TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG & TIN HỌC

BÀI GIẢNG

TÍNH TOÁN SONG SONG

Giảng viên: Đoàn Duy Trung
Bộ môn : Toán Tin

BÀI 3. NGÔN NGỮ LẬP TRÌNH TÍNH TOÁN SONG SONG & NGÔN NGỮ THUẬT TOÁN

NỘI DUNG BÀI HỌC

- Ngôn ngữ lập trình song song
- Mô hình lập trình song song
- Ngôn ngữ giả thuật toán
- Đánh giá độ phức tạp của thuật toán
- Bài toán ví dụ

1. NGÔN NGỮ LẬP TRÌNH SONG SONG

1.1.Ngôn ngữ lập trình song song

- Một tiến trình có thể tương tác với tiến trình khác khi:
  - Cùng 1 thời điểm có một số tiến trình muốn cập nhật vào một biến chia sẻ hoặc truy cập vào tài nguyên dùng chung.
  - Một tiến trình cùng kết hợp để thực hiện một số phép toán trên cơ sở quan sát hành động của nhau.

1.1. Ngôn ngữ lập trình song song

- Hướng giải quyết vấn đề:
  - Tất cả tiến trình sử dụng cấu trúc chung để cập nhật và trao đổi với nhau
  - Đồng bộ tiến trình bằng cách sử dụng hàm wait() và pass().
- Hai cách để phát triển chương trình song song:
  - Mở rộng ngôn ngữ lập trình sẵn có
  - Xây dựng ngôn ngữ mới hỗ trợ lập trình song song

1.2.Một số ngôn ngữ lập trình song song

- Fortran 90 (Formula Translator/Translation):
  - Ngôn ngữ lập trình song song theo dữ liệu được nâng cấp từ Fortran 77.
  - Hỗ trợ: con trỏ, kiểu dữ liệu, trừu tượng, khả năng tính toán mảng…
  - Dựa vào mô hình PRAM: CPU, CU, ALU, đơn vị số học vecto, bộ nhớ chia sẻ.

1.2.Một số ngôn ngữ lập trình song song

- Fortran 90:
  - Khai báo tổng quát có dạng:
    - Type [(kind)] [, attribute]….:: Variable – List
      - Type: Kiểu cơ sở hoặc định nghĩa bởi NSD
      - Kind: Định nghĩa về kiểu
      - Attribute: danh sách thuộc tính của Fortran định nghĩa đặc tính của biến
      - :: phần tử phân cách giữa kiểu và danh sách
  - Ví dụ:
    - CHARACTER (LEN=10) :: s1
    - INTEGER, DIMENSON (1:10) :: SA

1.2. Một số ngôn ngữ lập trình song song

- OCCAM:
  - Được phát triển bởi Inmos Company vào năm 1988
  - Thiết kế cài đặt chip gọi là Transputer
  - Thường nhiều tiến trình và ánh xạ sang một số transputer bất kỳ để thực hiện song song và trao đổi dữ liệu thông qua các kênh vào ra.
  - Số lượng transputer có thể tăng hoặc giảm

1.2.Một số ngôn ngữ lập trình song song

- OCCAM:
  - Là ngôn ngữ lập trình bậc cao sử dụng lập trình cho những hệ thống gồm nhiều máy tính kết nối với nhau hoặc các hệ phân tán.
  - Thiếu một số đặc tính: hỗ trợ cơ chế đệ quy, kiểu dữ liệu, con trỏ.
  - Mỗi câu lệnh khai báo như một tiến trình

1.2.Một số ngôn ngữ lập trình song song

- OCCAM:
  - Ba tiến trình cơ bản:
    - Tiến trình gán: Gán giá trị cho các đối tượng
    - Tiến trình nhập dữ liệu: Nhận dữ liệu từ các kênh vào
    - Tiến trình ra: Gửi dữ liệu ra các kênh
  - Ba cấu trúc điều khiển:
    - SEQ (Sequence)
    - PAR (Parallel)
    - ALT (Alternative)

1.2.Một số ngôn ngữ lập trình song song

- Máy ảo PVM (Parallel Virtual Machine):
  - Máy tính nối mạng trong môi trường UNIX
  - Cho phép tuyển tập các trạm máy tính kết nối với nhau để xử lý song song
  - Thực hiện theo mô hình truyền thông báo tường minh:
    - Hỗ trợ sự kết nối không thuần nhất
    - Hỗ trợ đa bộ xử lý
    - Tính toán dựa trên tiến trình
    - Thay đổi cấu hình theo yêu cầu

1.2.Một số ngôn ngữ lập tình song song

- Máy ảo PVM:
  - PVM xử lý tất cả các vấn đề định tuyến, truyền thông báo, chuyển đổi dữ liệu, lập lịch trong mạng máy tính
  - Gồm 2 thành phần chính:
    - Khối gồm PVMD hoặc PVM3: đặt thường trú trên tất cả các máy tính để tạo máy ảo
    - Thư viện các chương trình con giao diện của PVM
  - Máy tính hỗ trợ PVM: BBN butterfly TC200, Intel Paragon…

2. MÔ HÌNH LẬP TRÌNH SONG SONG

2.Mô hình lập trình song song

- Một số mô hình lập trình song song:
  - Chia sẻ bộ nhớ
  - Truyền thông
  - Dữ liệu song song
  - Hỗn hợp

2.1.Tổng quan

- Các mô hình lập trình song song tồn tại như một sự trừu tượng hóa trên phần cứng và kiến trúc bộ nhớ
- Những mô hình này không thể cho ra một loại máy tính hay kiến trúc bộ nhớ cụ thể
- Thực tế bất kỳ mô hình nào trong số những mô hình này cũng có thể hiện thực trên bất kỳ phần cứng nằm bên dưới

2.1.Tổng quan

- Ví dụ:
  - Mô hình Shared Memory Model trên máy tính Distributed Memory: sử dụng hệ thống shared Memory KSR (Kendall Square Research):
    - Bộ nhớ máy tính phân phối vật lý qua các máy tính nối mạng
    - Người dùng sử dụng như bộ nhớ chia sẻ

2.2.Mô hình chia sẻ bộ nhớ

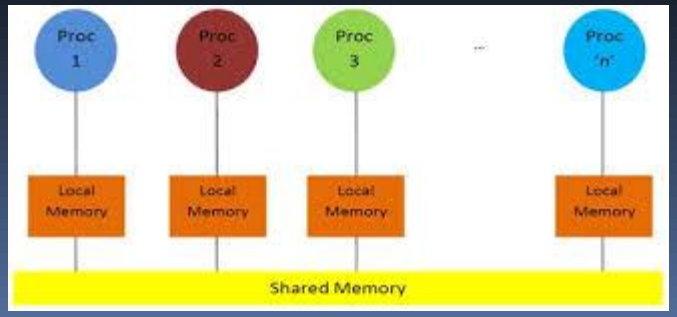
- Đặc điểm:
  - Tác vụ chia sẻ nhau một không gian địa chỉ chung, ở đó các tác vụ đọc ghi không đồng bộ.
  - Sử dụng các cơ chế lock/ semaphores (cờ hiệu) kiểm soát (điều khiển) truy cập vào bộ nhớ chia sẻ.
  - Lợi thế:
    - Khái niệm "sở hữu" dữ liệu đang thiếu do đó việc lập trình đơn giản.
  - Bất lợi:
    - Khó hiểu và khó quản lý dữ liệu
    - Khó kiểm soát dữ liệu cục bộ

2.2.1.Chia sẻ bộ nhớ dựa vào tiến trình

- Đặc điểm:
  - Tạo lập và hủy bỏ tiến trình
    - Lệnh tạo tiến trình:
      - id = create_process (N);
        - Tạo ra N+1 tiến trình
      - Sau khi công việc thực hiện xong, sử dụng tiến trình chủ thực hiện việc tuần tự:
        - join_process(N,id)
    - Các tiến trình trong UNIX sử dụng như các đơn vị tính toán độc lập

2.2.2.Chia sẻ bộ nhớ dựa vào luồng

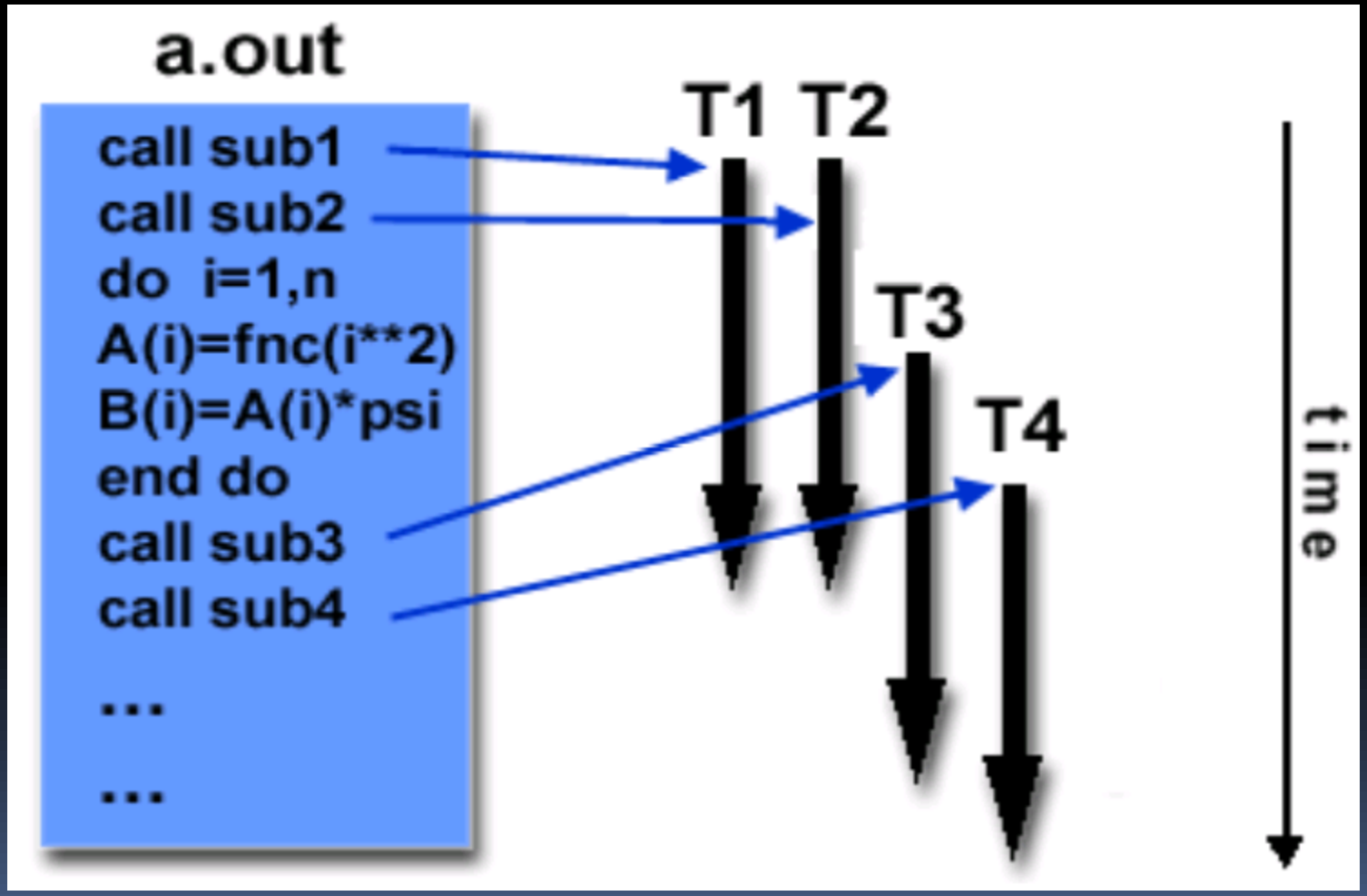
- Khái niệm về luồng (Threads):
  - Là quá trình riêng lẻ thực hiện đồng thời trong các hệ điều hành đa nhiệm.
  - Thiết kế chương trình tách ra nhiều luồng và xử lý cùng một lúc.
  - Hệ thống có 1 BXL thì các luồng sẽ thực hiện luân phiên nhau
  - Không có luồng ưu tiên, các luồng lập lịch để thực hiện



2.2.2.Chia sẻ bộ nhớ dựa vào luồng

- Lợi ích của luồng (Threads):
  - Loại trừ sự cần thiết để cho hệ điều hành liên tục tải thông tin vào/ra bộ nhớ.
  - Giảm phát chi phí phát sinh đối với bộ nhớ và thời gian tạo thông tin trong bộ nhớ.
  - Mỗi luồng tương tác với phần riêng của hệ thống

2.2.2.Chia sẻ bộ nhớ dựa vào luồng



2.2.2.Chia sẻ bộ nhớ dựa vào luồng

- Đặc điểm:
  - Công việc của luồng được mô tả như một chương trình con trong chương trình chính.
  - Một luồng bất kỳ có thể thực hiện chương trình con đồng thời với các luồng khác.
  - Các luồng giao tiếp với nhau thông qua bộ nhớ toàn cục
  - Các luồng liên kết với kiến trúc bộ nhớ chia sẻ và điều hành.

2.2.2.Chia sẻ bộ nhớ dựa vào luồng

- Dưới góc nhìn lập trình:
  - Thư viện của chương trình con được gọi từ bên trong mã nguồn song song.
  - Một tập chỉ thị nhúng trong trình biên dịch mã nguồn song song hoặc tuần tự
- Chuẩn hóa 2 bổ sung của luồng:
  - POSIX Threads
  - OpenMP
- Hệ điều hành hỗ trợ đa luồng: SUN Solaris, Windows NT, OS/2…

2.2.2.Chia sẻ bộ nhớ dựa vào luồng

- POSIX Threads:
  - Dựa vào thư viện đòi hỏi viết mã song song
  - Xác định tiêu chuẩn IEEE POSIX 1003.1c (1995)
  - Sử dụng ngôn ngữ C
  - Gọi là PThreads
  - Pthreads được cung cấp bởi các hãng phần cứng

2.2.2.Chia sẻ bộ nhớ dựa vào luồng

- OpenMP:
  - Dựa vào trình biên dịch; có thể sử dụng mã tuần tự
  - OpenMP phát triển trên: Fortran và C/C++ API
  - Hỗ trợ nền tảng Windows NT & UNIX
  - Dễ dàng sử dụng và đơn giản tăng khả năng tính toán song song.

2.3.Mô hình truyền thông (Message Passing Model)

- Đặc điểm:
  - Tập các tác vụ có sử dụng bộ nhớ cục bộ riêng của chúng trong thời gian tính toán.
  - Nhiều công việc có thể nằm trên cùng một máy hoặc qua một số máy khác.
  - Các tác vụ trao đổi dữ liệu thông qua truyền thông bằng cách gửi nhận thông báo
  - Chuyển giao dữ liệu thường đòi hỏi hợp tác thực hiện bởi quá trình.

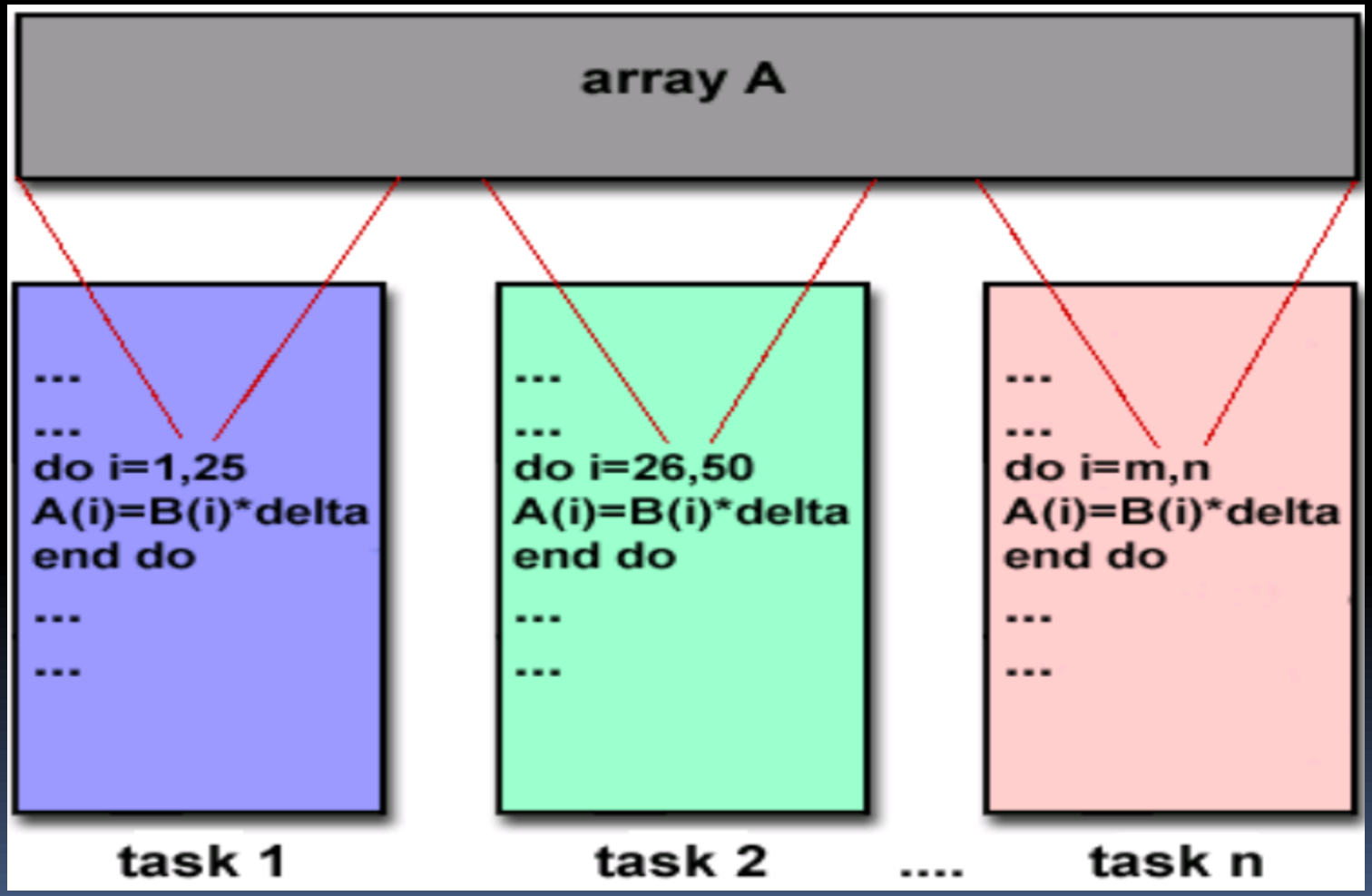
2.3.Mô hình Message Passing Model

- Đặc điểm:
  - 1992: Forum MPI thành lập với mục tiêu thiết lập giao diện chuẩn cho việc thực hiện truyền thông
  - 1994: phát hành MPI-1
  - 1996: phát hành MPI-2

2.4.Mô hình dữ liệu song song (Data Parallel)

- Đặc điểm:
  - Công việc song song tập trung vào việc thực hiện các thao tác trên một tập dữ liệu (array, vecto)
  - Tập dữ liệu thường tổ chức thành một cấu trúc chung như bảng, mảng hoặc khối không giao nhau.
  - Tập các tác vụ cùng làm công việc trên cùng một cấu trúc dữ liệu giống nhau
  - Mỗi tác vụ hoạt động trên một phân vùng khác nhau của cùng cấu trúc dữ liệu.

2.4.Mô hình Data Parallel



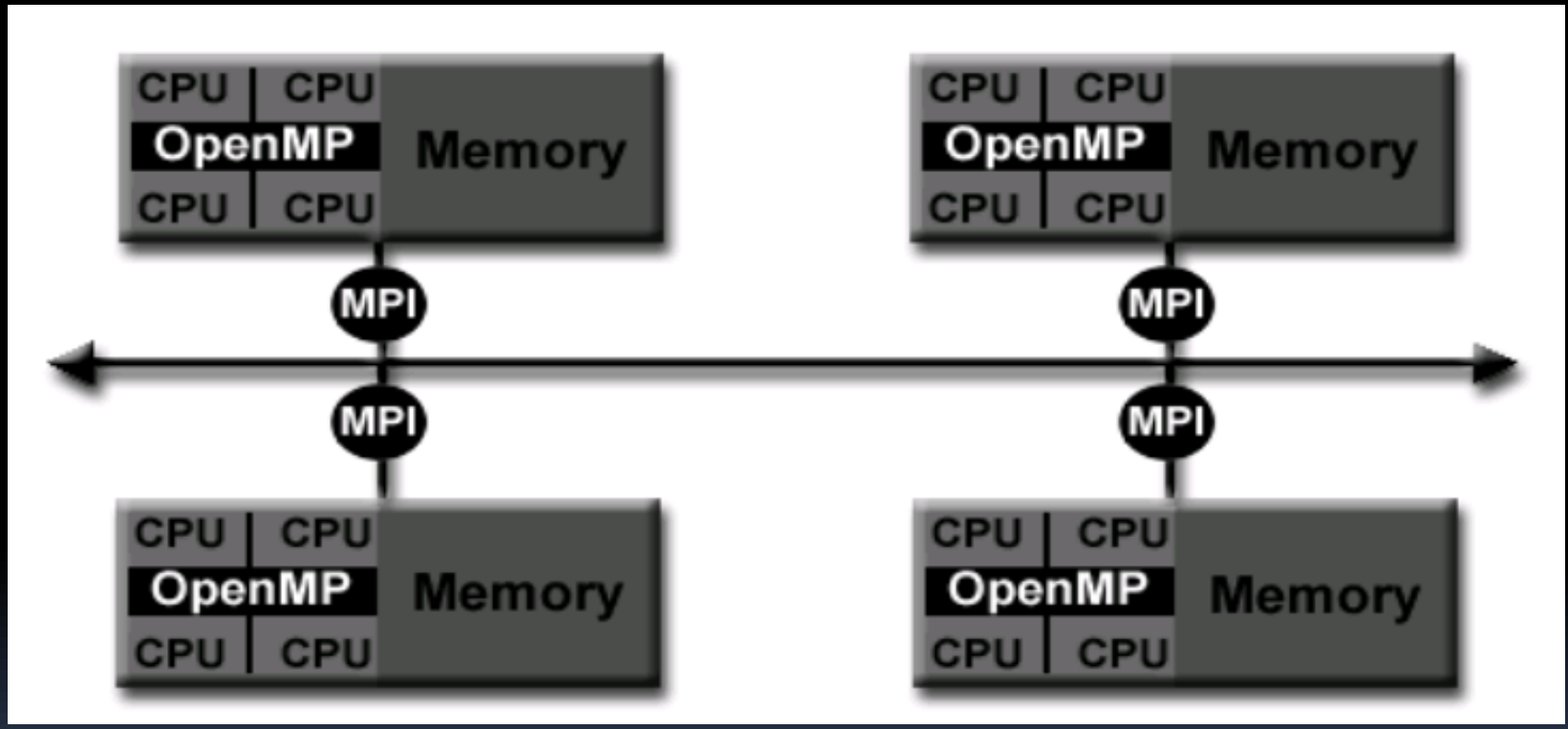
2.4.Mô hình Data Parallel

- High Performance Fortran (HPF): Phần mở rộng của Fortran 90 để hỗ trợ lập trình dữ liệu song song:
  - Mở rộng từ Fortran 90 để hỗ trợ lập trình song song dữ liệu
  - Có đầy đủ tính năng của Fortran 90
  - Các chỉ thị để trình biên dịch biết các phân phối dữ liệu được thêm vào.
  - Nâng cao việc tối ưu khi sinh ra mã máy
  - Cấu trúc song song dữ liệu được thêm vào

2.5.Mô hình hỗn hợp (Hybrid Model)

- Đặc điểm:
  - Sự kết hợp của hai hay nhiều mô hình lập trình song song
  - Ví dụ: MPI + POSIX Threads (OpenMP)
    - Thread thực hiện tính toán dùng dữ liệu địa phương của node tính toán
    - Khi truyền dữ liệu giữa các tiến trình trên những node khác nhau dùng MPI
  - Tăng môi trường phần cứng chung cho các máy SMP được kết mạng.

2.5.Mô hình hỗn hợp (Hybrid Model)

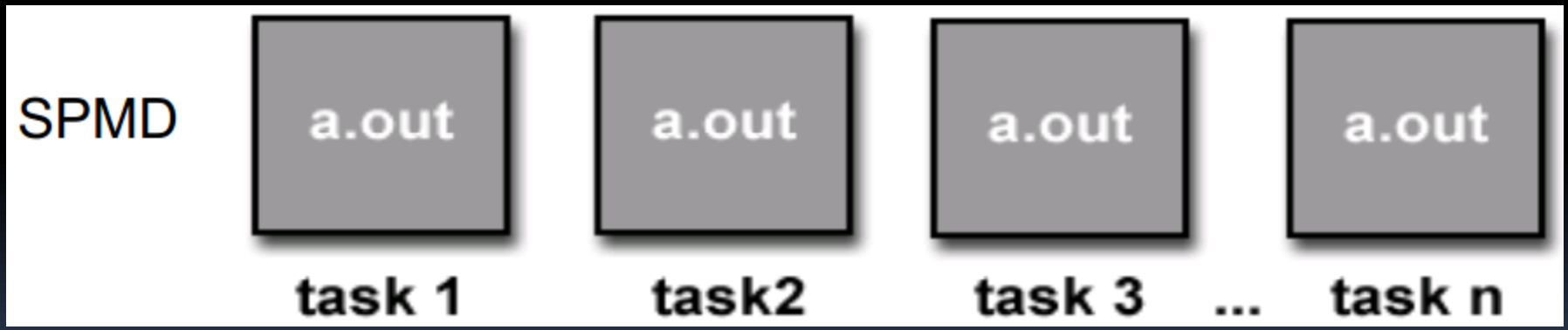


2.5.Mô hình hỗn hợp (Hybrid Model)

- Đơn chương trình – đa dữ liệu (SPMD):
  - Là mô hình lập trình bậc cao c
  - Xây dựng dựa trên một tổ hợp nào đó của các mô hình lập trình song song
  - Một đơn chương trình được thực hiện đồng thời bởi tất cả các tác vụ.
  - Các tác vụ có thể cùng thực hiện các chỉ thị như nhau hay khác nhau trong cùng một chương trình

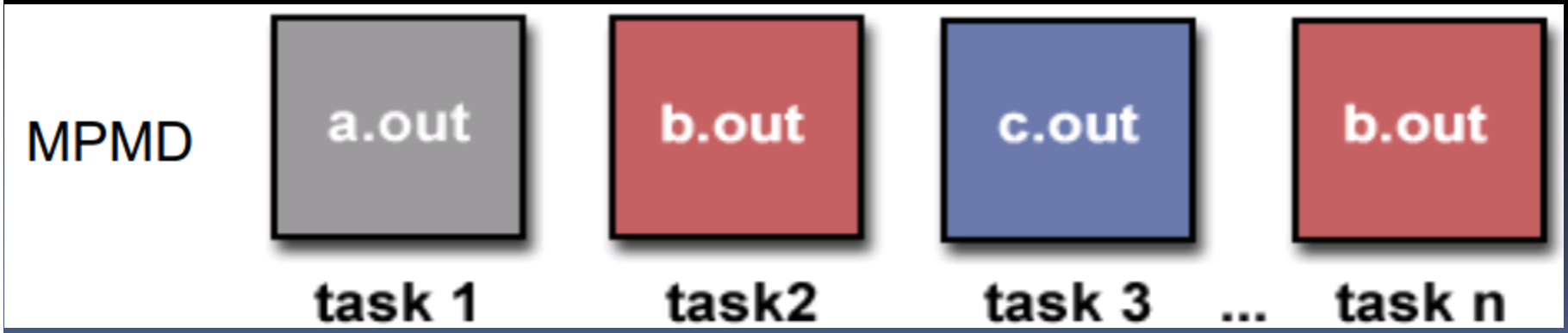
2.5.Mô hình hỗn hợp (Hybrid Model)

- Đơn chương trình – đa dữ liệu (SPMD):
  - Cho phép các tác vụ khác nhau có thể rẽ nhánh
  - Tất cả các tác vụ có thể sử dụng dữ liệu khác nhau



2.5.Mô hình hỗn hợp (Hybrid Model)

- Đa chương trình – Đa dữ liệu (MPMD):
  - Là mô hình lập trình bậc cao dựa trên các mô hình lập trình song song
  - Có nhiều tệp đích để thực thi
  - Các tác vụ có thể sử dụng dữ liệu giống nhau hoặc khác nhau



3. NGÔN NGỮ GIẢ THUẬT TOÁN

3.1.Khái niệm

- Giả ngôn ngữ:
  - Không phải là một ngôn ngữ lập trình cụ thể.
  - Hỗ trợ đầy đủ các khía cạnh của một ngôn ngữ lập trình
  - Không có khái niệm biên dịch
  - Thực hiện trên máy PRAM
- Không quá chặt chẽ về cú pháp câu lệnh, mà mục đích chính là diễn giải ý tưởng thực hiện của thuật toán.

3.2.Hệ thống ký hiệu và cú pháp

- Khai báo biến:
  - Cú pháp: var_name: type_data.
- Phép gán:
  - Cú pháp: var_name = expression.
  - Nếu vế phải là giá trị (value) thì phép gán dẫn đến việcghi dữ liệu vào bộ nhớ.
  - Nếu vế phải là một biến khác thì phép gán thực hiện cả đọc và ghi dữ liệu.

3.2.Hệ thống ký hiệu và cú pháp

- Điều kiện và rẽ nhánh:
  - IF cond. THEN do_sth ENDIF
  - IF cond. THEN do_sth1 ELSE do_sth2 ENDIF.
  - Các từ khóa ELSEIF,… cũng được dùng.
  - Mục đích sử dụng:
    - Điều khiển các BXL thông qua chỉ số của nó.
    - Thực hiện các công việc với điều kiện dữ liệu khác nhau.

3.2.Hệ thống ký hiệu và cú pháp

- Lặp tuần tự:
  - Thường làm việc với dữ liệu kiểu mảng.
  - Các cú pháp tuần tự đã học: FOR, WHILE, DO

```
FOR biến_chạy = khởi_điểm TO kết_thúc STEP bước_nhảy
        … Thực hiện các công việc …
END FOR
```

```
WHILE điều_kiện DO
        …. Thực hiện các công việc….
END WHILE.
```

3.2.Hệ thống ký hiệu và cú pháp

- Lặp song song:
  - Điều khiển các BXL thực hiện song song thông qua chỉ số.
  - Cũng có các cú pháp như tuần tự, nhưng có thêm từ khóa Parallel.

```
FOR index  =  1 TO N DO IN PARALLEL
        ….. các bộ xử lý thực hiện các công việc song song …
END PARALLEL
```

```
FOR index of S DO IN PARALLEL
                … các công việc song song …
END PARALLEL.
```

4. ĐÁNH GIÁ THUẬT GIẢI SONG SONG

4.1.Đánh giá độ phức tạp

- Hiệu quả một thuật toán song song phụ thuộc vào 3 yếu tố:
  - Thời gian thực hiện.
  - Số bộ xử lý tham gia.
  - Kiến trúc siêu máy tính
- Trong đó:
  - Số bộ xử lý tham gia và kiến trúc siêu máy tính có thể xác định được cụ thể.
  - Thời gian thực hiện phụ thuộc vào thuật toán làm việc

4.1.Đánh giá độ phức tạp

- Khi sử dụng vòng lặp cho các máy song song cùng chạy thì trong 1 đơn vị thời gian chỉ cần tính như 1 phép toán.
- Trong cách tính số phép toán thực hiện ta chỉ quan tâm đến các phép toán tích cực, tức là các phép toán mà có số lần sử dụng nhiều nhất so với các phép toán khác.
- Thời gian tính của thuật toán là 1 hàm phụ thuộc vào kích thước của dữ liệu đầu vào (nếu coi N là số đơn vị dữ liệu thì thời gian tính là hàm của N)

4.1. Đánh giá độ phức tạp

- Để đánh giá cận của độ phức tạp, chúng ta sử dụng các ký hiệu so sánh tiệm cận như sau:
  - T(n)=O(f(n)) nếu tìm được các số dương c và m sao cho T(n) < c*f(n) với mọi giá trị n > m.
  - T(n)=Ω(f(n)) nếu tìm được các số dương c và m sao cho T(n) > c*f(n) với mọi giá trị n > m.
  - T(n)=∂(f(n)) nếu tìm được các số dương c1, c2 và m sao c1*f(n) < T(n) < c2*f(n) với mọi n > m.
- Thông thường chúng ta thường quan tâm đến đánh giá cận trên O(n)

4.1..Đánh giá độ phức tạp

- Ta cũng sử dụng các quy tắc để xác định thời gian tính cho 1 thuật toán như:
  - Quy tắc cộng: Nếu 2 đoạn chương trình P1, P2 thực hiện tuần tự kế tiếp nhau, và tương ứng với mỗi đoạn có 1 thời gian tính xác định bởi O(f1(n)) và O(f2(n)) thì thời gian thực hiện cả chương trình là O(f1(n)+f2(n)).
  - Quy tắc nhân: Tương tự như trên, nếu 2 đoạn chương trình P1, P2 được lồng vào nhau, khi đó thời gian tính sẽ là O(f1(n) * f2(n)).

4.1.Đánh giá độ phức tạp

- Trong thuật toán tuần tự, thuật toán tốt có độ phức tạp dạng đa thức đối với kích thước dữ liệu đầu vào (P algorithms).
- Trong thuật toán song song, thuật toán tốt là thuật toán:
  - Số BXL là hàm đa thức đối với kích thước dữ liệu đầu vào.
  - Độ phức tạp là hàm đa thức theo logarithm đối với kích thước dữ liệu đầu vào.

4.2.Speedup

- Thuật giải tuần tự với độ phức tạp theo thời gian là $T_s(n)$.
- Thuật giải song song trên P bộ xử lý có độ phức tạp là $T_p(n)$.
- Khi đó speedup $S_p(n)$ được định nghĩa

$$S_p(n) = \frac{T_s(n)}{T_p(n)}$$

4.2.Speedup

- Đặc điểm:
  - Speedup của hệ thống song song không phải là 1 số cố định.
  - Phụ thuộc vào kích thước bài toán (n) và số bộ xử lý P => Là hàm của (n,P)
  - Phân tích định tính thuật giải thuật song song bằng cách cố định n hoặc cố định P để vẽ các đường cong tương ứng giá trị của speedup

4.2.Speedup

- Đặc điểm:
  - Cố định n, vẽ đồ thị đường cong speedup theo P:
    - 1 trục là P
    - 1 trục là speedup
    - Phân tích hiệu năng của thuật giải khi số bộ xử lý tăng
  - Cố định P, vẽ đồ thị đường cong speedup theo n:
    - 1 trục là n
    - 1 trục là speedup
    - Nghiên cứu dáng điệu của thuật giai khi kích thước tăng lên

4.2. Speedup

- Ví dụ thuật giải sàng Eratosthenses:
  - Bài toán tìm các số nguyên tố nhỏ hơn hay bằng số nguyên dương N cho trước:
  - Chi phí thời gian thực hiện của giải thuật được tính theo công thức

$$\left\lceil \frac{N + 1 - t^2}{t} \right\rceil$$

4.3.Speedup

- Ví dụ:
  - N=1000 có các số nguyên tố: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31
  - Chi phí để sàng từng số nguyên tố

Số nguyên tố	Thời gian
2	499
3	331
5	196
7	137
11	80
13	64
17	42
19	34
23	21
29	6
31	1

4.3.Speedup

- Ví dụ:
  - Chi phí tuần tự là 1411
  - Chi phí song song được tính theo số task tham gia
- Song song:
  - 2 task tham gia giải quyết:
    - Task 1: Sàng các bội của 2, 7, 17, 23, 29, 31
    - Task 2: Sàng các bội của 3, 5, 11, 13, 19
    - Chi phí thời gian là 706
    - Speedup = 1411/706

4.3.Speedup

- Ví dụ kết quả theo biểu thức:
  - Xét bài toán có kích thước N trên máy có P bộ xử lý. Giả sử thời gian tuần tự thực hiện là $N + N^2$
  - Giả sử có 3 thuật giải song song:
    - $T_{1p}(N) = N^2/p + N$
    - $T_{2p}(N) = (N+N^2)/P + 100$
    - $T_{3p}(N)= (N+N^2)/P + 0.6P^2$
    - Xét N = 100, P=12, Speedup = 10.8

4.3. Speedup

- Ví dụ kết quả theo biểu thức:
  - Nếu tăng P thì thuật giải T_{3p} rất xấu

$$S_{3p} = \frac{N+N^2}{\frac{N+N^2}{P}+0.6P^2} \rightarrow 0 \; khi \; P \rightarrow \infty$$

4.3.Speedup

- Ví dụ kết quả theo biểu thức
  - Khi P khá lớn, thuật giải T_{1p}, T_{2p} có dạng như sau:
  - $$S_{1p} = \frac{N + N^2}{N + \frac{N^2}{P}} \rightarrow \frac{N + N^2}{N} \quad khi\ P \rightarrow \infty$$
  - $$S_{2p} = \frac{N + N^2}{\frac{N + N^2}{P} + 100} \rightarrow \frac{N + N^2}{100} \quad khi\ P \rightarrow \infty$$
  - N = 100 thì T_{1p} và T_{2p} có Speedup như nhau

4.3.Speedup

- Ví dụ theo kết quả biểu thức
  - Với P > 1 và khi đó nếu

$$N > \frac{100P}{P-1} \leftrightarrow N > 100 + \frac{N}{P}$$

$$\frac{N^2}{P} + N > \frac{N^2}{P} + 100 + \frac{N}{P} \leftrightarrow N + \frac{N^2}{P} > \frac{N + N^2}{P} + 100$$

  - T_{2p} tốt hơn T_{1p}

4.3.Efficiency

- Đặc điểm:
  - Efficiency có giá trị tối đa là 1
  - Cung cấp khả năng tận dung tối đa hiệu quả các các bộ xử lý
  - Được đo bằng công thức

$$E_p(n) = \frac{T_s(n)}{pT_p(n)} = \frac{S_p(n)}{p}$$

4.4. Định luật Amdahl

- Trong một vài chương trình có nhiều đoạn khác nhau, có thể có những đoạn dễ dàng song song hóa, có những đoạn chỉ có thể thực hiện tuần tự
- Khi đó đoạn tuần tự không thể song song được, đó là trạng thái bottle-neck

4.4.Định luật Amdahl

- Định luật:
  - Một chương trình bao gồm 2 đoạn, một đoạn chỉ có thể thực hiện tuần tự và một đoạn hoàn toàn có thể song song hóa được. Nếu đoạn tuần tự tiêu phí f của tổng thời gian thì speedup của thuật giải song song không vượt quá 1/f
- Chi thời gian thi hành $T_s(n)$ của thuật giải tuần tự (ban đầu) thành f phần

$$T_s(n) = f.T_s(n) + (1-f).T_s(n)$$

4.4.Định luật Amdahl

- Nếu giải thuật bao gồm tuần tự và song son thì thời gian thực hiện song song trên P bộ xử lý là $T_p(n)$:

$$T_p(n) = fT_s(n) + \frac{(1-f).T_s(n)}{p}$$

- Speedup là:

$$S_p(n) = \frac{T_s(n)}{T_p(n)} = \frac{T_s(n)}{f.T_s(n) + \frac{(1-f).T_s(n)}{p}} = \frac{1}{f + \frac{1-f}{p}}$$

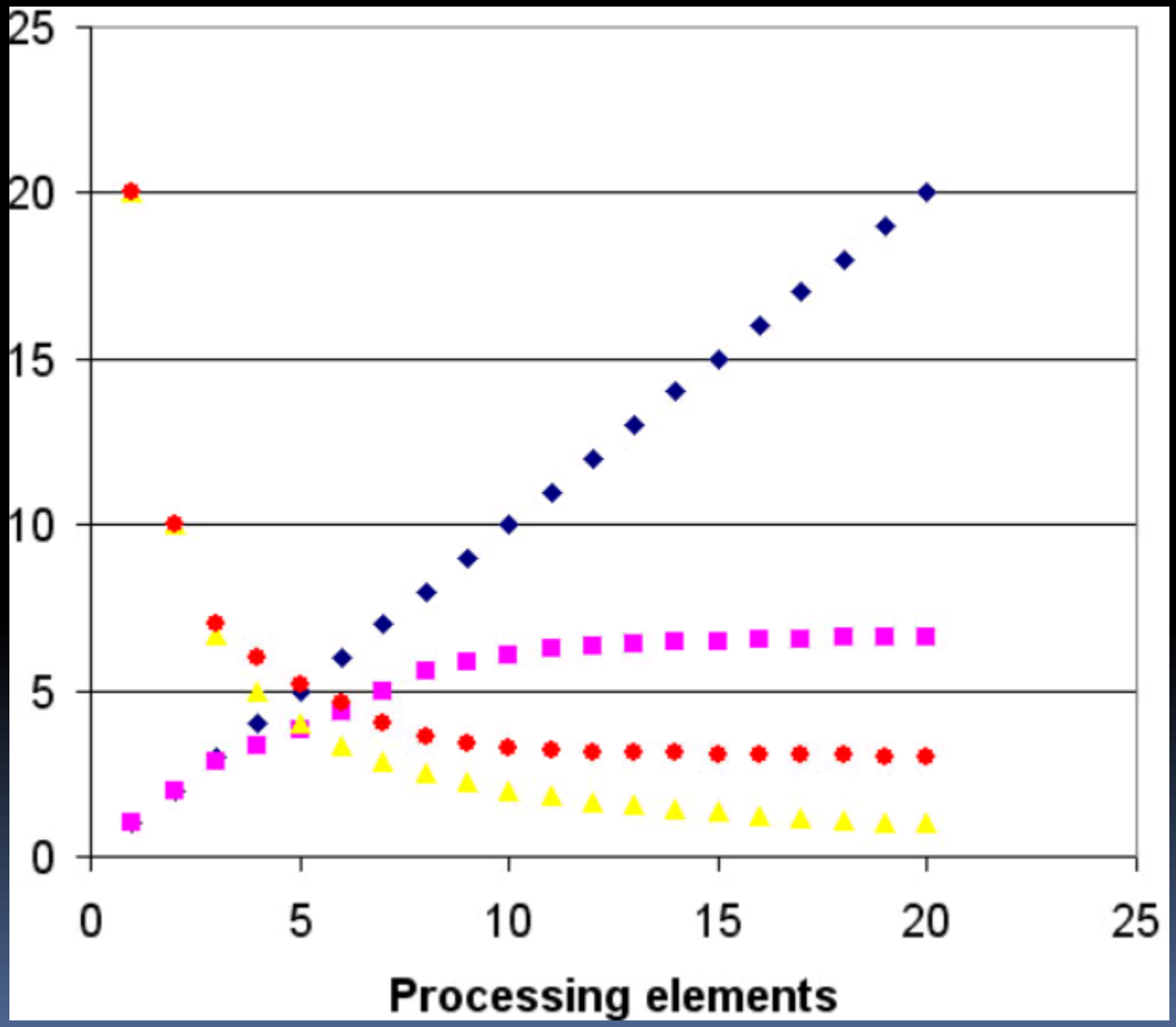
4.4.Định luật Amdahl

- Khi p khá lớn, speedup bị chặn trên bởi 1/f

$$S_p(n) = \frac{1}{f + \frac{1-f}{p}} \leq \frac{1}{f} \forall p$$

- Trạng thái tốt nhất tăng lên P bộ xử lý thì giảm đi P thời gian
- Ví dụ khi f = 10% thì speedup tối đá là 10

4.4.Định luật Amdahl



4.5.Chi phí thời gian

- Chi phí thời gian (execution time) của thuật giải song song thông thường bao gồm:
  - Thời gian tính toán (computation time)
  - Thời gian giao tiếp (communication time)
  - Thời gian chờ/ nhàn rỗi (idle time)

4.5.Chi phí thời gian

- Thời gian thi hành T của thuật giải song song bao gồm:
  - Thời gian tính toán
  - Thời gian giao tiếp
  - Thời gian chờ
- Thời gian của tiến trình thực hiện chậm nhất
  - $T = \max \{T^i_{comp} , T^i_{comm} , T^i_{idle}\}$
- Đôi khi còn lấy tổng thời gian tính toán, thời gian giao tiếp và thời gian chờ
  - $T= T^i_{comp} + T^i_{comm} + T^i_{idle}$

4.5.Chi phí thời gian

- Hoặc tính bằng trung bình thời gian tính toán, giao tiếp và chờ trên các process

$$T = \frac{T_{comp} + T_{comm} + T_{idle}}{P} = \frac{\sum_{i=0}^{P-1} T_{comp}^{i} + \sum_{i=0}^{P-1} T_{comm}^{i} + \sum_{i=0}^{P-1} T_{idle}^{i}}{P}$$

4.5. Chi phí thời gian

- Thời gian tính toán:
  - Thời gian tính toán của thuật giải T_{comp} là thời gian sử dụng cho những thao tác tính toán
  - Thời gian tính toán thường phụ thuộc vào kích thước bài toán
  - Kích thước thường được biểu diễn bằng tham số N, hoặc tập các tham số N1, N2,…Nm.
  - Nếu giải thuật song song lặp lại việc tính toán, thì thời gian tính toán cũng sẽ phụ thuộc vào số task hay số process

4.5.Chi phí thời gian

- Thời gian giao tiếp:
  - Thời gian giao tiếp Tcomm là thời gian mà những task sử dụng cho việc truyền các thông điệp giữa các task với nhau
  - Chi phí của việc gửi thông điệp giữa hai task được định vị trên những processoer khác nhau có thể biểu diễn bằng hai tham số
    - Thời gian khởi động việc truyền thông điệp t_st
    - Thời gian truyền dữ liệu L T_wL (trong đó Tw là thời gian truyền 1 dữ liệu
  - Tcomm = $t_s + t_w L$

4.5.Chi phí thời gian

- Thời gian chờ:
  - Một processor thường rảnh rỗi do việc thiếu công bằng việc tính toán hoặc thiếu dữ liệu
  - Trong trường hợp thứ nhất, thời gian rảnh rỗi này có thể ngăn chặn bằng cách sử dụng kỹ thuật load balancing
  - Trường hợp thứ hai, processor ở tình trạng nhàn rỗi do việc tính toán và truyền thông điệp đòi hỏi những dữ liệu từ xa => cho processor tính toán và thực hiện những giao tiếp khác

4.5.Chi phí thời gian

- Thời gian chờ:
  - Trường hợp thứ 2
    - Sử dụng kỹ thuật thay thế tính toán là overlapping computation and communication
    - Tính toán địa phương được thực hiện đồng thời với những tính toán và giao tiếp từ xa
    - Tạo ra nhiều task trên một processor. Khi một task làm trở ngại việc đợi dữ liệu từ xa, chuyển sang task khác mà dữ liệu đã sẵn sàng
    - Hiệu quả khi chi phí thực hiện task mới < chi phí thời gian nhàn rỗi.

4.5. Chi phí thời gian

- Ví dụ minh họa
  - Xét 1 miền 3 chiều N x N x Z điểm lưới, ở đó Z là số điểm theo phương thẳng đứng, N là số điểm theo hai phương ngang
  - Một task tương ứng với một miền con N x (N/P) x Z điểm lưới, với P là số bộ xử lý (chia miền theo một phương ngang)

4.5.Chi phí thời gian

- Giả sử mỗi bước thời gian, mỗi task thực hiện cùng một tính toán tại mỗi điểm lưới
- Khi đó thời gian tính toán là
  - $T_{comp}=t_c.N^2Z$ (t_c thời gian tính toán trung bình cho 1 điểm lưới)
- Đối với thời gian truyền dữ liệu, mỗi task sẽ giao tiếp với hai task lân cận (2 mặt), trong đó mỗi mặt gồm NZ phần tử, mỗi task sẽ trao đổi với 2NZ dữ liệu

4.5.Chi phí thời gian

- Khi đó tổng chi phí truyền tin (gửi và nhận) trên P bộ xử lý sẽ là:
  - $T_{comm} = 2P(t_s + t_w 2NZ)$
- Điều kiện tối ưu: P chia hết cho N, số lượng tính toán trên mỗi điểm lưới là hằng, thời gian chờ không đáng kể
- Chi phí thời gian:

$$T = \frac{T_{comp} + T_{comm}}{P} = \frac{N^2 Z}{P} . t_c + 2t_s + 4NZt_w$$

4.5. Chi phí thời gian

- Hiệu suất của giải thuật

$$E = \frac{t_c N^2 Z}{PT} = \frac{t_c N^2 Z}{N^2 Z t_c + P2t_s + P4NZt_w}$$

- Do đó:
  - Hiệu suất sẽ giảm đi khi tăng P, t_s và t_w
  - Hiệu suất sẽ tăng lên khi tăng N, Z và t_c
  - Thời gian thi hành giảm khi tăng P nhưng bị chặn bởi chi phí chuyển đổi giữa hai miếng dữ liệu
  - Thời gian thi hành tăng khi tăng N, Z , t_c, t_s và t_w

5. VÍ DỤ BÀI TOÁN SONG SONG

5.Ví dụ

```
INPUT     : cho 2 mảng A[1..n], B[1..n] trong bộ nhớ chung.
OUTPUT : mảng C[1..n]  =  A[1..n] + B[1..n] trong bộ nhớ chung.
BEGIN
        FOR i  =  1 TO n DO IN PARALLEL
                X       =       A[i];
                Y       =       B[i];
                C[i]    =       X + Y;
        END PARALLEL.
END;
```

- Mỗi BXL thứ i đọc giá trị A[i], B[i] từ bộ nhớ chung và ghi vào biến cục bộ X, Y.
- Giá trị C[i] được ghi bằng tổng X, Y.
- Độ phức tạp: O(1).
- Máy tính PRAM EREW.

5.Ví dụ

- Bài toán Boolean – And
  - Phát biểu bài toán:
    - INPUT: A[1..n] OF BOOLEAN.
    - OUTPUT: RESULT=A[1] and A[2] and…and A[n]
  - Thuật giải tuần tự:
    - Độ phức tạp thuật toán: O(n) với 1 BXL.

```
1  Begin
2      Result = true;
3      For i = 1 To n Do
4          Result = Result and A[i];
5      End For;
6  End
7
```

5.Ví dụ

- Bài toán Boolean – And:
  - Phân tích:
    - Nếu tất cả A[i] = TRUE suy ra KQ: TRUE.
    - Nếu có A[i] = FALSE suy ra KQ: FALSE
  - Giải thuật cho PRAM ERCW – hỗ trợ ghi đồng thời:
    - Áp dụng cơ chế ECR cho CW
    - Tổng quát cho mọi cơ chế CW

5.Ví dụ

- Bài toán Boolean – And
  - Giải thuật ECR
  - Độ phức tạp O(1)

```
1  Begin
2      Result = False;
3      For i = 1 To n Do In Parallel
4          X = A[i];
5          Result = X;
6      End For;
7  End
```

5.Ví dụ

- Bài toán Boolean – And
  - Giải thuật tổng quát với CW
  - Độ phức tạp: O(1)

```
1  Begin
2      Result = True;
3      For i = 1 To n Do In Parallel
4          If A[i] = False Then
5              Result = False;
6          End If;
7      End For
8  End
```

Định lý Brent

- Gọi M là một PRAM của một kiểu bất kỳ
- Chương trình song song P chạy trên M
- W – tổng số lượng toán tử của P trên M và ký hiệu thời gian thực hiện song song của P trên M bởi $T_{par,M}(P)$.
- Giảm số lượng đơn vị xử lý của M đến p và ký hiệu máy thu được số lượng giảm của các đơn vị xử lý là R

Định lý Brent

- Giả sử $T_{par,R}(P) \geq T_{par,M}(P)$.
- Câu hỏi: Có thể định lượng $T_{par,R}(R)$?
- Theo định lý Brent:

$$T_{par,R}(P) = O\left(\frac{W}{p} + T_{par,M}(P)\right)$$

Hết bài!!!