các cấu trúc điều khiển trên, PRAM cũng sử dụng cấu trúc điều khiển quen thuộc như: if ...then...else...endif, for...endfor, while...endwhile và repeat...until.

Dưới đây chúng ta xem xét một số thuật toán PRAM

a. Thuật toán song song Rút gọn (Parallel Reduction)

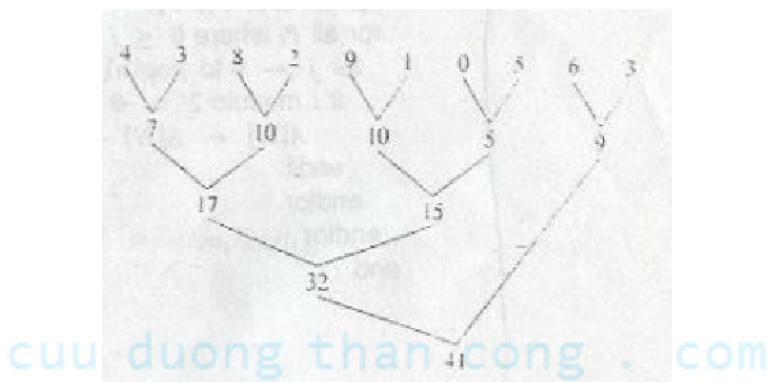
Cây nhị phân là một trong những cấu trúc quan trọng nhất của xử lý song song. Trong nhiều giải thuật trên-xuống (top-down), luồng dữ liệu đi từ nút gốc đến các nút lá của cây. Chẳng hạn như giải thuật phân phát (broadcast) thì nút gốc gửi dữ liệu đến tất cả các nút lá. Giải thuật chia để trị (divide and conquer) thì nút gốc biểu diễn bài toán ban đầu, các nút con biểu diễn các bài toán con.

Một số thuật toán dưới đi lên (bottom-up), thì luồng dữ liệu đi từ các nút lá đến nút gốc của cây. Đó là giải thuật rút gọn.

Bài toán được phát biểu như sau: “cho một tập n số a_1, a_2, \dots, a_n và một phép toán nhị phân kết hợp \oplus , rút gọn là một quá trình tính $a_1 \oplus a_2 \oplus \dots \oplus a_n$ ”

Tính tổng song song là một ví dụ của phép toán rút gọn.

Các bộ xử lý trong PRAM thao tác với dữ liệu được lưu trữ trong các thanh ghi toàn cục. Để triển khai thuật toán tính tổng, ta biểu diễn mỗi nút của cây nhị phân là một phần tử của mảng.



Hình 2.4. Tính tổng một mảng

Giả sử các số cần tính tổng được lưu vào một mảng A, kích thước n. Dưới đây là chương trình giả mă:

SUM (EREW PRAM)

Đầu vào: Danh sách $n > 0$ phần tử được lưu trong $A[0, \dots, (n-1)]$

Đầu ra: Tổng các phần tử được lưu trong $A[0]$

Các biến toàn cục: $n, A[0, \dots, (n-1)]$

Begin

 Spawn($P_0, P_1, \dots, P_{\lfloor n/2 \rfloor}$)

 For all P_i với $(0 \leq i \leq \lfloor n/2 \rfloor)$ do

 For $j=0$ to $\lceil \log N \rceil - 1$ do

 If $i \bmod 2^j = 0$ and $2^j \cdot i + 2^j < n$ then

$A[2^j \cdot i] = A[2^j \cdot i] + A[2^j \cdot i + 2^j]$

 Endif

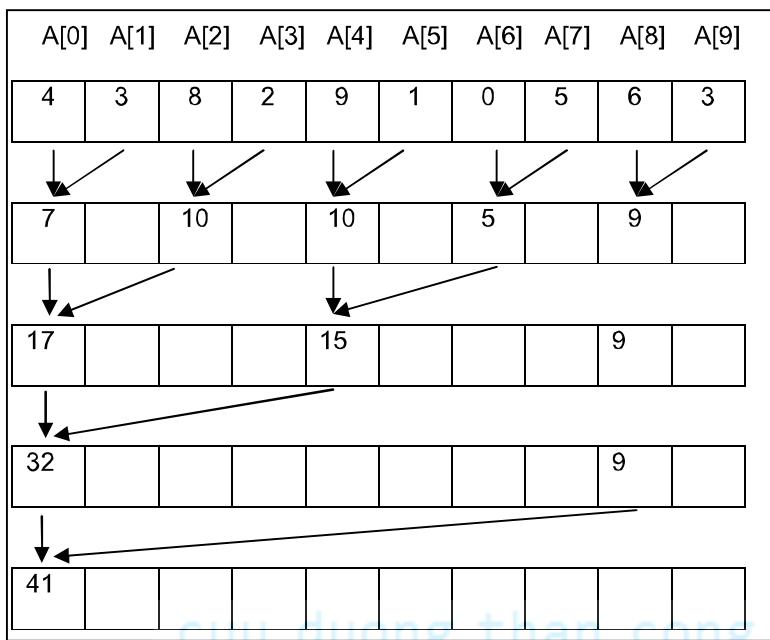
 Endfor

 Endfor

End.

Ta hãy phân tích độ phức tạp của thuật toán này. Thủ tục Spawn có độ phức tạp $\log(n/2)$ vì phải kích hoạt $n/2$ bộ xử lý dùng cho thuật toán. Vòng lặp tuần tự thực hiện logn lần, và mỗi bước lặp có độ phức tạp thời gian là hằng số. Vì vậy, độ phức tạp chung của thuật toán là $O(\log n)$.

Hình dưới đây mô tả từng bước của giải thuật qua một ví dụ về tính tổng của một mảng gồm 10 phần tử:



b. Thuật toán song song Tính tổng tiền tố (Prefix Sums)

Bài toán Tính tổng tiền tố như sau: “cho một tập gồm n số a_1, a_2, \dots, a_n và một phép toán nhị phân kết hợp \oplus , hãy tính các biểu thức:

$$\begin{aligned} & a_1 \\ & a_1 \oplus a_2 \oplus \\ & a_1 \oplus a_2 \oplus a_3 \\ & \dots \\ & a_1 \oplus a_2 \oplus \dots \oplus a_n \end{aligned}$$

Chẳng hạn, nếu phép toán là phép cộng (+) và đầu vào là mảng số nguyên [3,1,0,4,2], thì tổng tiền tố của mảng là [3,4,4,8,10].

Tổng tiền tố được gọi là tiền tố song song và quét (scan). Tổng tiền tố có nhiều ứng dụng trong thực tế. Ví dụ về nén kí tự như sau: giả sử ta có một mảng A gồm n kí tự. Ta muốn nén các kí tự hoa vào các vị trí đầu của mảng mà vẫn giữ nguyên thứ tự. Trình tự tính toán thể hiện trong hình dưới đây:

(a) Mảng A:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | b | C | D | e | F | g | h | I |
|---|---|---|---|---|---|---|---|---|

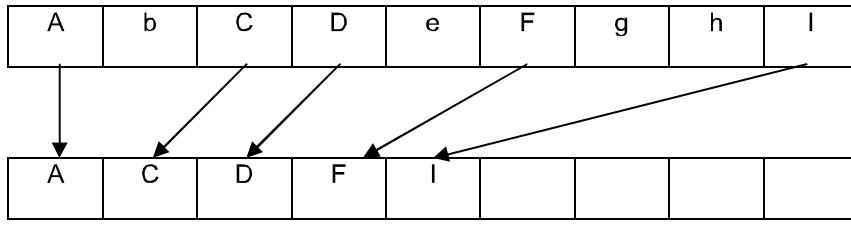
(b) Mảng T:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|

(c) Mảng T (sau khi tính tổng tiền tố):

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 3 | 4 | 4 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|

(d) Mảng A (sau khi được nén):



Hình 2.5. Nén các phần tử của mảng là một ứng dụng của Tổng tiền tố.

Bước đầu tiên (bước b) là khởi động một mảng phụ kích thước n gồm các phần tử 0 và 1. Phần tử $T[i] = 0$ nếu phần tử $A[i]$ là kí tự thường và $T[i] = 1$ nếu $A[i]$ là kí tự hoa. Tiếp theo (bước c), ta tính tổng tiền tố các phần tử của mảng T. Kết quả ta thu được là $A[T[i]]$ chứa các kí tự hoa. Dưới đây là giả mã của thuật toán song song tính tổng tiền tố.

PREFIX.SUM (CREW PRAM)

Đầu vào: Danh sách $n > 0$ phần tử được lưu trong $A[0, \dots, (n-1)]$

Đầu ra: Mỗi phần tử $A[i] = A[0] + A[1] + \dots + A[i-1]$

Các biến toàn cục: $n, A[0, \dots, (n-1)], j$

Begin

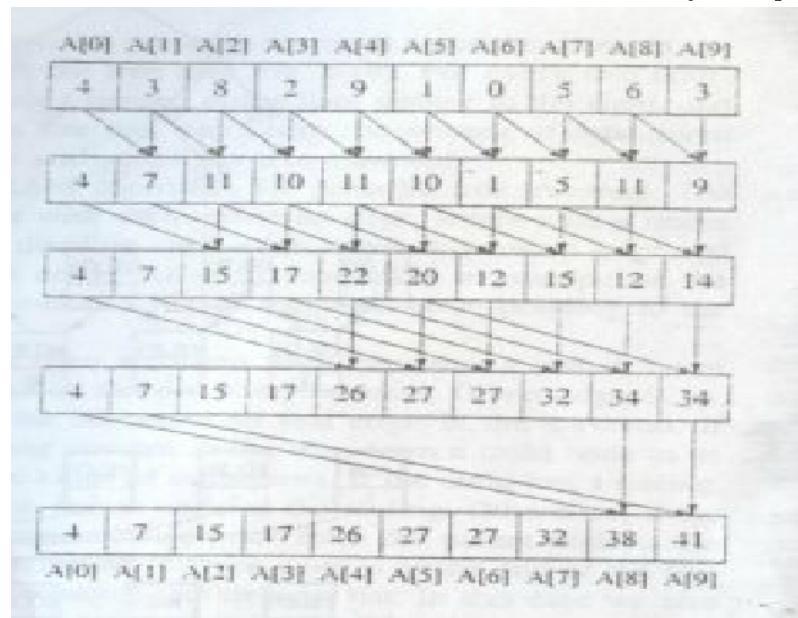
```

        Spawn( $P_0, P_1, \dots, P_{[n-1]}$ )
        For all  $P_i$  với ( $0 \leq i \leq [n-1]$ ) do
            For  $j=0$  to  $\lceil \log(n) \rceil - 1$  do
                If  $i - 2^j \geq 0$  then
                     $A[i] = A[i] + A[i - 2^j]$ 
                Endif
            Endfor
        Endfor
    End.

```

Độ phức tạp của thuật toán trên tương tự như thuật toán tìm tổng. Thủ tục spawn có độ phức tạp là $O(\log(n))$. Vòng lặp tuần tự thực hiện $\lceil \log(n) \rceil$ lần, mỗi lần lặp cần thời gian là hằng số. Vì vậy, độ phức tạp của thuật toán trên là $O(\log(n))$.

Dưới đây là minh họa các bước thực hiện của thuật toán đối với một mảng có 10 phần tử:



Hình 2.6.Tổng tiền tố của một mảng gồm 10 phần tử

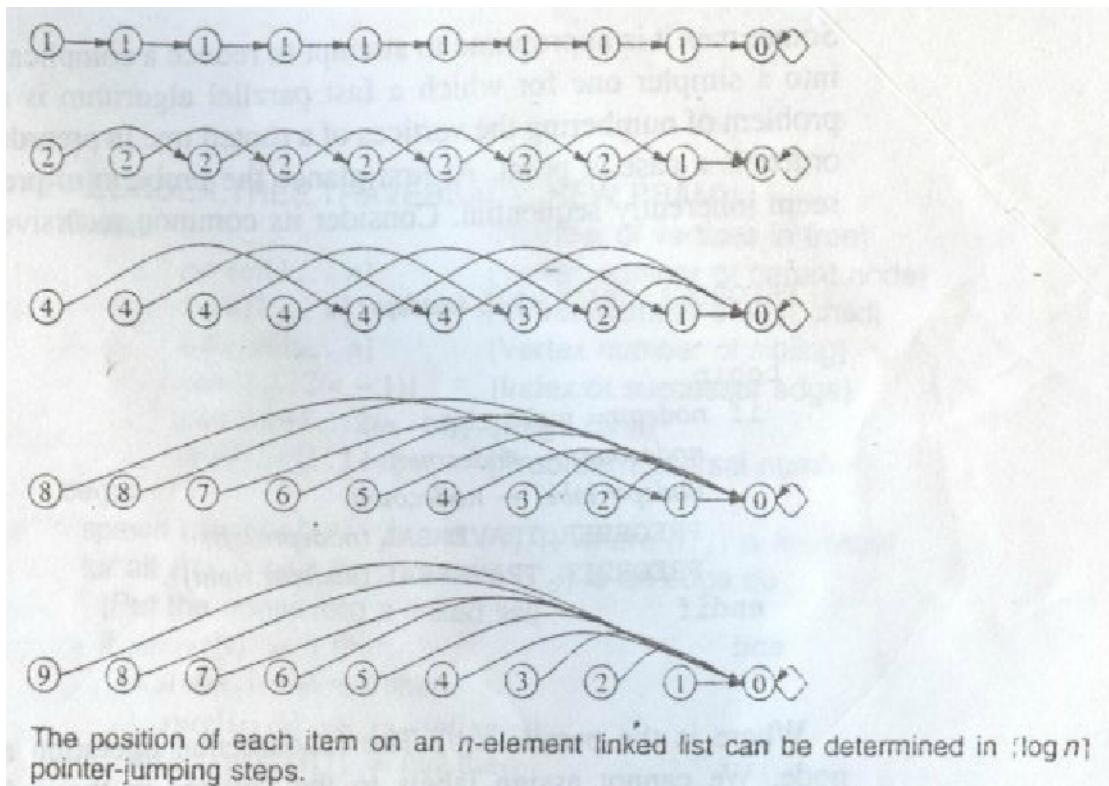
c. Thuật toán song song Xếp loại danh sách (List Ranking)

Xét bài toán tính các tổng hậu tố (suffix sums) của i phần tử cuối cùng trong một danh sách gồm n phần tử với $0 \leq i \leq n$. Bài toán tìm các tổng hậu tố là một biến thể (variant) của bài toán tìm tổng các tiền tố mà ta đã xét ở trên với mảng được thay thế bởi một danh sách liên kết, và các tổng được tính từ cuối danh sách. Trong trường hợp mỗi phần tử trong danh sách là 0 hoặc 1, và toán tử kết hợp \oplus là phép cộng thì bài toán trở thành bài toán xếp loại danh sách.

Một cách để xác định vị trí danh sách là đếm số liên kết chuyển tiếp giữa các phần tử và liên kết cuối cùng trong danh sách. Như vậy, danh sách gồm n con trỏ, vậy liệu tồn tại một thuật toán duyệt danh sách với thời gian ít hơn $O(n)$.

Nếu ta gắn một bộ xử lý với mỗi phần tử trong danh sách và các con trỏ có thể di chuyển một cách song song, khoảng cách tới điểm cuối cùng của danh sách được giảm một nửa bởi lệnh $next[i] = next[next[i]]$.

cuu duong than cong . com



Hình 2.7.Tìm vị trí của một phần tử trong một danh sách n phần tử

Vì vậy, có thể ta chỉ cần $O(\log(n))$ bước nhảy là đủ để mọi phần tử trong danh sách có thể trỏ tới được phần tử cuối cùng. Nếu một bộ xử lý bổ sung vào biến đếm $position[i]$ để lưu số lần duyệt qua các liên kết. Nhờ vậy, vị trí danh sách sẽ được xác định.

Dưới đây là thuật toán PRAM để xác định vị trí của mỗi phần tử trong một danh sách liên kết đơn.

cuu duong than cong . com

LIST.RANKING (CREW PRAM)

Đầu vào:Các giá trị trong mảng next biểu diễn một danh sách liên kết.
Đầu ra: Giá trị trong mảng position chứa khoảng cách ban đầu của mỗi phần tử từ cuối danh sách.

Các biến toàn cục: n, position[0, ...(n-1)], next [0, ...(n-1)], j

Begin

```
    Spawn(P0, P1, ..., Pn-1)
    For all Pi với (0≤i≤n-1) do
        If next[i]= i then position[i]=0
        else
            position[i]=1
        Endif
        For j=1 to [log(n)] do
            position[i]= position[i]+ position[next[i]]
            next[i]= next[next[i]]
        Endfor
    Endfor
End.
```

Thủ tục Spawn cần thời gian logn để kích hoạt n bộ xử lý hoạt động. Thời gian thực hiện các lệnh trong vòng lặp là hằng số và được lặp logn lần, nên độ phức tạp của thuật toán song song là O(logn).

d. Thuật toán song song trộn hai danh sách đã sắp xếp (Merging Two Sorted Lists)

Một thuật toán RAM tối ưu tại một thời điểm tạo ra danh sách trộn gồm một phần tử và cần nhiều nhất là n-1 phép so sánh để trộn hai danh sách gồm n/2 phần tử. Vì vậy, thuật toán RAM trộn hai danh sách đã được sắp xếp thành một danh sách được sắp xếp với độ phức tạp O(n).

Một thuật toán PRAM có thể thực hiện công việc trên với độ phức tạp O(logn) bằng cách gán mỗi phần tử trong danh sách cho mỗi bộ xử lý. Mọi bộ xử lý tìm vị trí cho phần tử của nó trong danh sách còn lại bằng cách sử dụng tìm kiếm nhị phân. Do ta đã biết trước chỉ số của các phần tử trên danh sách của nó, vị trí của nó trong danh sách trộn có thể được tính khi chỉ số của nó trên danh sách còn lại được tìm thấy và hai chỉ số được bổ sung. Tất cả các phần tử có thể được chèn vào trong danh sách trộn với thời gian hằng số.

Chương trình giả mã dưới đây biểu diễn thuật toán PRAM trộn hai danh sách :

Để đơn giản, trong phiên bản này của thuật toán ta giả sử các giá trị của hai danh sách không giao nhau.

cuu duong than cong . com

MERGE.LISTS (CREW PRAM)

Đầu vào: Hai danh sách gồm $n/2$ phần tử đã được sắp xếp $A[1..(n/2)]$ và $A[(n/2)+1..n]$.

Đầu ra: Danh sách $A[1..n]$ đã được sắp xếp, được trộn từ hai danh sách trên.

Các biến toàn cục: n , $A[1..n]$

Các biến cục bộ: x , low, high, index

Begin

```
Spawn(P1, P2, ..., Pn)
For all Pi với (1 ≤ i ≤ n) do
    /*Mỗi bộ xử lý thiết lập biên cho tìm kiếm nhị phân*/
    If i ≤ (n/2) then
        Low = (n/2)+1
        High = n
    else
        Low = 1
        High = (n/2)
    Endif
    /*Mỗi bộ xử lý thực hiện tìm kiếm nhị phân*/
    x=A[i]
    Repeat
        Index=[(low+high)/2]
        If x ≤ A[index] then
            High = index-1
        else
            Low = index+1
        Endif
        until low>high
    /*Đưa giá trị vào đúng vị trí trong danh sách trộn*/
    A[High+i-(n/2)]=x
Endfor
```

End

Trộn hai danh sách đã sắp xếp vào một danh sách

Như thường lệ, các bộ vi xử lý được kích hoạt tại bước đầu tiên của thuật toán. Trong thuật toán này ta cần n bộ xử lý, mỗi bộ xử lý tìm vị trí cho một phần tử từ hai danh sách ban đầu. Sau khi các bộ xử lý được kích hoạt, chúng sẽ xác định khoảng chỉ số mà chúng tìm kiếm một cách song song. Các bộ xử lý gắn với các phần tử trong nửa “thấp” của mảng (nửa chứa các giá trị bé hơn phần tử đang cần tìm vị trí) sẽ được thực hiện tìm kiếm nhị phân trên các phần tử trong nửa “cao” của mảng và ngược lại.

| | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| A[1] A[2] A[3] A[4] A[5] A[6] A[7] A[8] | | | | | | | | | | | | | | | |
| 1 5 7 9 13 17 19 23 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| 1 2 4 5 7 8 9 11 12 13 17 19 21 22 23 24 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| 2 4 6 8 10 21 23 24 | | | | | | | | | | | | | | | |
| A[9] A[10] A[11] A[12] A[13] A[14] A[15] A[16] | | | | | | | | | | | | | | | |

Hình 2.8. Trộn hai mảng đã sắp xếp thành một mảng đã sắp xếp

Mỗi bộ xử lý có duy nhất một giá trị x , một phần tử sẽ được trộn. Vòng lặp repeat...until thực hiện tìm kiếm nhị phân. Khi một bộ xử lý thoát khỏi vòng lặp thì biến $high$ sẽ được gán lại giá trị là chỉ số của phần tử lớn nhất trong danh sách chứa các phần tử nhỏ hơn x .

Xét một bộ xử lý P_i và giá trị $A[i]$ được gán cho P_i trong nửa “thấp” của danh sách. Giá trị low cuối cùng phải nằm giữa $(n/2)$ và n . phần tử $A[i]$ lớn hơn $i-1$ phần tử trong nửa thấp của danh sách. Đồng thời, nó cũng lớn hơn $high-(n/2)$ phần tử trên nửa cao của danh sách. Do vậy, ta sẽ đặt $A[i]$ vào danh sách được sắp xếp sau $i+high-(n/2)-1$ phần tử khác và nó có chỉ số $i+high-(n/2)$.

Trường hợp tiếp theo ta xét bộ xử lý P_i và giá trị $A[i]$ được gán cho P_i trong nửa “cao” của danh sách. Giá trị của biến $high$ phải nằm giữa 0 và $(n/2)$. Phần tử $A[i]$ lớn hơn $i-(n/2)-1$ phần tử khác trong nửa cao của danh sách và lớn hơn $high$ phần tử trong nửa thấp của danh sách. Do vậy, $A[i]$ sẽ được đặt vào danh sách được sắp xếp sau $i+high-(n/2)-1$ phần tử khác và nó có chỉ số $i+high-(n/2)$.

Do tất cả các bộ xử lý sử dụng cùng một cách tìm vị trí cho các phần tử trong danh sách trộn, nên mọi bộ xử lý cùng sử dụng một phép gán khi kết thúc thuật toán.

Tổng số thao tác được thực hiện để trộn các danh sách tăng từ $O(n)$ trong thuật toán tuần tự lên $O(n\log n)$ trong thuật toán song song. Tuy nhiên, với n bộ xử lý thực hiện song song thì độ phức tạp về thời gian chỉ còn là $O(\log n)$. Với giả thiết (cho các thuật toán PRAM) là số bộ xử lý là vô hạn thì thuật toán này hiệu quả đáng kể. Nhưng khi cài đặt thực tế thì số lượng bộ xử lý là có hạn, nên ta phải xem xét chi phí thực tế cho thuật toán.

2.2 Các thuật toán song song nhân hai ma trận

Phần này giới thiệu về các thuật toán song song nhân ma trận được thực hiện cho các mô hình SIMD trên các máy tính song song khác nhau. Thuật toán song song nhân hai ma trận trên máy

SIMD với tổ chức bộ vi xử lý theo mạng hình lưới là O(n); Một thuật toán song song được thiết kế cho máy SIMD với các bộ xử lý được tổ chức theo mạng siêu khối và mạng hoán vị di chuyển với độ phức tạp là O(logn).

2.2.1 Thuật toán nhân ma trận tuần tự

Tích của ma trận A kích thước l x m với một ma trận B kích thước m x n là một ma trận C kích thước l x n, mà các phần tử của nó được xác định bởi:

$$C[i,j] = \sum_{k=1}^{m-1} A[i,k] * B[k,j]$$

Một thuật toán tuần tự nhân hai ma trận cấp n có độ phức tạp O(n³) vì nó cần tới n³ phép cộng và n³ phép nhân.

MATRIX_MULTIPLICATION()

Đầu vào: Hai ma trận: A[1..m,1..n], B[1..n,1..k].

Đầu ra: Ma trận C[1..m, 1..k] là ma trận tích của A và B

Begin

 For i=1 to m do

 For j=1 to k do

 Begin

 t=0

 for h=1 to n do

 t=t+A[i,h]*B[h,j]

 endfor

 C[i,j]=t

 End;

 Endfor

 Endfor

 End.

2.2.2 Thuật toán nhân ma trận trên máy SIMD với các bộ xử lý được tổ chức theo mạng hình lưới hai chiều (2-D Mesh SIMD).

Cận dưới của thuật toán: Gentleman đã chỉ ra rằng nhân hai ma trận cấp n x n trên máy tính SIMD với các bộ xử lý được tổ chức theo hình lưới 2 chiều cần không ít hơn Ω(n) bước định tuyến dữ liệu.

Định nghĩa 2.1: Cho một mục dữ liệu ban đầu có sẵn trên một bộ xử lý trong một mô hình tính toán song song nào đó, và p(k) là số lượng lớn nhất các bộ xử lý mà dữ liệu có thể chuyển tới trong k hoặc ít hơn k bước định tuyến dữ liệu (data routing steps).

Chẳng hạn, trong máy tính SIMD với các bộ xử lý được tổ chức theo hình lưới 2 chiều thì p(0)=1, p(1)=5, p(2)=13 và trong trường hợp tổng quát thì p(k)=2k²+2k+1.

Bố đề 7.1. Giả sử ta muốn nhân hai ma trận A và B kích thước $n \times n$, và mỗi phần tử của A và B được lưu đúng một lần và không có bộ xử lý nào chứa nhiều hơn một phần tử của ma trận còn lại. Nếu ta bỏ qua sự thuận tiện về truyền thông (broadcasting facility), phép nhân hai ma trận A và B để tạo ra ma trận C yêu cầu ít nhất s bước định tuyến dữ liệu, mà $p(2s) \geq n^2$.

Chứng minh: Xét một phần tử bất kỳ $c[i,j]$ của ma trận tích. Phần tử này là kết quả của tổng các tích của các phần tử của hàng i của ma trận A và cột j của ma trận B. Do vậy, sẽ có một đường đi từ các bộ xử lý lưu các phần tử này tới bộ xử lý chứa kết quả $c[i,j]$. Gọi s là độ dài đường đi dài nhất như vậy. Nói cách khác, việc tạo ra ma trận C cần ít nhất s bước định tuyến dữ liệu.

Chú ý rằng, các đường đi cũng có thể được định nghĩa là một tập hợp các đường đi có độ dài lớn nhất $2s$ từ bất kỳ phần tử $B[u,v]$ tới mọi phần tử $A[i,j]$ nào đó. Do tồn tại một đường đi với độ dài không vượt quá s đi từ một bộ xử lý chứa phần tử $B[u,v]$ đến bộ xử lý chứa $C[i,v]$ và cũng có một đường đi với độ dài không vượt quá s đi từ $A[i,j]$ đến $C[i,v]$. Vì vậy, tồn tại một đường đi với độ dài không vượt quá $2s$ đi từ mỗi phần tử $B[u,v]$ đến $A[i,j]$. Tương tự như vậy các đường đi này xác định một tập các đường đi với độ dài không vượt quá $2s$ từ mọi phần tử $A[u,v]$ bất kỳ tới mọi phần tử $B[i,j]$, với mọi $1 \leq i,j \leq n$; n^2 phần tử của ma trận A được lưu trong các bộ xử lý riêng biệt (mỗi bộ xử lý chứa một phần tử). Do tồn tại một đường đi với độ dài không vượt quá $2s$ đi từ một bộ xử lý chứa $B[u,v]$, tới các bộ xử lý lưu các phần tử của ma trận A, nên theo định nghĩa 2.1 ở trên ta sẽ có $p(2s) \geq n^2$.

Định lý 7.1. Nhân ma trận trên máy tính SIMD với các bộ xử lý được tổ chức theo hình lưới 2 chiều yêu cầu $\Omega(n)$ bước định tuyến dữ liệu, hoặc với n lớn $s \geq 0.35n$.

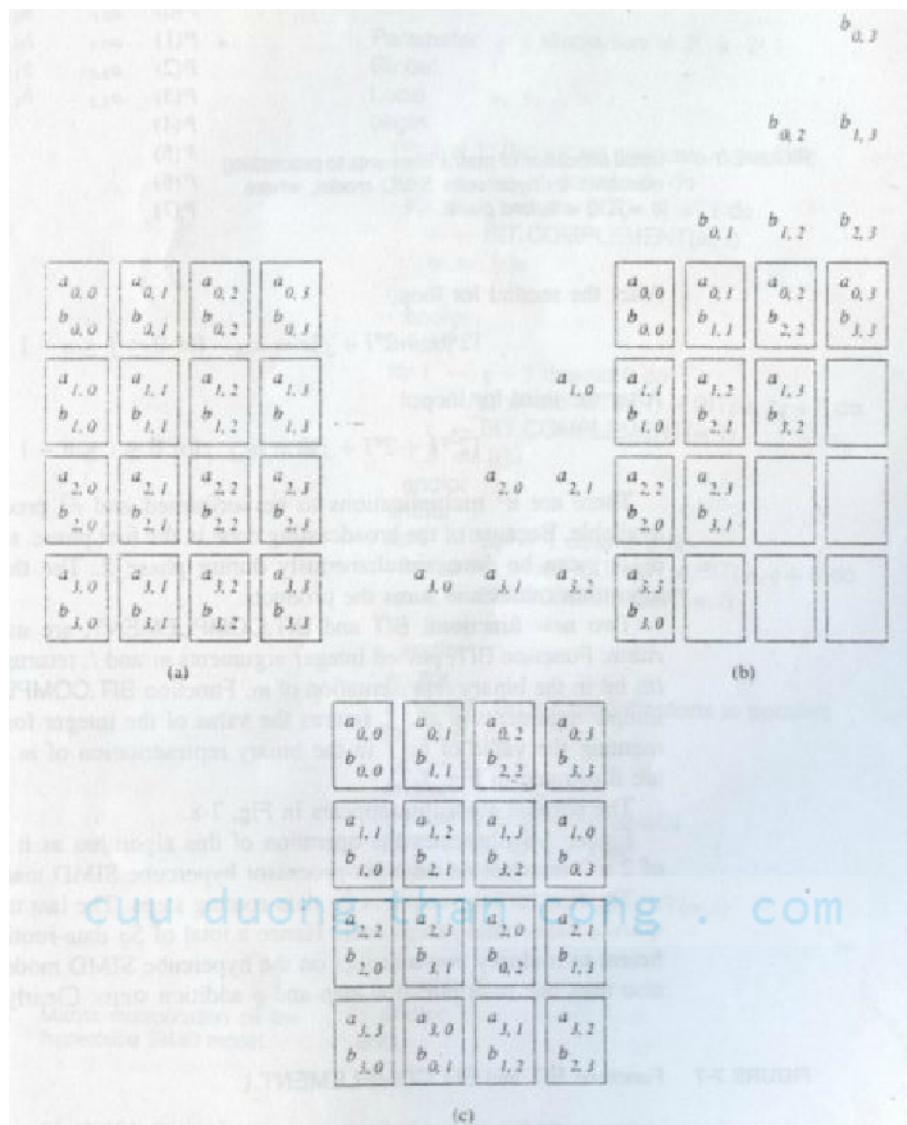
Chứng minh: Từ bố đề 7.1, ta có $p(2s) \geq n^2$, trên máy tính SIMD với các bộ xử lý được tổ chức theo hình lưới 2 chiều thì ta có $p(s) = 2s^2 + 2s + 1$, nên $p(2s) = 8s^2 + 4s + 1$, do đó

$$\begin{aligned} 8s^2 + 4s + 1 &\geq n^2 \\ \Leftrightarrow s^2 + (s/2) + 1/8 &\geq (n^2/8) \\ \Leftrightarrow (s + 1/4)^2 + 1/16 &\geq (n^2/8) \\ \Leftrightarrow s &\geq \sqrt{2n^2 - 1}/4 - 1/4 \approx 0.35n. \text{ khi } n \text{ đủ lớn.} \end{aligned}$$

Một thuật toán tối ưu,

Cho một máy tính SIMD với các bộ xử lý được tổ chức theo hình lưới 2 chiều có kết nối vòng (wrap-around), thì dễ nhận thấy có thể đưa ra một thuật toán sử dụng n^2 bộ xử lý để nhân hai ma trận kích thước $n \times n$ với độ phức tạp về thời gian là $\Theta(n)$.

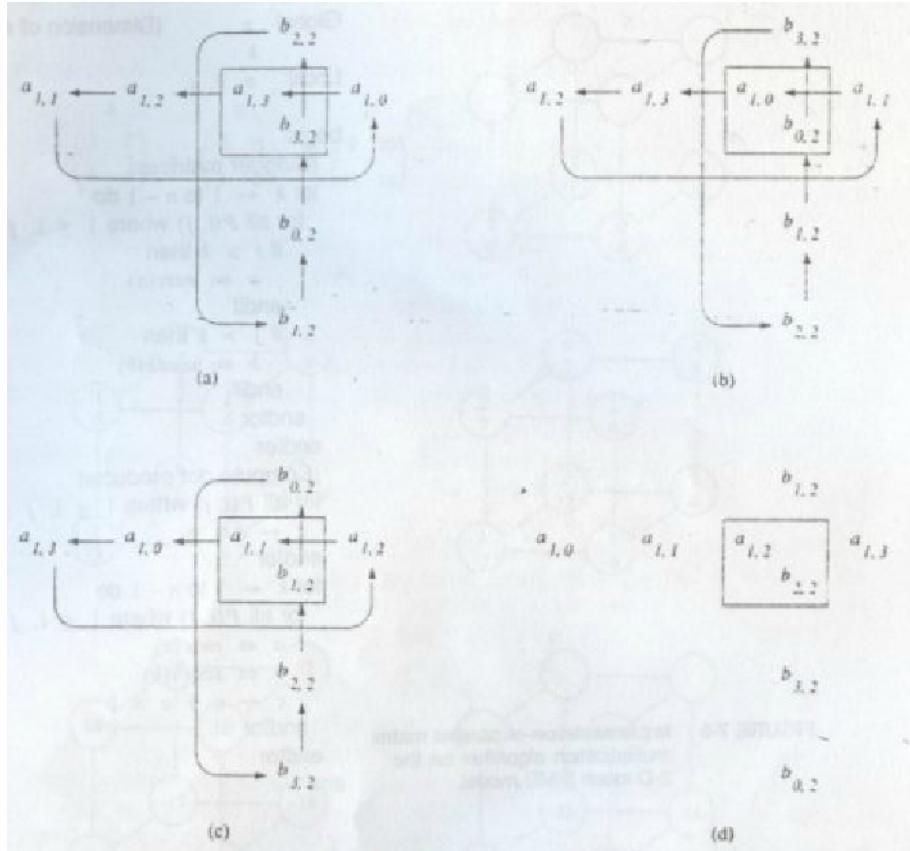
Trong thuật toán tuần tự cần n^3 phép nhân, và để n^2 bộ xử lý có thể hoàn thành tính nhân hai ma trận trong thời gian $\Theta(n)$ thì tất cả n^2 bộ xử lý phải cùng thực hiện tính toán ma trận kết quả tại mỗi bước. Pha khởi tạo cấp phát các phần tử của ma trận cho các bộ vi xử lý, được minh họa trên hình (?). Các bộ xử lý ở dòng i cột j trong mạng lưới chứa $A[i,j]$ và $B[i,j]$. Chú ý rằng, trong bước khởi tạo chỉ có n bộ xử lý chứa một cặp cho phép nhân. Tuy nhiên, nó có thể di chuyển ma trận A và B để mỗi bộ xử lý chứa một cặp phần tử cần được nhân với nhau (hình 2.9). Hơn nữa, mỗi lần quay lên trên của các phần tử trong ma trận B và quay trái các phần tử trong ma trận A sẽ làm cho mỗi bộ xử lý có một cặp phần tử mới để thực hiện phép nhân.



Hình 2.9. Thuật toán nhân ma trận trên máy SIMD tổ chức theo hình lưới 2 chiều

Ở pha sắp xếp các phần tử của thuật toán nhân ma trận trên máy tính SIMD với các bộ xử lý được tổ chức theo hình lưới 2 chiều. (a) Khởi tạo và phân phát các cặp phần tử $A[i,j]$ và $B[i,j]$ cho các bộ vi xử lý, sau đó thuật toán tiến hành nhân tất cả các cặp $(A[i,k], B[k,j])$ (chú ý rằng chỉ có các bộ xử lý $P(0,0), P(1,1), P(2,2)$ và $P(3,3)$ chứa các cặp như vậy). (b) thuật toán di chuyển mỗi hàng i của ma trận A sang bên trái bởi j vị trí cột. Đồng thời, di chuyển mỗi cột j của ma trận B lên phía trên bởi i vị trí hàng. (c) giống phần (b), sau khi di chuyển vòng quanh (wrap-around), bây giờ mỗi bộ xử lý $P(i,j)$ có một cặp phần tử để nhân.

Ta hãy xem các hành động của một bộ xử lý (hình ..). Sau khi các phần tử của hai ma trận A và B được di chuyển, bộ xử lý $P(1,2)$ thực hiện các phép nhân và cộng để tính $C[1,2]$.



Hình 2.9. Thuật toán nhân ma trận trên máy tính SIMD với các bộ xử lý được tổ chức theo hình lưới 2 chiều từ cách nhìn các bước thực hiện của bộ xử lý P(1,2). Các phần tử của ma trận đã được di chuyển. (a) là bước đầu tiên. (b) bước thứ hai được thực hiện sau khi các phần tử của được “quay vòng” sang trái và các phần tử của B được di chuyển lên trên. (c) bước 3 được thực hiện sau bước (b). (d) được thực hiện sau bước 3, ở bước này bộ xử lý P(1,2) đã tính xong C[1,2].

MATRIX_MULTIPLICATION(2-D MESH SIMD)
Đầu vào: Hai ma trận: A[0..l-1,0..m-1], B[0..m-1,0..n-1].
Đầu ra: Ma trận C[0..l-1, 0..n-1] là ma trận tích của A và B
Biến toàn cục: n,k
Biến cục bộ: a,b,c

```

Begin
    /*Di chuyển ma trận*/
    For k=1 to n-1 do
        For all P(i,j) where 1 ≤ i,j ≤ n do
            If i > k then
                a = east(a)
            endif
            If i > k then
                b = south(b)
            endif
        Endfor
    Endfor
    /*tính tích */
    For all P(i,j) where 1 ≤ i,j ≤ n do
        c = a *b
    Endfor
    For k=1 to n-1 do
        For all P(i,j) where 1 ≤ i,j ≤ n do
            a = east(a)
            b = south(b)
            c = c + a*b
        endfor
    Endfor
Endfor
End.

```

Pha đầu của thuật toán là phân phát và di chuyển các phần tử của hai ma trận. Pha thứ hai tính tất cả các tích $A[i,k]*B[k,j]$ và tính tổng chúng lại với nhau.

2.2.3 Thuật toán nhân ma trận trên máy SIMD với các bộ xử lý được tổ chức theo mạng hình siêu khối (Hypercube SIMD).

Định lý 2.2: Cho một máy tính SIMD với các bộ vi xử lý được tổ chức theo hình siêu khối với $n^3=3^{3q}$ bộ xử lý, thuật toán hai ma trận kích thước $n \times n$ thực hiện trên máy tính này có độ phức tạp $\Theta(\log n)$ (Dekel et al., 1989).

Chứng minh: Ý tưởng chính của giải thuật nằm ở chỗ cài tiến chiến lược định tuyến dữ liệu; chỉ cần $5q=5\log_3 n$ bước định tuyến là đủ để phân phát các giá trị dữ liệu ban đầu qua một mảng các bộ xử lý và kết hợp các kết quả.

Các bộ xử lý được xem như một lưới (lattice) 3 chiều $n \times n \times n$. Bộ xử lý $P(x)$, với $0 \leq x \leq 2^{3q}-1$ có các vị trí a, b, c, s và t trong bộ nhớ cục bộ.

Khi thuật toán song song bắt đầu thực hiện. Trong suốt pha đầu $A[i,j]$ và $B[i,j]$ với $0 \leq i,j \leq n-1$, được lưu trong các biến a và b của bộ xử lý $P(2^q*i + j)$ (xem giả mã). Sau khi thuật toán song song hoàn thành thì ma trận $C[i,j]$ với $0 \leq i,j \leq n-1$, được lưu trong biến c của bộ xử lý $P(2^q*i + j)$. Thuật toán có 3 pha. Trong pha đầu, $A[i,j]$ và $B[i,j]$ được phân phát cho các bộ xử lý còn lại. Sau vòng lặp for thứ nhất thì:

$$[2^{2q}*k + 2^q*i + j] a = A[i,j]$$

$$[2^{2q}*k + 2^q*i + j] b = B[i,j]$$

Với mọi $0 \leq k \leq n-1$,

Sau vòng lặp for thứ hai thì:

$$[2^{2q}*k + 2^q*i + j] a = A[i,k] \text{ với } 0 \leq j \leq n-1$$

Sau vòng lặp for thứ ba thì:

$$[2^{2q}*k + 2^q*i + j] a = B[k,j] \text{ với } 0 \leq i \leq n-1$$

Có n^3 phép nhân được thực hiện và sử dụng n^3 bộ xử lý. Do quá trình phân phát hoàn thành trong pha đầu tiên thì tất cả các tích $A[i,k] * B[k,j]$ có thể được hoàn thành một cách đồng thời trong pha thứ 2. Pha thứ 3 của thuật toán thực hiện việc định tuyến và tính tổng các phép nhân.

Có hai hàm mới là BIT và BIT.COMPLEMENT được sử dụng trong thuật toán này. Hàm BIT nhận đầu vào là hai số nguyên m và l , trả lại giá trị là bít thứ l trong biểu diễn nhị phân của m . Hàm BIT.COMPLEMENT nhận đầu vào là hai số nguyên m và l , trả lại giá trị là một số nguyên được sinh ra do phép bù bít thứ l trong biểu diễn nhị phân của m .

Dưới đây là các ví dụ về hàm BIT và BIT.COMPLEMENT

$$\text{BIT}(9,0)=1 \quad \text{BIT.COMPLEMENT}(9,0)=8$$

$$\text{BIT}(9,1)=0 \quad \text{BIT.COMPLEMENT}(9,1)=11$$

$$\text{BIT}(9,3)=1 \quad \text{BIT.COMPLEMENT}(9,3)=1$$

$$\text{BIT}(9,4)=0 \quad \text{BIT.COMPLEMENT}(9,4)=25$$

$$\text{BIT}(9,5)=0 \quad \text{BIT.COMPLEMENT}(9,5)=41$$

Thuật toán song song đầy đủ được trình bày ở dưới đây.

Ví dụ minh họa (hình ?) các bước của thuật toán khi thực hiện nhân hai ma trận kích thước 2×2 trên một máy tính SIMD với các bộ vi xử lý được tổ chức theo hình siêu khối với 8 bộ xử lý.

Vòng lặp for đầu tiên yêu cầu $2q$ bước định tuyến dữ liệu. Ba vòng lặp cuối yêu cầu q bước định tuyến dữ liệu cho mỗi vòng. Do đó, tổng số cần $5q$ bước định tuyến dữ liệu là đủ để nhân hai ma trận trên một máy tính SIMD với các bộ vi xử lý được tổ chức theo hình siêu khối. Đồng thời, thuật toán cũng sử dụng 1 bước thực hiện phép nhân và q bước thực hiện phép cộng. Từ đó ta có độ phức tạp của thuật toán song song nhân ma trận thực hiện trên một máy tính SIMD với n^3 bộ xử lý được tổ chức theo hình siêu khối là $\Theta(q) = \Theta(\log n)$.

MATRIX.MULTIPLICATION (Hypercube SIMD)

Tham số: q {kích thước ma trận là $2^q \times 2^q$ }

Biến toàn cục: l

Biến cục bộ: a,b,c,s,t

Begin

{Pha 1: phân phát các phần tử A[i,j] và B[i,j]}

For l = 3q-1 downto 2q do

For all P_m với BIT(m,l)=1 do

t = BIT.COMPLEMENT(m,l)

a = [t]a

b = [t]b

endfor

endfor

For l = q-1 downto 0 do

For all P_m với BIT(m,l) ≠ BIT(m,2q+l) do

t = BIT.COMPLEMENT(m,l)

a = [t]a

endfor

endfor

For l = 2q-1 downto q do

For all P_m với BIT(m,l) ≠ BIT(m,q+l) do

t = BIT.COMPLEMENT(m,l)

b = [t]b

endfor

endfor

{Pha 2: thực hiện các phép nhân song song}

For all P_m do

c = a x b

endfor

{Pha 3: tính tổng}

For l = 2q to 3q-1 do

For all P_m do

t = BIT.COMPLEMENT(m,l)

s = [t] c

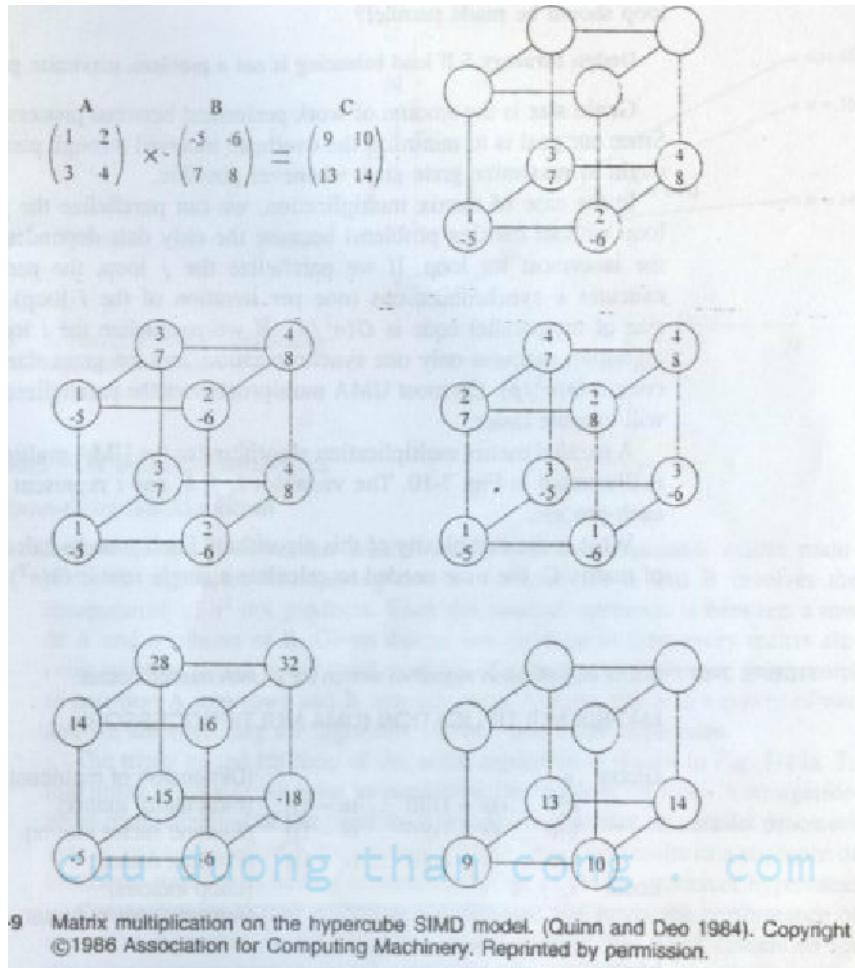
c = c+s

endfor

endfor

End.

Một ví dụ:



Hình 2.10. Thuật toán nhân ma trận trên máy SIMD siêu khối

2.2.4 Thuật toán nhân ma trận trên máy đa bộ xử lý.

Bài toán nhân ma trận thể hiện đầy đủ yếu tố song song hóa trên máy tính đa bộ xử lý. Trong thuật toán nhân ma trận tuần tự, ba vòng lặp For có thể song song hóa được. Điều này phát sinh một câu hỏi thú vị: có phải tất cả các vòng lặp (và các vòng lặp lồng nhau (nested loops)) có thể song song hóa được không?

Trước hết, ta tìm hiểu một khái niệm liên quan đến hiệu suất của các hệ thống song song: độ hoàn thành công việc cộng tác (grain size).

Độ hoàn thành công việc cộng tác là khối lượng công việc được thực hiện qua sự tương tác (interations) giữa các bộ vi xử lý. Do mục tiêu của ta là giảm các chi phí (overhead) do quá trình song song gây ra, bất cứ khi nào độ hoàn thành công việc cộng tác càng lớn càng tốt.

Xét bài toán nhân hai ma trận, ta có thể song song vòng lặp với j hoặc vòng lặp với i mà không bị ảnh hưởng bởi sự phụ thuộc dữ liệu từ vòng lặp trong cùng (innermost loop). Nếu ta song song hóa vòng lặp j , thuật toán song song thực hiện n lần đồng bộ (mỗi lần lặp là một lần đồng bộ), và độ hoàn thành công việc cộng tác của đoạn mã song song là $O(n^2/p)$. Nếu ta song song vòng lặp i ,

thuật toán song song thực hiện 1 lần đồng bộ, và độ hoàn thành công việc cộng tác của đoạn mã song song là $O(n^3/p)$. Đối với máy tính đa bộ xử lý truy cập bộ nhớ đồng bộ (UMA multiprocessors), thì vòng lặp song song i sẽ thực hiện nhanh hơn.
Thuật toán song song nhân ma trận cho máy tính đa xử lý truy cập bộ nhớ đồng bộ được trình bày dưới đây.

MATRIX_MULTIPLICATION(UMA MULTIPROCESSORS)

Biến toàn cục:

n,
 $A[0..n-1,0..n-1]$, $B[0..n-1,0..n-1]$, $C[0..n-1,0..n-1]$

Biến cục bộ: i,j,k,t

Begin

```

For all  $P_m$  ( $1 \leq m \leq p$ ) do
    For  $i = m$  to  $n$  step  $p$  do
        for  $j=1$  to  $n$  do
             $t=0$ 
            for  $k=1$  to  $n$  do
                 $t=t+A[i,k]*B[k,j]$ 
            endfor
             $C[i,j]=t$ 
        Endfor
    Endfor
End.

```

Ta hãy tính độ phức tạp của thuật toán trên. Mỗi tiền trình tính n/p hàng của ma trận C. Thời gian cần thiết để tính một hàng là $\Theta(n^2)$. Các tiền trình đồng bộ chính xác một lần, chi phí đồng bộ là $\Theta(p)$. Vì vậy độ phức tạp của thuật toán song song này là $\Theta(n^3/p + p)$. Chú ý rằng, vì chỉ có n hàng, nên có nhiều nhất n tiền trình có thể được thực hiện trong thuật toán này. Nếu ta bỏ qua khả năng tranh chấp bộ nhớ (memory contention), thì tốc độ (speedup) sẽ tiệm cận đến tuyến tính.

Trong thực tế liệu ta có thể bỏ qua thời gian truy cập bộ nhớ?. Điều này có thể được khi thực hiện thuật toán trên máy tính đa bộ xử lý truy cập bộ nhớ đồng bộ. Trên máy tính đa bộ xử lý truy cập bộ nhớ đồng bộ thì khoảng cách của tất cả các ô nhớ trong bộ nhớ toàn cục tới các bộ xử lý là như nhau.

Một phương pháp khác có thể cải tiến thuật toán trên là *nhân ma trận khối* (block matrix multiplication). Giả sử A và B là hai ma trận kích thước $n \times n$ với $n=2k$ thì mỗi ma trận A hoặc B được coi như một khối gồm 4 ma trận con kích thước $k \times k$:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \text{ và } B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

Khi đó, ma trận kết quả C sẽ được xác định như sau:

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

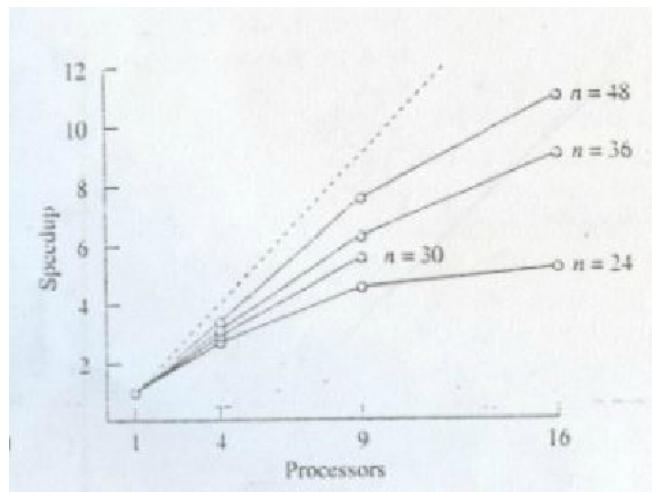
Nếu ta gán các process để thực hiện tính các ma trận khối, thì số lượng phép nhân và cộng và đọc dữ liệu từ bộ nhớ trên mỗi ma trận khối sẽ tăng. Chẳng hạn, có $p=(n/k)^2$ tiến trình, thì phép nhân ma trận được thực hiện bởi việc chia A và B thành p khối kích thước $k \times k$. Mỗi khối $2k^2$ lần đọc bộ nhớ (fetches), k^3 phép cộng, và k^3 phép nhân. Số các thao tác tính toán trên một lần truy xuất bộ nhớ sẽ tăng từ 2, trong thuật toán trước lên $k=n/\sqrt{p}$ trong thuật toán này, đây là một sự cải tiến đáng kể. Dưới đây là một ví dụ về thuật toán nhân ma trận hướng khối.

Block-matrix approach to parallel-matrix multiplication.

$$\begin{array}{c}
 \begin{array}{c} \text{A} \\ \left(\begin{array}{cccc} 1 & 0 & 2 & 3 \\ 4 & -1 & 1 & 5 \\ -2 & -3 & -4 & 2 \\ -1 & 2 & 0 & 0 \end{array} \right) \end{array}
 \quad
 \begin{array}{c} \text{B} \\ \left(\begin{array}{cccc} -1 & 1 & 2 & -3 \\ -5 & -4 & 2 & -2 \\ 3 & -1 & 0 & 2 \\ 1 & 0 & 4 & 5 \end{array} \right) \end{array}
 \quad
 \begin{array}{c} \text{C} \\ \left(\begin{array}{cccc} 8 & -1 & 14 & 16 \\ 9 & 7 & 26 & 17 \\ 7 & 14 & -2 & 14 \\ -9 & -9 & 2 & -1 \end{array} \right) \end{array}
 \end{array}
 =
 \begin{array}{c}
 \begin{array}{c} \text{STEP 1: Compute } C_{i,j} = A_{i,1} B_{1,j} \\ \left(\begin{array}{cc} 1 & 0 \\ 4 & -1 \\ -2 & -3 \\ -1 & 2 \end{array} \right) \left(\begin{array}{cc} -1 & 1 \\ -5 & -4 \\ -1 & 1 \\ -5 & -4 \end{array} \right) = \left(\begin{array}{cc|cc} -1 & 1 & 2 & -3 \\ 1 & 8 & 6 & -10 \\ 17 & 10 & -10 & 12 \\ -9 & -9 & 2 & -1 \end{array} \right) \end{array}
 \end{array}$$

$$\begin{array}{c}
 \begin{array}{c} \text{STEP 2: Compute } C_{i,j} = C_{i,j} + A_{i,2} B_{2,j} \\ \left(\begin{array}{cc} 2 & 3 \\ 1 & 5 \\ -4 & 2 \\ 0 & 0 \end{array} \right) \left(\begin{array}{cc} 3 & -1 \\ 1 & 0 \\ 3 & -1 \\ 1 & 0 \end{array} \right) = \left(\begin{array}{cc|cc} 2 & 3 & 0 & 2 \\ 1 & 5 & 4 & 5 \\ -4 & 2 & 0 & 2 \\ 0 & 0 & 4 & 5 \end{array} \right) = \left(\begin{array}{cc|cc} 8 & -1 & 14 & 16 \\ 9 & 7 & 26 & 17 \\ 7 & 14 & -2 & 14 \\ -9 & -9 & 2 & -1 \end{array} \right) \end{array}
 \end{array}$$

Hình 2.11. Ví dụ minh họa thuật toán nhân ma trận khối.



Hình 2.12. Tốc độ của thuật toán nhân ma trận.

2.3 Các thuật toán sắp xếp song song.

Sắp xếp là một trong những bài toán cơ bản nhất của khoa học máy tính. Trong nhiều bài toán tìm kiếm thông tin yêu cầu tập dữ liệu phải được sắp xếp trước khi tìm kiếm để tăng hiệu quả tìm kiếm. Sắp xếp có ý nghĩa quan trọng đối với các nhà thiết kế thuật toán song song: nó được sử dụng thường xuyên để thực hiện các phép hoán vị dữ liệu tổng quát trên máy tính với các bộ nhớ phân tán. Các thao tác di chuyển dữ liệu này có thể được sử dụng để giải các bài toán trong lý thuyết đồ thị, hình học giải tích, và xử lý ảnh với thời gian tối ưu hoặc cận tối ưu.

Đã có nhiều kết quả trong nghiên cứu phát triển các thuật toán sắp xếp song song mà ta sẽ thảo luận trong các phần sau như: các thuật toán sắp xếp song song trên máy tính bộ xử lý mảng, máy tính đa bộ vi xử lý, và đa máy tính. Tất cả thuật toán này được gọi là sắp xếp trong. Nghĩa là, chúng sắp xếp các mảng có kích thước nhỏ hơn hoặc vừa với bộ nhớ chính. Đồng thời, các thuật toán này sắp xếp bằng cách so sánh từng cặp phần tử.

2.3.1 Sắp xếp bằng liệt kê (enumeration sort) và cận dưới (lower bounds) của sắp xếp song song.

a. Sắp xếp bằng liệt kê:

Bài toán được phát biểu như sau: cho một mảng gồm n phần tử a_0, a_1, \dots, a_{n-1} với thứ tự đã được xác định. Đối với 2 phần tử a_i, a_j thì có các trường hợp sau: $a_i < a_j$, $a_i = a_j$, và $a_i > a_j$. Mục tiêu của sắp xếp là tìm một hoán vị $(t_0, t_1, \dots, t_{n-1})$ sao cho $a_{t_0} \leq a_{t_1}, \dots, \leq a_{t_{n-1}}$.

Dưới đây là thuật toán sắp xếp theo liệt kê:

cuu duong than cong . com

ENUMERATION SORT(SPECIAL CRCW PRAM)

Tham số: n

Biến toàn cục: A[0.. n-1], Position[0..n-1], Sorted[0..n-1] .

Begin

```
    Spawn(Pi,j for all 0 ≤ i,j ≤ n-1)
    For all Pi,j where 0 ≤ i,j ≤ n-1 do
        Position[i]=0
        If a[i] < a[j] or (a[i] =a[j] and i <j) then
            Position[i]=0
        endif
    endfor
    For all Pi,0 where 0 ≤ i ≤ n-1 do
        Sorted[Position[i]] = A[i]
    endfor
End;
```

Một tập gồm n phần tử có thể được sắp xếp trong thời gian $\Theta(\log n)$ với n^2 bộ xử lý trên một máy tính mô hình PRAM CRCW trong đó ghi đồng thời vào cùng một vị trí làm cho tổng các giá trị được gán. Nếu thời gian kích hoạt các bộ xử lý không tính đến thì thời gian thực hiện thuật toán là hằng số.

b. Cận dưới của sắp xếp song song:

Ở trên ta đã chỉ ra rằng sắp xếp có thể được thực hiện trong thời gian là hằng số nếu có đủ bộ xử lý và một mô hình tính toán song song đủ mạnh (như một phiên bản của PRAM). Vậy cận dưới để sắp xếp song song trên một mô hình tính toán song song hợp lý là gì?. Trong phần này ta đi tìm cận dưới của các thuật toán sắp xếp song song dựa trên máy tính với các bộ xử lý được tổ chức theo mạng hình lưới 1 chiều, 2 chiều và mạng hoán vị-di chuyển.

Định lý 2.3-1: Giả sử ta cần sắp xếp n phần tử trên một máy tính mang các bộ xử lý được tổ chức như một mạng hình lưới 1 chiều. Và giả sử rằng trước và sau khi sắp xếp các phần tử được phân phát đều (evenly distributed) với mỗi phần tử trên một bộ xử lý. Thì cận dưới của độ phức tạp về thời gian là $\Theta(n)$.

Chứng minh: Độ rộng phân đôi của một mạng hình lưới một chiều với n nút là 1. Giả sử rằng, các vị trí được sắp xếp của tất cả các phần tử ban đầu trên một bên của mạng phân đôi (bisection). Thì tất cả n phần tử phải đi qua 1 liên kết để sang nửa bên kia. Do tại một thời điểm một liên kết chỉ có thể chứa một phần tử, số bước cần để đổi chỗ (swap) các phần tử qua hai nửa của mạng ít nhất là n. Vì vậy, cận dưới của độ phức tạp của bất kỳ thuật toán sắp xếp nào được thực hiện trên máy tính với các bộ xử lý được tổ chức theo một mạng hình lưới 1 chiều là $\Theta(n)$.

Định lý 2.3-2: Giả sử ta cần sắp xếp n phần tử trên một máy tính mang các bộ xử lý được tổ chức như một mạng hình lưới 2 chiều. Và giả sử rằng trước và sau khi sắp xếp các phần tử được phân phát đều (evenly distributed) với mỗi phần tử trên một bộ xử lý. Thì cận dưới của độ phức tạp về thời gian là $\Theta(\sqrt{n})$.

Chứng minh: Độ rộng phân đôi của một mạng hình lưới 2 chiều với n nút không vượt quá \sqrt{n} . Giả sử rằng, các vị trí được sắp xếp của tất cả các phần tử ban đầu trên một bên của mạng phân đôi (bisection). Thì tất cả n phần tử phải đi qua 1 hoặc không nhiều hơn \sqrt{n} liên kết để sang nửa bên kia. Do tại một thời điểm một liên kết chỉ có thể chứa một phần tử, số bước cần để đổi chỗ (swap) các phần tử qua hai nửa của mạng ít nhất là n/\sqrt{n} . Vì vậy, cận dưới của độ phức tạp của bất kỳ thuật toán sắp xếp nào được thực hiện trên máy tính với các bộ xử lý được tổ chức theo một mạng hình lưới 2 chiều là $\Theta(n/\sqrt{n}) = \Theta(\sqrt{n})$.

Định lý 2.3-3: Giả sử ta cần sắp xếp $n = 2^k$ phần tử trên một máy tính mảng các bộ xử lý được tổ chức theo một mạng hoán vị-di chuyển. Và giả sử rằng trước và sau khi sắp xếp các phần tử được phân phát đều (evenly distributed) với mỗi phần tử trên một bộ xử lý. Thì cận dưới của độ phức tạp về thời gian là $\Theta(\log n)$.

Chứng minh: Giả sử vị trí đã được sắp xếp của một phần tử ban đầu ở nút 0 là nút $n-1$. Chuyển phần tử đó từ nút 0 đến nút $n-1$ cần ít nhất $\log n$ thao tác hoán vị và ít nhất $\log n - 1$ thao tác di chuyển. Vì vậy cận dưới bất kỳ thuật toán sắp xếp nào dựa trên mạng hoán vị di chuyển cũng có độ phức tạp không vượt quá $\Theta(\log n)$.

2.3.2 Sắp xếp song song đổi chỗ chẵn lẻ (odd-even transposition).

Sắp xếp đổi chỗ chẵn lẻ được thiết kế cho máy tính mảng các bộ xử lý trong đó các bộ xử lý được tổ chức trong một mạng hình lưới 1 chiều.

Giả sử rằng $A = (a_0, a_1, \dots, a_{n-1})$ là một tập n phần tử cần được sắp xếp, với n là số chẵn. Mỗi bộ xử lý có hai biến cục bộ t và a , trong đó a là một phần tử của mảng A và t là biến chứa giá trị được lấy từ bộ xử lý bên cạnh.

Thuật toán thực hiện $n/2$ lần lặp, mỗi lần lặp gồm hai pha:

Pha đầu được gọi là đổi chỗ lẻ chẵn (odd-even exchange), giá trị a trên mỗi bộ xử lý có số hiệu lẻ (trừ bộ xử lý thứ $n-1$) được so sánh và hoán vị (nếu cần) với giá trị nằm trên bộ xử lý kế tiếp. Do vậy bộ xử lý có số hiệu nhỏ sẽ lưu giá trị nhỏ hơn. Pha thứ hai được gọi là đổi chỗ chẵn lẻ, trong đó giá trị a trong mỗi bộ xử lý có số hiệu chẵn được so sánh và hoán vị (nếu cần) với giá trị đang được lưu ở bộ xử lý kế tiếp và do đó bộ xử lý có số hiệu nhỏ sẽ lưu giá trị nhỏ hơn. Sau $n/2$ lần lặp thì tất cả các giá trị sẽ được sắp xếp.

ODD-EVEN TRANSPOSITION SORT(ONE DIMENTIONAL MESH)

Tham số: n

Biến toàn cục: i.

Biến cục bộ: a {phần tử cần sắp}

t {phần tử lấy từ bộ xử lý bên cạnh}

Begin

 Spawn(P_i for all $0 \leq i \leq n-1$)

 For $i=1$ to $n/2$ do

 For all P_i where $0 \leq i \leq n-1$ do

 If $j < n$ and odd(j) then /* hoán vị lẻ chẵn */

 t = successor(a)

 successor(a) = max (a,t)

 a = min(a,t)

 endif

 If even(j) then /* hoán vị chẵn lẻ */

 t = successor(a)

 successor(a) = max (a,t)

 a = min(a,t)

 endif

 endfor

 endfor

End.

Ví dụ về thuật toán song song hoán vị chẵn lẻ: <https://www.cuuduongthancong.com>

| Chỉ số | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|--------------------------|-------|-------|-------|-------|-------|---|---|---|
| Mảng ban đầu | G | H | F | D | E | C | B | A |
| Sauk hi hoán vị lẻ chẵn | G | F < H | D < E | B < C | A | | | |
| Sau khi hoán vị chẵn lẻ | F < G | D < H | B < E | A < C | | | | |
| Sau khi hoán vị lẻ chẵn | F | D < G | B < H | A < E | C | | | |
| Sau khi hoán vị chẵn lẻ | D < F | B < G | A < H | C < E | | | | |
| Sau khi hoán vị lẻ chẵn | D | B < F | A < G | C < H | E | | | |
| Sau khi hoán vị chẵn lẻ | B < D | A < F | C | G | E < H | | | |
| Sau khi hoán vị lẻ chẵn | B | A < D | C < F | E < G | H | | | |
| Sau khi hoán vị lchẵn lẻ | A < B | C < D | E < F | G < H | | | | |

Định lý 2.3-4: Độ phức tạp của thuật toán sắp xếp n phần tử trên máy tính mảng các bộ xử lý trong đó các bộ xử lý được tổ chức trong một mạng hình lưới 1 chiều sử dụng sắp xếp hoán vị chẵn lẻ là $\Theta(n)$.

Chứng minh: Chứng minh dựa trên một thực tế là với i lần lặp bên ngoài của vòng lặp for, không phần tử nào di chuyển quá $n-2i$ vị trí so với vị trí cuối cùng (vị trí đã được sắp xếp). Do vậy $n/2$ lần lặp là đủ để sắp xếp các phần tử và độ phức tạp về thời gian là $\Theta(n)$ với n bộ vi xử lý.

Từ định lý 2.3-1 thì thuật toán sắp xếp hoán vị chẵn lẻ là một thuật toán tối ưu.

2.3.3 Sắp xếp song song trộn bitonic (bitonic merge)

Thuật toán sắp xếp song song trộn Bitonic (được đề xuất bởi Batcher, 1968) là thuật toán có độ phức tạp $O(\log^2 n)$. Trước hết, ta tìm hiểu một số khái niệm sẽ sử dụng trong thuật toán.

a. Các mạng so sánh (comparation networks)

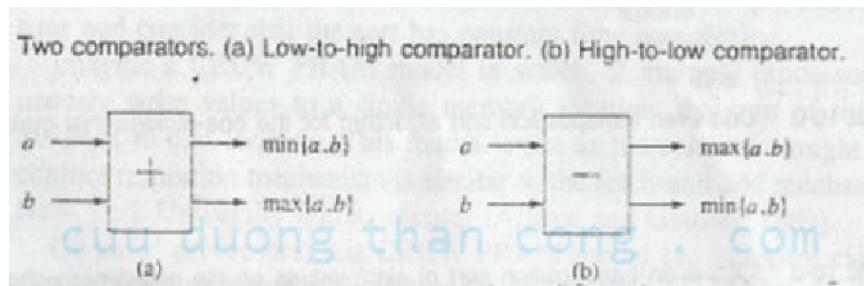
Các mạng sắp xếp (sorting networks) là các mạng so sánh mà chúng luôn so sánh các đầu vào của chúng. Một mạng so sánh bao gồm các dây dẫn (wires) và các bộ so sánh. Một bộ so sánh (comparator) là một thiết bị gồm 2 đầu vào a và b , và 2 đầu ra a' và b' , thực hiện như sau:

$$a' = \min(a, b)$$

$$b' = \max(a, b)$$

So sánh- đổi chỗ (compare-exchange) là hai giá trị đầu vào của bộ so sánh có thể đổi chỗ nếu cần thiết để chúng có thứ tự đúng.

Nếu kể cả thứ tự của đầu ra thì ta có hai loại bộ so sánh: bộ so sánh từ bé đến lớn (low-to-high) và bộ so sánh từ lớn đến bé (high-to-low). Hình dưới minh họa hai bộ so sánh trên:



Hình 2.13. a Bộ so sánh nhỏ đến lớn; b. Bộ so sánh lớn đến bé.

Thời gian thực hiện thao tác so sánh (tính a' và b') là hằng số. Một dây dẫn có thể truyền một giá trị từ nơi này đến nơi khác. Các dây dẫn có thể kết nối đầu ra của bộ so sánh này với đầu vào của bộ so sánh khác. Một mạng sắp xếp gồm n dây dẫn chứa n giá trị đầu vào cần được sắp xếp (a_1, a_2, \dots, a_n), và n dây dẫn đầu ra (b_1, b_2, \dots, b_n) là kết quả của việc sắp xếp n đầu vào.

Hình dưới đây là một minh họa về một mạng so sánh.

Mạng so sánh này gồm 4 dây dẫn chứa 4 đầu vào và 5 bộ so sánh A, B, C, D và E.

b. Dãy bitonic (bitonic sequence)

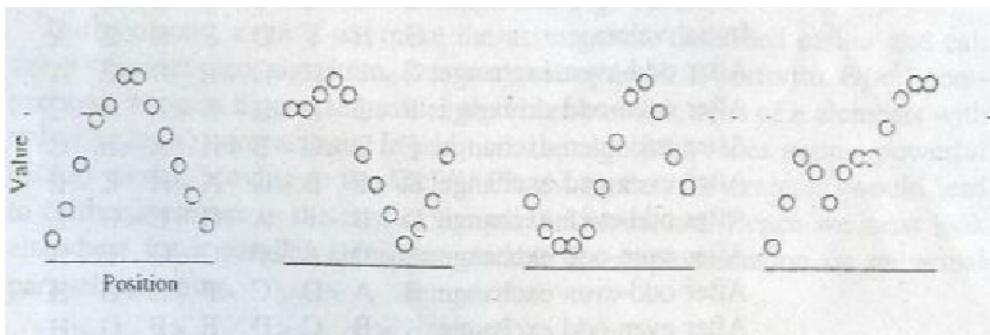
Một dãy bitonic là một dãy a_1, a_2, \dots, a_n sao cho:

1) tồn tại một chỉ số i với $1 \leq i \leq n$ sao cho dãy con a_1, a_2, \dots, a_i là dãy đơn điệu tăng và dãy con a_i, \dots, a_n là dãy đơn điệu giảm, hoặc:

2) sau một số bước thực hiện phép quay (trái hoặc phải) thì 1) thỏa mãn.

Nếu nhìn đồ thị của một dãy bitonic thì nó sẽ gồm một điểm cao nhất (peak) và một điểm thấp nhất (valley).

Dưới đây là một vài ví dụ về dãy bitonic:



Hình 2.14. Ba dãy đầu tiên là các dãy bitonic, dãy cuối cùng không phải là bitonic.

Các dãy: $(1,4,6,8,3,2)$, $(6,9,4,2,3,5)$ và $(9,8,3,2,4,6)$ là các dãy bitonic.

Một bước so sánh- đổi chỗ (compare-exchange) có thể chia một dãy bitonic thành hai dãy bitonic như bồ đề 2.3 dưới đây:

Bố đề 2.3: Nếu n là chẵn, thì $n/2$ bộ so sánh là đủ để chuyển một dãy bitonic gồm n phần tử a_1, a_2, \dots, a_n thành hai dãy bitonic với mỗi dãy có $n/2$ phần tử:

$$\min(a_1, a_{n/2+1}), \min(a_2, a_{n/2+2}), \dots, \min(a_{n/2}, a_n)$$

và

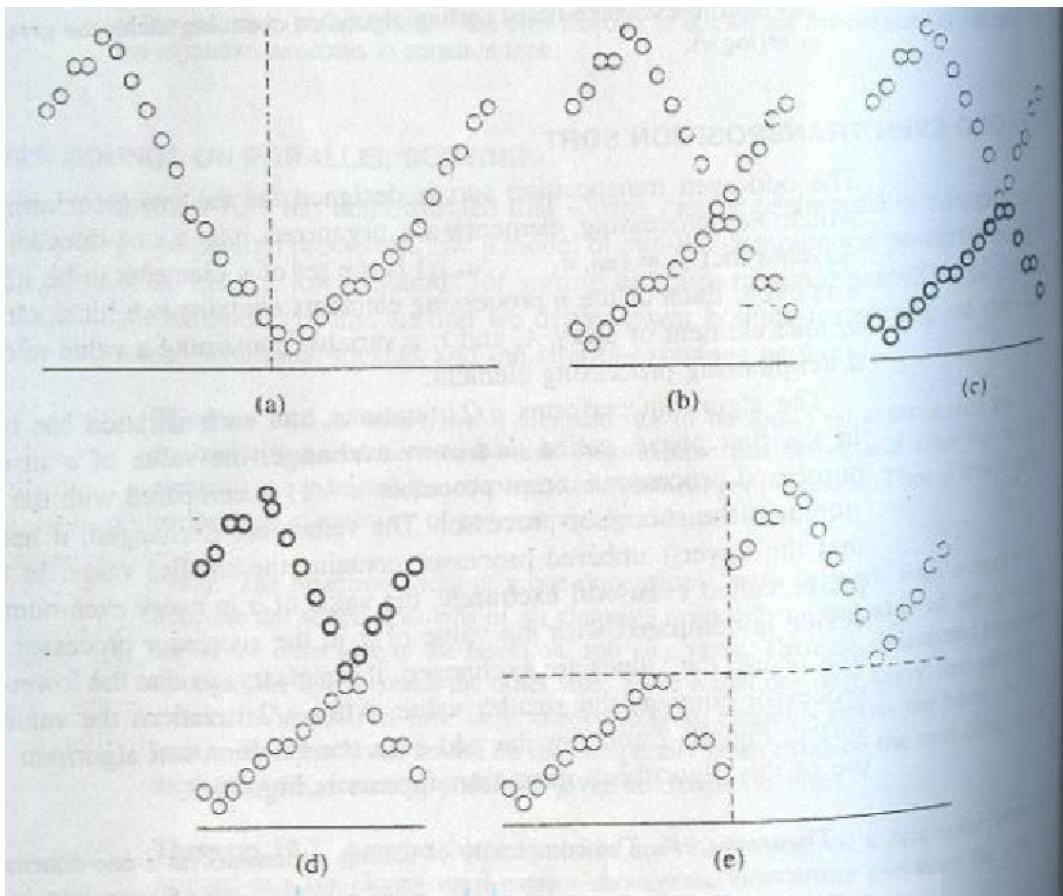
$$\max(a_1, a_{n/2+1}), \max(a_2, a_{n/2+2}), \dots, \max(a_{n/2}, a_n)$$

sao cho không có phần tử nào trong dãy đầu tiên lớn hơn bất kỳ phần tử nào trong dãy thứ 2.

Chứng minh: Hình (a) dưới đây là một dãy bitonic trước khi được chia đôi.

cuu duong than cong . com

cuu duong than cong . com



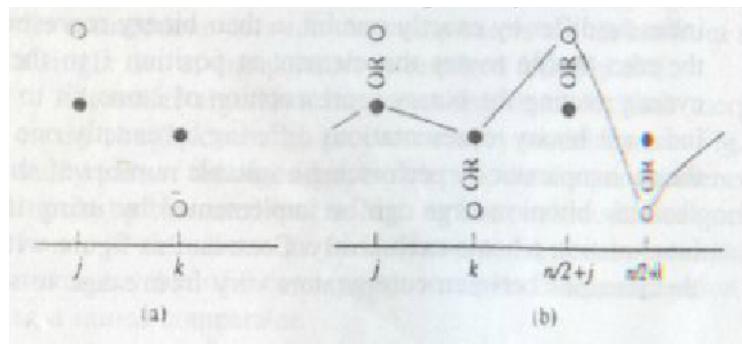
Hình 2.15. (a). dãy bitonic ban đầu; (b) dãy có nửa đầu và nửa cuối giao nhau; (c). các giá trị bé nhất; (d). các giá trị lớn nhất; (e). Dãy đã được biến đổi thành 2 dãy bitonic.

Một dãy bitonic có thể được phân chia để trở thành hai dãy con.

Theo định nghĩa, dãy ban đầu có nhiều nhất một điểm cao nhất và một điểm thấp nhất. Ta sẽ sử dụng $n/2$ bộ so sánh để so sánh mọi phần tử ở nửa đầu của dãy với phần tử tương ứng của dãy thứ 2.

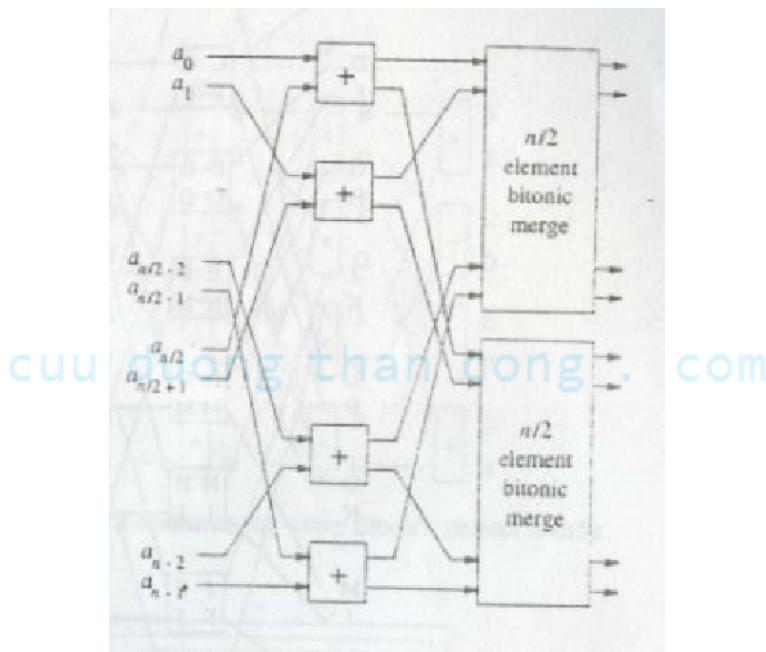
Ta sẽ chứng minh bằng phản chứng để chỉ ra rằng không có phần tử nào trong nửa đầu tiên của dãy lớn hơn phần tử bất kỳ trong nửa còn lại. Nếu tồn tại một phần tử trong nửa đầu tiên của dãy lớn hơn một phần tử nào đó trong nửa còn lại thì giá trị nhỏ nhất được trả về từ một bộ so sánh sẽ lớn hơn một giá trị lớn nhất trả về từ một bộ so sánh khác. Nếu điều này đúng, thì dãy không giao nhau ban đầu sẽ chứa hai đỉnh cao nhất (peak) và hai “đáy” thấp nhất (valley) và dẫn tới dãy ban đầu không phải là dãy bitonic, và dẫn tới mâu thuẫn. Mâu thuẫn cho ta kết luận “không có phần tử nào trong nửa đầu tiên của dãy lớn hơn phần tử bất kỳ trong nửa còn lại” là đúng.

cuu duong than cong . com



Hình 2.15. Minh họa chứng minh bổ đề.

Cho một dãy bitonic, một bước so sánh và đổi chỗ (compare-exchange step) sẽ chia đôi dãy thành 2 nửa dãy bitonic có độ dài bằng nhau..



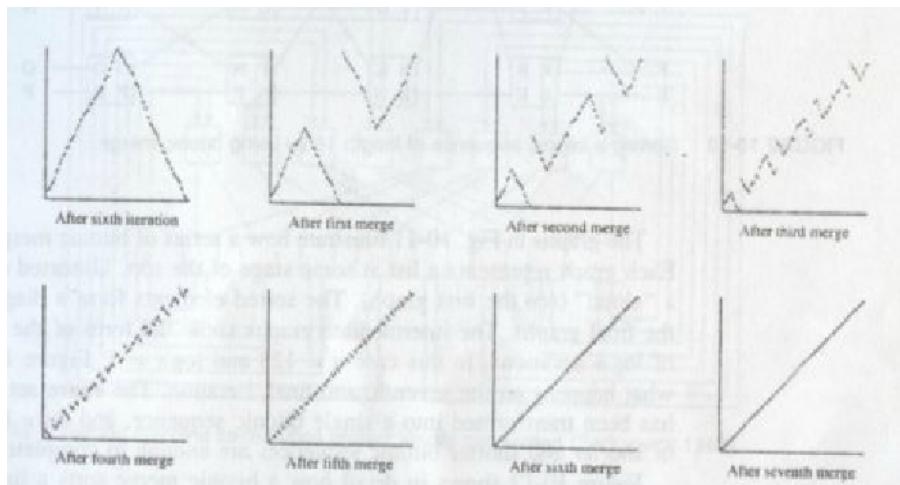
Hình 2.16. Dạng đệ qui tự nhiên của sắp xếp trộn bitonic.

Áp dụng bước này một cách đệ qui sẽ sinh ra dãy được sắp xếp. Nói cách khác, cho trước một dãy bitonic với $n=2^k$ phần tử với $k>0$ thì cần k bước so sánh và đổi chỗ để sắp xếp dãy bitonic ban đầu.

Định lý: Một dãy gồm $n=2^k$ phần tử chưa được sắp xếp có thể được sắp xếp bằng cách sử dụng một mạng so sánh gồm $2^{k-2}k(k+1)$ bộ so sánh với độ phức tạp $\Theta(\log^2 n)$ (Batcher, 1968).

Chứng minh: Sắp xếp trộn bitonic nhận đầu vào là một dãy bitonic và biến đổi nó thành một dãy được sắp xếp. Dãy sắp xếp có thể được xem như một nửa của dãy bitonic. Nếu một dãy bitonic có độ dài 2^m được sắp xếp theo thứ tự tăng dần, trong khi đó dãy liền kề với độ dài 2^m được sắp xếp theo thứ tự giảm dần, thì sau m bước so sánh và đổi chỗ sẽ kết hợp chúng thành dãy 2^{m+1} bitonic.

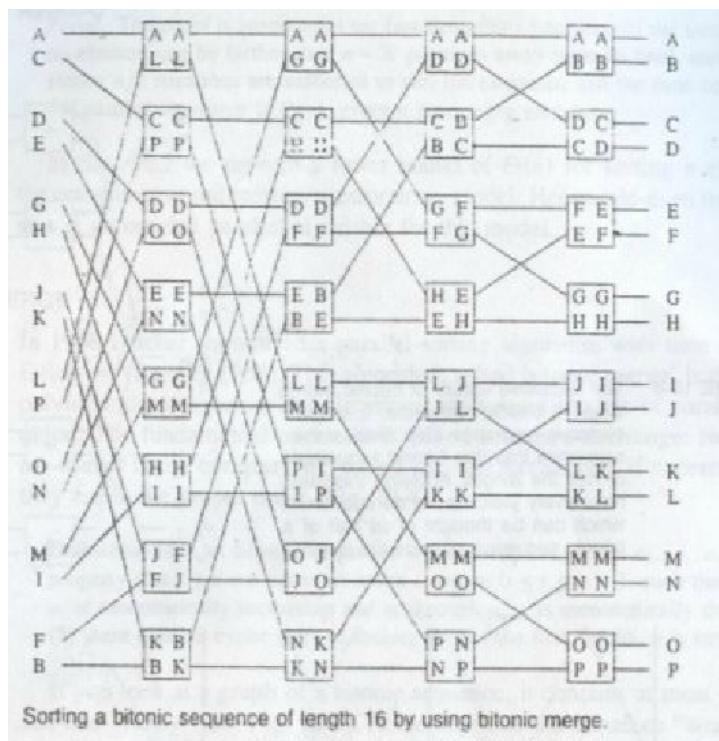
Một dãy gồm n phần tử cần sắp xếp có thể được xem như n dãy còn độ dài 1 hay $n/2$ dãy bitonic độ dài 2. Vì vậy, ta có thể sắp xếp dãy bất kỳ bằng việc trộn liên tiếp chúng để trở thành các dãy bitonic có độ dài lớn hơn. Cho một mảng $n=2^k$ phần tử chưa được sắp xếp, và một mảng so sánh với $k(k+1)/2$ mức. Mỗi mức chứa $n/2 = 2^{k-1}$ bộ so sánh. Thì tổng số cần $2^{k-1} * k(k+1)/2 = 2^{k-2}k(k+1)$ bộ so sánh. Nếu thực hiện song song thì mỗi mức cần thời gian là hằng số. Tổng số mức là $k(k+1)/2 = \log(n) / 2$ nên thuật toán có độ phức tạp $\Theta(\log^2 n)$.



Hình 2.17. Cần 7 bước so sánh và đổi chỗ để sắp xếp trộn một dãy bitonic 128 phần tử.

Dưới đây là một ví dụ minh họa về một dãy các thao tác trộn bitonic để sắp xếp một danh sách gồm 16 phần tử.

cuu duong than cong . com

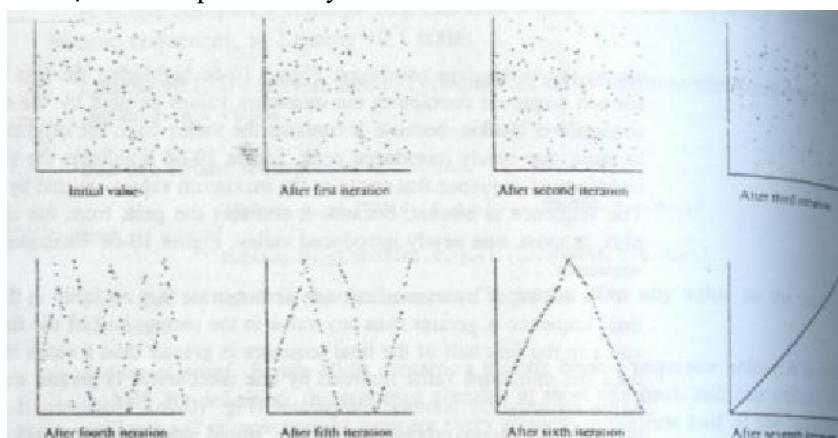


Sorting a bitonic sequence of length 16 by using bitonic merge.

Hình 2.18. Sắp xếp trộn bitonic cho một dãy bitonic gồm 16 phần tử (theo thuật toán Batcher).

Mỗi mức độ thi biểu diễn một bước của quá trình sắp xếp. Các phần tử chưa được sắp xếp ở mức đầu tiên. Sau logn lần lặp thì toàn bộ n phần tử đã được sắp xếp. Trong trường hợp này, n=128 và logn=7.

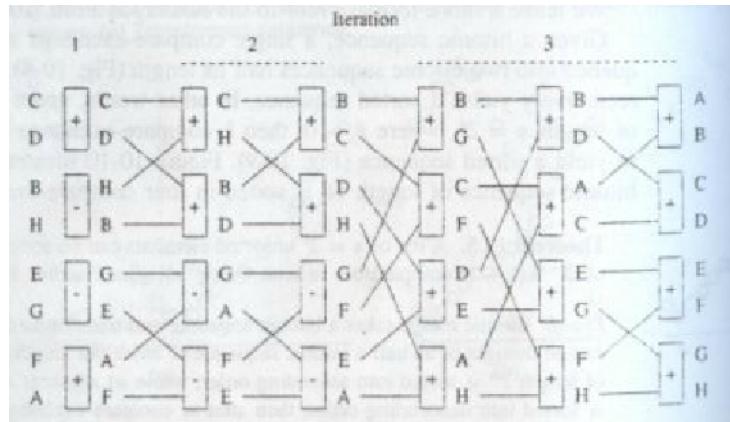
Hình dưới minh họa chi tiết quá trình này:



Hình 2.19. Quá trình sắp xếp 128 phần tử qua 7 bước lặp so sánh và đổi chỗ.

Trong hình 2.19, sau bước thứ 7, toàn bộ 128 phần tử được biến đổi thành một dãy bitonic, và logn lần trộn từ các dãy bitonic con để hoàn thành việc sắp xếp.

Hình dưới đây minh họa thêm một ví dụ nữa về sắp xếp trộn bitonic sử dụng các bộ so sánh bé-đến-lớn hoặc bộ so sánh lớn-đến-bé.

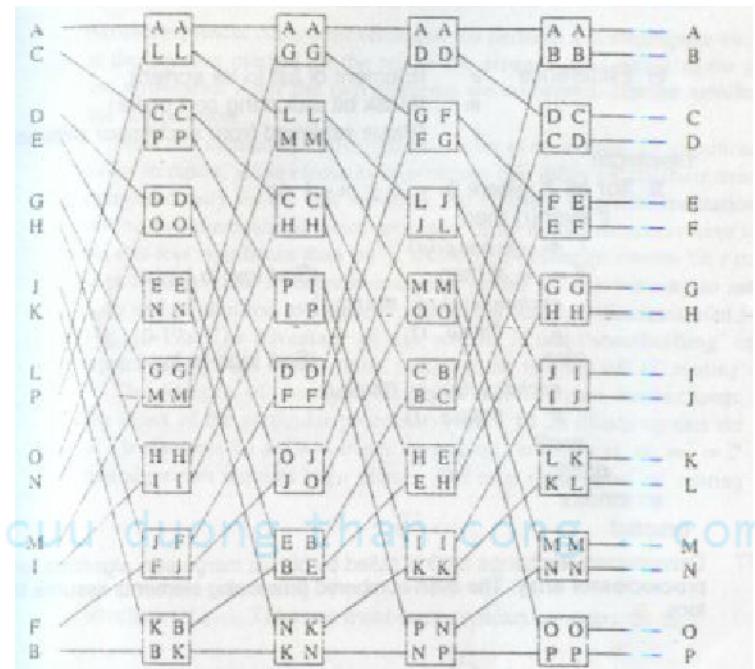


Hình 2.20. Sắp xếp trộn bitonic trên một dãy gồm 8 phần tử.

c. Sắp xếp trộn bitonic trên mạng di chuyển hoán vị (Shuffle-Exchange)

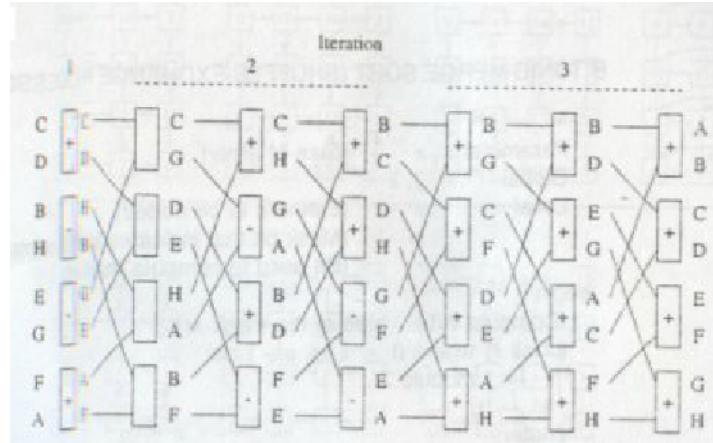
Định lý 2.3: Một dãy chưa được sắp xếp gồm $n=2^k$ phần tử có thể được sắp xếp trong thời gian $\Theta(\log^2 n)$ với một mạng $2^{k-1} [k(k-1)+1]$ bộ so sánh trên mạng di chuyển hoán vị (Stone, 1971). Stone nhận thấy rằng bộ sắp xếp của Batcher luôn luôn so sánh các phần tử với chỉ số khác nhau chính xác một bit trong biểu diễn nhị phân của chúng. Ta nhớ lại rằng một mạng di chuyển hoán hảo (perfect shuffle) định tuyến phần tử ở vị trí i tới vị trí được tìm thấy bởi quay trái một bit trong biểu diễn nhị phân của i . Vì vậy, hai chỉ số với biểu diễn nhị phân khác nhau chính xác một bit có thể di chuyển đến đầu vào của cùng một bộ so sánh bằng việc thực hiện một số bước di chuyển.

Hình dưới đây minh họa trộn bitonic thực hiện trên mạng di chuyển hoán vị.



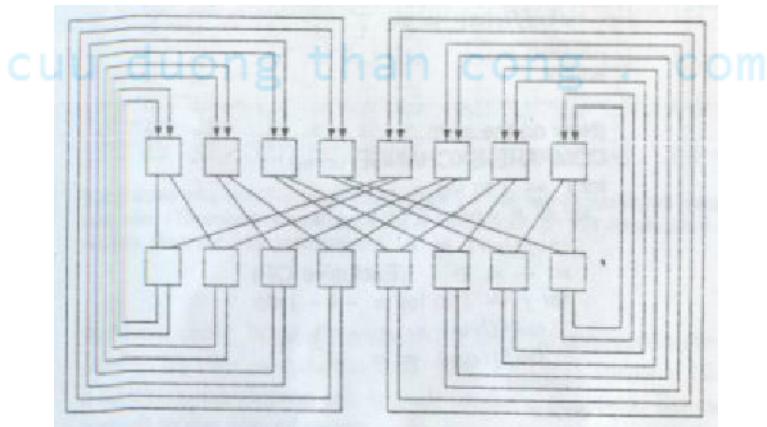
Hình 2.21. Sắp xếp dãy bitonic gồm 16 phần tử trên mạng di chuyển hoán vị (theo thuật toán Stone)

Khác với thuật toán Batcher, các kết nối của thuật toán Stone đa dạng và biến đổi theo từng bước. Một sắp xếp toàn bộ sẽ được hoàn thành với các liên kết di chuyển hoán vị. Cả hai giải thuật đều cần tới k bước để sắp xếp 2^k phần tử, nhưng trong bước thứ i của thuật toán Batcher ở trên cần i bước, cho tổng số $k(k+1)/2$ bước. Thuật toán Stone cần $k(k-1)+1$ bước. Để sắp xếp một dãy gồm 8 phần tử thì cần thêm một thao tác ở lần lặp thứ hai xem hình (10-14).



Hình 2.22. Sắp xếp trộn bitonic cho 8 phần tử trên mạng di chuyển hoán vị.

Một số biến thể của mạng di chuyển hoán vị như mạng di chuyển hoàn hảo của Sedgewick, trong đó các kết nối giữa các bộ so sánh giống nhau trên tất cả các bước. Chỉ cần một tầng chứa các bộ so sánh là đủ để sắp xếp như hình dưới đây:



Hình 2.23. Một máy sắp xếp dựa trên mạng kết nối di chuyển hoàn hảo.

Giải thuật song song sắp xếp bitonic của Batcher được trình bày dưới dạng giả mã như sau:

cuu duong than cong . com

BITONIC-MERGE SORT(SHUFFLE-EXCHANGE PROCESSOR ARRAY)

Tham số: n

Biến toàn cục: j,k

Biến cục bộ: a {phần tử cần sắp}

m,r {bit mặt nạ và bit được dùng để tính bit mặt lật }

Begin

{Tính giá trị của mặt nạ M}

For all P_i where $0 \leq i \leq n-1$ do

$r = i \bmod 2$

$m = r$

endfor

For $i=1$ to $\log n$ do

 For all P_i where $0 \leq i \leq n-1$ do

$m = m \otimes r$ {exclusive OR}

 shuffle(m) = m

 endfor

endfor

{sắp xếp}

COMPARE-EXCHANGE(a,m)

For $k=1$ to $\log n - 1$ do

 For all P_i where $0 \leq i \leq n-1$ do

 shuffle(r) = r

$m = m \otimes r$ {exclusive OR}

 For $j=1$ to $\log n - k - 1$ do

 shuffle(a) = a

 shuffle(m) = m

 endfor

 endfor

 For $j=\log n - k$ to $\log n$ do

 For all P_i where $0 \leq i \leq n-1$ do

 shuffle(a) = a

 shuffle(m) = m

 endfor

 COMPARE-EXCHANGE(a,m)

 endfor

endfor

end.

Một vấn đề mà thuật toán phải giải quyết là xác định cặp phần tử cần so sánh sẽ được sắp xếp từ bé đến lớn hay từ lớn đến bé. Thuật toán Stone sử dụng một véc tơ mặt nạ M để kiểu sắp xếp được thực hiện bởi từng phần tử xử lý. Giá trị 0 tương ứng với bộ so sánh (+) (từ bé đến lớn) và giá trị 1 tương ứng với bộ so sánh (-) (từ lớn đến bé).

Dưới đây là thủ tục COMPARE-EXCHANGE(a,m)

COMPARE-EXCHANGE(a,m)

Tham số: a {phần tử của dãy cần sắp xếp}
 m {bิต mặt nạ chỉ thứ tự sắp xếp }
 t {giá trị lấy từ bộ xử lý kế tiếp}

Begin

```

For all  $P_{ij}$  where  $0 \leq i,j \leq n-1$  do
  If even(i) then
    t = exchange (a)
    If m=0 then
      Exchange(a) = max (a,t)
      a = min (a,t)
    else
      Exchange(a) = max (a,t)
      a = min (a,t)
    endif
  endif
endfor

```

End.

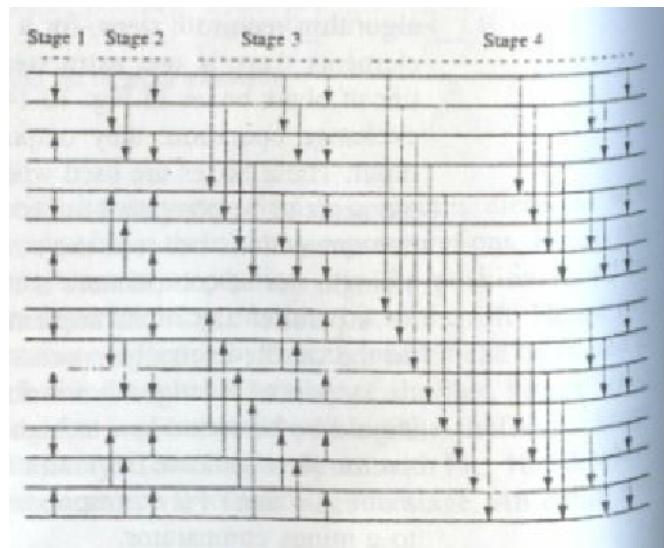
Thuật toán này cần $\log(n) (\log n + 1)/2$ bước so sánh hoán vị và $\log(n-1)$ bước di chuyển của véc tơ A, và $2\log(n-1)$ bước di chuyển của véc tơ M và R. Độ phức tạp của thuật toán là $O(\log^2 n)$.

d. Sắp xếp trộn bitonic trên mạng hình lưới 2 chiều (2-D Mesh network)

Định lý: Tồn tại một thuật toán sắp xếp $n = m^2 = 2^k$ phần tử trên máy tính mảng các bộ xử lý được tổ chức theo mạng hình lưới hai chiều có độ phức tạp $\Theta(\sqrt{n})$.

Chứng minh: Ta có thể sử dụng một biến thể của thuật toán trộn bitonic của Batcher đã được trình bày ở phần trên thực hiện trên máy tính mảng các bộ xử lý được tổ chức theo mạng hình lưới hai chiều. Với $n = 2^k$ phần tử, sắp xếp trộn bitonic gồm k lần lặp, lần lặp thứ i có i bước so sánh- đổi chỗ. Mỗi bước so sánh- đổi chỗ cần hai lần định tuyến dữ liệu: lần định tuyến thứ nhất là mang hai phần tử so sánh với nhau, và lần định tuyến thứ hai là phân phát chúng đúng vị trí. Dưới đây là hình ảnh minh họa thuật toán sắp xếp trộn bitonic với 16 phần tử trên máy tính mảng các bộ xử lý được tổ chức theo mạng hình lưới hai chiều (đề xuất bởi Knuth, 1973).

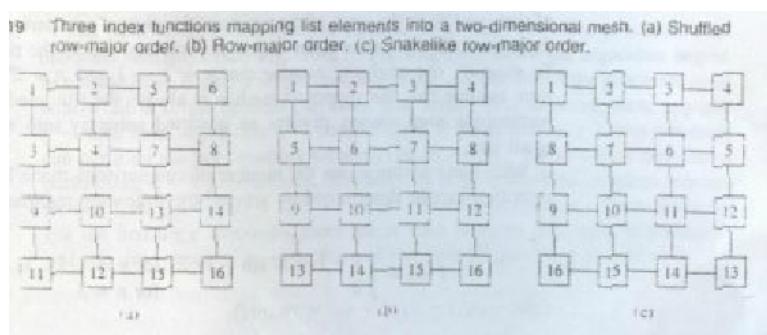
cuu duong than cong . com



Hình 2.24. Mạng sáp xếp trộn bitonic với 16 phần tử trên máy tính mảng các bộ xử lý được tổ chức theo mạng hình lưới hai chiều.

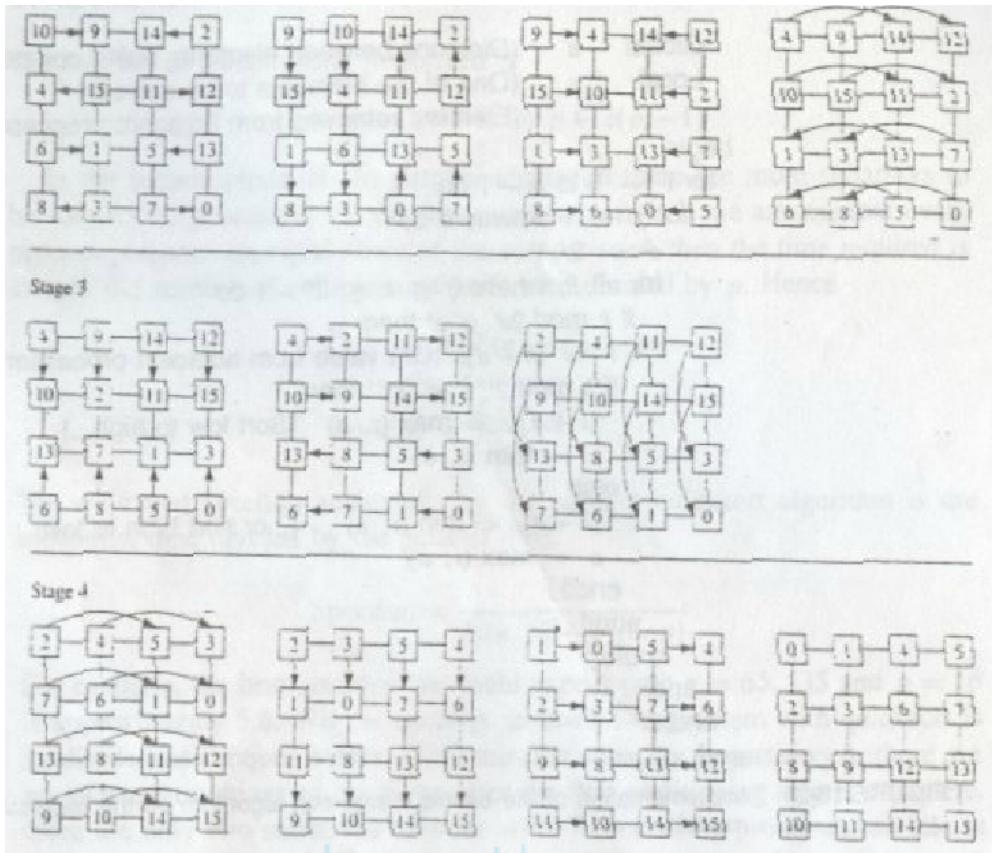
Các mũi tên thể hiện các bộ so sánh- đổi chỗ. Để thực hiện một lần so sánh- đổi chỗ, phần tử ở vị trí đầu và đuôi của mũi tên sẽ được so sánh, phần tử nhỏ hơn sẽ được đưa lên đầu mũi tên và phần tử lớn sẽ được đưa đến đuôi của mũi tên.

Chú ý rằng các phần tử ở các vị trí chỉ khác nhau ở một bít có trọng số nhỏ nhất và chúng được so sánh trong mỗi lần lặp. Ở vòng lặp cuối, các phần tử với trọng số bít lớn nhất khác nhau sẽ được so sánh. Thuật toán hiệu quả nhất khi: bít i có trọng số nhỏ hơn bít j, một bước so sánh- đổi chỗ đối với phần tử chứa bít i yếu cầu dữ liệu ít di chuyển hơn đối với phần tử chứa bít j. Một cách để thỏa mãn điều kiện này là sử dụng cách di chuyển hàng là chính (shuffled row-major addressing schema). Hình dưới minh họa một mạng hình lưới 4×4 có sử dụng các thao tác di chuyển trên hình vuông con của mạng hình lưới. Điều này làm giảm đi các thao tác di chuyển giữa các bộ xử lý.



Hình 2.25. Các hàm ánh xạ các phần tử cần sắp xếp vào mạng hình lưới 2 chiều.

Hướng di chuyển của tại mỗi bước so sánh- trao đổi phụ thuộc vào chỉ số của từng bộ xử lý. Hình dưới đây mô tả sắp xếp 16 phần tử trên mạng hình lưới kích thước 4×4 . Trong trường hợp tổng quát, để sắp xếp $n=m^2=2^k$ phần tử bằng thuật toán này cần logn pha.



Hình 2.26 Các bước ánh xạ các phần tử cần sắp xếp vào mạng hình lưới 2 chiều.

Tổng số các thao tác di chuyển cần:

$$\sum_{i=1}^{\log n} \sum_{j=1}^i 2^{(i-j-1)/2}$$

Kết quả trên bằng $\Theta(\sqrt{n})$. Tổng số các bước so sánh là $\sum_{i=1}^{\log n} i = \Theta(\log^2 n)$

Vì vậy, trong trường hợp xấu nhất, sắp xếp trộn bitonic có độ phức tạp $\Theta(\sqrt{n})$.

d. Sắp xếp trộn bitonic trên mạng siêu khối (Hypercube)

Sắp xếp trộn bitonic luôn luôn so sánh mà các chỉ số khác nhau đúng một bít. Ta nhớ lại rằng trong mạng siêu khối thì các bộ vi xử lý có chỉ số khác nhau đúng một bít được kết nối với nhau. Chính vì vậy, ta có thể dễ dàng trộn bitonic trên mô hình này với các bộ vi xử lý thay thế các bộ so sánh. Thay vì từng cặp phần tử thao tác trên các bộ so sánh, các bộ xử lý cạnh nhau sẽ so sánh và trao đổi dữ liệu cho nhau.

Dưới đây là giả mã của thuật toán sắp xếp $n=2^m$ phần tử trên mạng siêu khối.