



TÍNH TOÁN SONG SONG

PARALLEL COMPUTING

ThS. Phan Trọng Tiến
Bộ môn CNPM – Khoa CNTT
Học Viện Nông nghiệp Việt Nam
Email: phantien84@gmail.com
Website: <http://timoday.edu.vn>

1/1/2015

Tính toán song song

1



Tóm tắt

- ❑ Đây là bài thuyết trình bao gồm các kiến thức cơ bản của tính toán song song. Bắt đầu với những kiến thức tổng quan và một vài khái niệm và các thuật ngữ tính toán song song, các chủ đề về các kiến trúc song song hoá và tìm hiểu về các mô hình lập trình song song. Các chủ đề này sẽ được đi kèm với các bài thảo luận về một số vấn đề liên quan trong việc thiết kế các chương trình song song hoá. Phần cuối cùng của bài thuyết trình sẽ đi vào nghiên cứu cách song song hoá một số bài toán lập trình tuần tự.
- ❑ Điều kiện tiên quyết: Nguyên lý hệ điều hành

1/1/2015

Tính toán song song


2



GIỚI THIỆU VỀ TÍNH TOÁN SONG SONG

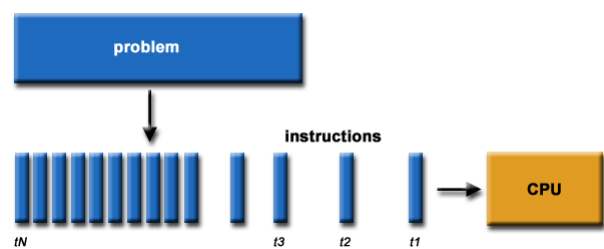
Introduction to Parallel Computing

1/1/2015 Tính toán song song 3



Tính toán song song là gì? (1)

- ❑ Thông thường, phần mềm được viết cho tính toán **tuần tự** (*serial computation*):
 - ❑ Được chạy trên máy tính đơn với một bộ xử lý trung tâm (CPU).
 - ❑ Một bài toán (problem) sẽ được chia thành một chuỗi các câu lệnh rời rạc.
 - ❑ Các câu lệnh được thực hiện một cách tuần tự.
 - ❑ Tại mỗi thời điểm chỉ thực hiện được một câu lệnh.

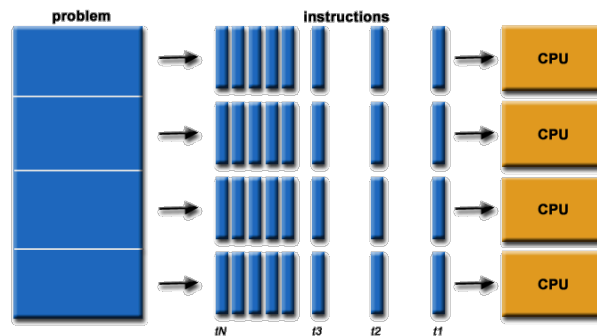


1/1/2015 Tính toán song song 4



Tính toán song song là gì? (2)

- ❑ Ý nghĩa đơn giản nhất của **tính toán song song** là việc sử dụng đồng thời nhiều tài nguyên máy tính để giải quyết bài toán về tính toán.
- ❑ Để chạy trên nhiều CPU
- ❑ Một bài toán được chia thành các phần riêng biệt mà có thể được giải quyết đồng thời.
- ❑ Mỗi phần được chia nhỏ hơn dưới một dãy các câu lệnh
- ❑ Các câu lệnh của mỗi phần thực thi đồng thời trên các CPU khác nhau



1/1/2015

Tính toán song song

5



Tính toán song song: tài nguyên

- ❑ Các nguồn tài nguyên tính toán có thể bao gồm:
 - ❑ Một máy tính đơn với nhiều bộ vi xử lý (CPU);
 - ❑ Một máy tính đơn với một hoặc nhiều CPU và một số tài nguyên chuyên dụng như GPU, FPGA ...;
 - ❑ Một số lượng tùy ý các máy tính được kết nối bởi một mạng máy tính;
 - ❑ Hoặc kết hợp của cả hai loại trên.

1/1/2015

Tính toán song song

6



Tính toán song song: vấn đề tính toán

- ❑ Vấn đề tính toán thường được thể hiện qua các đặc điểm như khả năng:
 - ❑ Chia thành các phần riêng biệt các công việc để có thể giải quyết cùng một lúc;
 - ❑ Thực thi nhiều câu lệnh chương trình tại nhiều thời điểm;
 - ❑ Giải quyết bài toán trong thời gian ít hơn với nhiều tài nguyên tính toán hơn là thực thi chỉ trên một tài nguyên tính toán duy nhất.

1/1/2015

Tính toán song song

7



Tính toán song song: để làm gì? (1)

- ❑ Tính toán song song là sự tiến hoá của tính toán tuần tự để cố gắng mô phỏng các trạng thái diễn ra trong thế giới tự nhiên: rất phức tạp, các sự kiện liên quan xảy ra cùng một thời điểm, nhưng trong cùng một chuỗi.
- ❑ Ví dụ:
 - ❑ Quỹ đạo hành tinh và thiên hà
 - ❑ Các mô hình thời tiết và đại dương
 - ❑ Kiến tạo địa chất
 - ❑ Giờ cao điểm ở Hà Nội
 - ❑ Dây truyền lắp ghép ô tô
 - ❑ Các hoạt động hàng ngày trong một doanh nghiệp
 - ❑ Xây dựng một trung tâm mua sắm
 - ❑ ...

1/1/2015

Tính toán song song

8



Tính toán song song: để làm gì? (2)

- ❑ Tính toán song song có thể được coi là “tính toán hiệu năng cao” và là động lực để mô phỏng cho các hệ thống phức tạp và giải quyết “các bài lớn” như:
 - ❑ Dự báo thời tiết và khí hậu
 - ❑ Các phản ứng hoá học và hạt nhân
 - ❑ Các bài toán sinh học và gen người
 - ❑ Các hoạt động địa chất
 - ❑ Các thiết bị cơ khí – như chân tay giả cho tàu vũ trụ
 - ❑ Các mạch điện tử
 - ❑ Các quy trình sản xuất

1/1/2015

Tính toán song song

9



Tính toán song song: để làm gì? (3)

- ❑ Ngày nay các ứng dụng thương mại đang là động lực thúc đẩy các nhà phát triển máy tính và phần mềm tạo ra các máy tính có tốc độ nhanh hơn. Vì các ứng dụng này yêu cầu xử lý một số lượng lớn dữ liệu và có độ tinh vi phức tạp cao. Ví dụ như các ứng dụng:
 - ❑ Các cơ sở dữ liệu song song, data mining
 - ❑ Thăm dò dầu khí
 - ❑ Các máy chủ tìm kiếm, các dịch vụ thương mại
 - ❑ Máy tính trợ giúp chẩn đoán trong y học
 - ❑ Quản lý các tập đoàn quốc gia và đa quốc gia
 - ❑ Cải tiến đồ hoạ và ảo hoá
 - ❑ Video mạng và các công nghệ đa phương tiện
 - ❑ Môi trường làm việc cộng tác
- ❑ Cuối cùng, giải pháp tính toán song song nhằm cố gắng để tối đa hoá những yêu vô hạn nhưng dường như chúng ta vẫn cần thêm thời gian.

1/1/2015

Tính toán song song

10



Tại sao phải tính toán song song? (1)

- ❑ Đây là một câu hỏi xác đáng! Tính toán song song là phức tạp theo nhiều khía cạnh!
- ❑ Những lý do chính cho việc sử dụng tính toán song song:
 - ❑ Tiết kiệm thời gian
 - ❑ Giải quyết những bài toán lớn
 - ❑ Xử lý đồng thời cùng một lúc

1/1/2015

Tính toán song song

11



Tại sao phải tính toán song song? (2)

- ❑ Các lý do khác có thể bao gồm:
 - ❑ Tận dụng các nguồn tài nguyên như khai thác tài nguyên tính toán có sẵn trên mạng diện rộng, hoặc thậm chí sử dụng Internet khi các tài nguyên cục bộ hạn chế
 - ❑ Tiết kiệm chi phí – sử dụng nhiều tài nguyên máy tính “rẻ” thay vì phải đầu tư một con siêu máy tính.
 - ❑ Khắc phục những hạn chế về bộ nhớ - Các máy tính đơn có tài nguyên bộ nhớ rất hữu hạn. Đối với những bài toán lớn, sử dụng bộ nhớ của nhiều máy tính có thể vượt qua trở ngại này.

1/1/2015

Tính toán song song

12



Các giới hạn của tính toán tuần tự

- ❑ Các giới hạn để tính toán tuần tự - Cả hai lý do giới hạn về vật lý và thực tiễn đặt ra những hạn chế đáng kể để xây dựng được ứng dụng chạy nhanh hơn trên máy tính tuần tự.
- ❑ **Tốc độ truyền dẫn** – Tốc độ của máy tính tuần tự phụ thuộc trực tiếp vào tốc độ di chuyển của dữ liệu trên phần cứng.
 - ❑ Giới hạn tuyệt đối là tốc độ của ánh sáng (30 cm/ns)
 - ❑ Giới hạn truyền của dây đồng (9 cm/ns).
 - ❑ Việc tăng tốc độ truyền cần tăng tốc độ xử lý các phần tử? (1 ns (nanosecond) = 10^{-9} s)
- ❑ **Giới hạn để thu nhỏ** - công nghệ bộ vi xử lý ngày càng cho phép tăng số transistor được đặt trên các con chip. Tuy nhiên, thậm chí ngay cả các transistor có kích thước là phân tử hoặc mức nguyên tử thì số lượng tích hợp trên thiết bị cũng sẽ đạt tới giới hạn.
- ❑ **Hạn chế về kinh tế** - Giá thành sẽ càng đắt khi tạo ra một bộ đơn vi xử lý (VXL) chạy nhanh. Sẽ kinh tế hơn nếu sử dụng một số lượng bộ vi xử lý nhanh vừa phải nhưng có thể đạt được hiệu suất như bộ VXL đơn chạy nhanh (hoặc tốt hơn).

1/1/2015

Tính toán song song

13



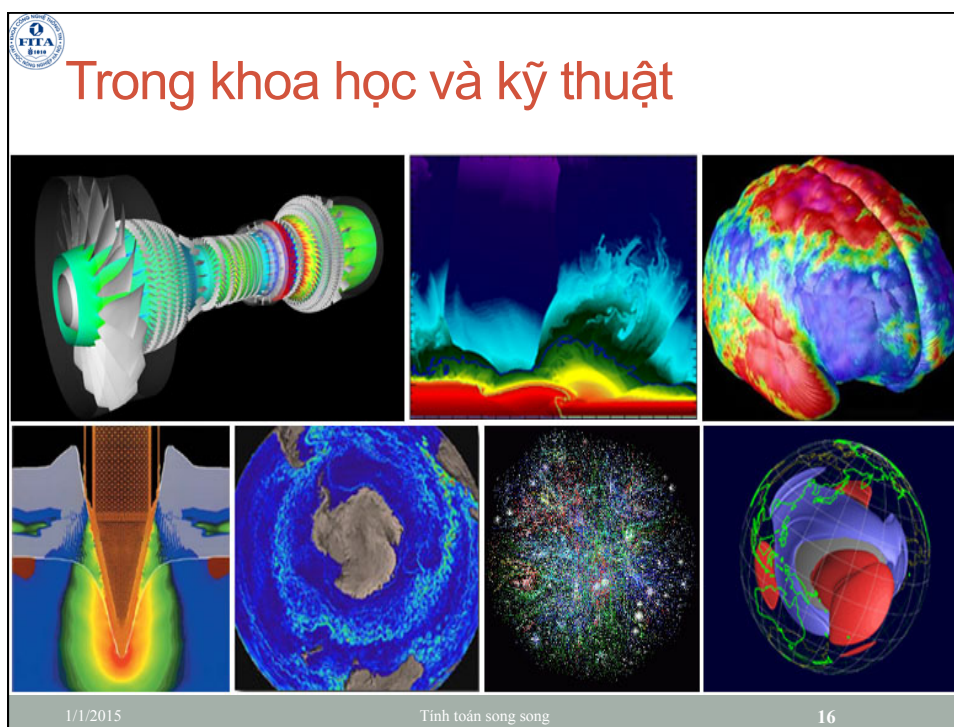
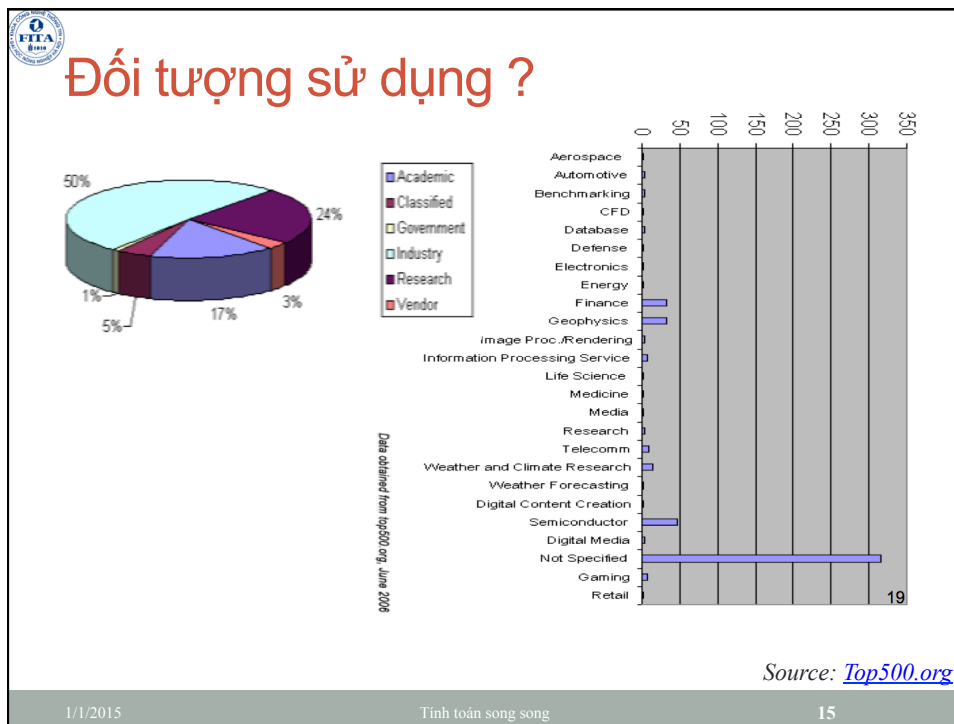
Tương lai


- ❑ Trong suốt 10 năm qua, xu hướng chỉ ra rằng mạng máy tính ngày càng nhanh hơn, có nhiều hệ thống phân tán, và các kiến trúc máy tính đa vi xử lý (bao gồm cả máy tính để bàn) cho thấy rõ ràng ***song song là tương lai của máy tính.***
- ❑ Sẽ rất đa dạng, pha trộn giữa các giải pháp thông dụng và cả các giải pháp chuyên dụng như IBM Cells, ClearSpeed, GPGPU từ NVidia ...

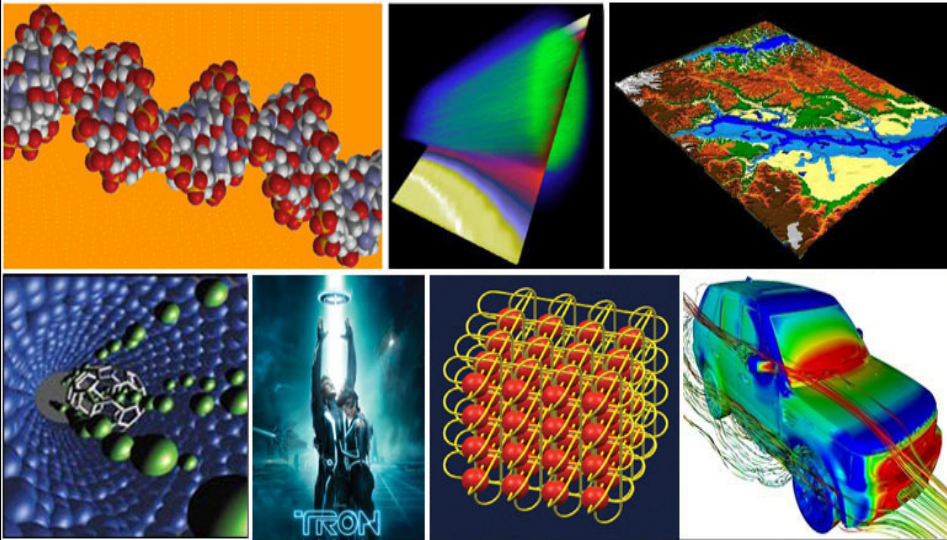
1/1/2015

Tính toán song song


14



 Công nghiệp và thương mại



1/1/2015 Tính toán sóng song 17

 CÁC KHÁI NIỆM VÀ THUẬT NGỮ

Concepts and Terminology

1/1/2015 Tính toán sóng song 18



Kiến trúc Von Neumann

- ❑ Trong hơn 40 năm, hầu hết tất cả các máy tính đã đi theo một mô hình máy tính phổ biến được gọi là máy tính Von Neumann. Được đặt tên theo nhà toán học Hungary John von Neumann.
- ❑ Một máy tính Von Neumann sử dụng khái niệm chương trình lưu trữ. CPU thực hiện chương trình được lưu trữ được chỉ định bởi một chuỗi tác vụ đọc và ghi trên bộ nhớ.

1/1/2015

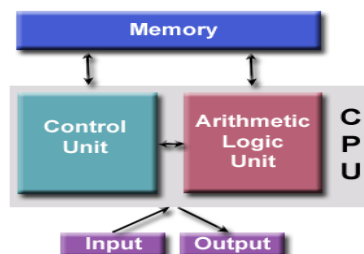
Tinh toán song song

19



Thiết kế cơ bản

- ❑ Thiết kế cơ bản
 - ❑ Bộ nhớ được sử dụng để lưu trữ dữ liệu (data) và các câu lệnh chương trình (instruction)
 - ❑ Các câu lệnh chương trình được mã hoá để “nói” cho máy tính làm một công việc nào đó
 - ❑ Dữ liệu chỉ đơn giản là thông tin được sử dụng bởi chương trình
- ❑ Bộ xử lý trung tâm (CPU) nhận các câu lệnh và dữ liệu từ bộ nhớ, giải mã các chỉ dẫn và thực thi tuần tự chúng.



1/1/2015

Tinh toán song song

20



Phân loại máy tính song song Flynn

- ❑ Có rất nhiều cách khác nhau để phân loại các máy tính song song. Một trong những cách phân loại được sử dụng rộng rãi từ năm 1966 được gọi là phân loại Flynn.
- ❑ Phân loại Flynn phân biệt các kiến trúc máy tính nhiều bộ vi xử lý theo hai khía cạnh chỉ thị lệnh (***Instruction***) và dữ liệu (***Data***). Mỗi khía cạnh này có thể có 2 trạng thái: ***Single*** hoặc ***Multiple***.

1/1/2015

Tinh toán song song

21



Ma trận Flynn

- ❑ Ma trận bên dưới định nghĩa 4 phân loại có thể có theo Flynn

SISD Single Instruction, Single Data	SIMD Single Instruction, Multiple Data
MISD Multiple Instruction, Single Data	MIMD Multiple Instruction, Multiple Data

1/1/2015

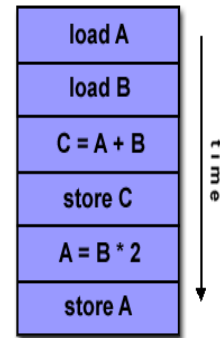
Tinh toán song song

22



Single Instruction, Single Data (SISD)

- ❑ Máy tính tuần tự (không song song)
- ❑ Đơn lệnh: chỉ có một dòng lệnh được thực hiện trong một chu kỳ đồng hồ
- ❑ Đơn dữ liệu: chỉ có một luồng dữ liệu được sử dụng làm đầu vào trong một chu kỳ đồng hồ
- ❑ Thực hiện theo một trật tự xác định
- ❑ Ra đời lâu nhất và vẫn dùng cho tới gần đây, thịnh hành nhất của máy tính
- ❑ Ví dụ: hầu hết các máy tính cá nhân, máy trạm và máy tính lớn một CPU



1/1/2015

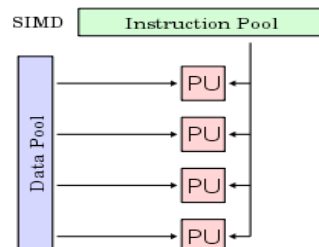
Tinh toán song song

23



Single Instruction, Multiple Data (SIMD)

- ❑ Một kiểu máy tính song song
- ❑ Đơn dòng lệnh: Tất cả các đơn vị xử lý thực thi cùng dòng lệnh trong cùng xung nhịp đồng hồ
- ❑ Đa dữ liệu: Mỗi đơn vị xử lý có thể thao tác trên các mục dữ liệu khác nhau
- ❑ Kiểu máy này thường có một điều phối lệnh, một mạng nội bộ có băng thông rất cao, và một mảng rất lớn của các đơn vị lệnh.



1/1/2015

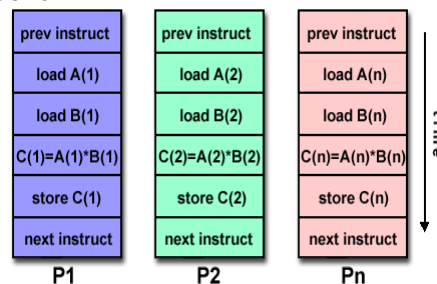
Tinh toán song song

24



Single Instruction, Multiple Data (SIMD)

- ❑ Phù hợp nhất cho các bài toán đặc biệt có độ tính toán cao như xử lý ảnh.
- ❑ Tính toán đồng bộ (khóa theo các bước) và xác định
- ❑ Có hai dạng: Processor Arrays and Vector Pipelines
- ❑ Ví dụ:
 - ❑ Processor Arrays: Connection Machine CM-2, Maspar MP-1, MP-2
 - ❑ Vector Pipelines: IBM 9000, Cray C90, Fujitsu VP, NEC SX-2, Hitachi S820



1/1/2015

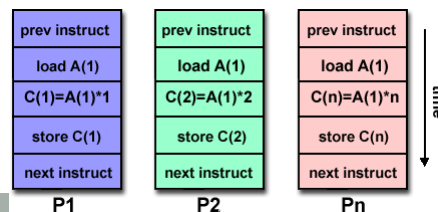
Tính toán song song

25



Multiple Instruction, Single Data (MISD)

- ❑ Một luồng đơn dữ liệu được nạp vào nhiều đơn vị xử lý.
- ❑ Một đơn vị xử lý hoạt động trên dữ liệu độc lập theo hướng các dòng lệnh độc lập.
- ❑ Một vài ví dụ thực tế của lớp máy tính song song này đã từng tồn tại. Một thử nghiệm của máy tính Carnegie-Mellon C.mmp (1971).
- ❑ Một số ứng dụng có thể sử dụng:
 - ❑ Nhiều bộ lọc tần số hoạt động dựa trên một luồng tín hiệu duy nhất
 - ❑ Nhiều thuật toán mã hoá cố gắng để bẻ khóa (crack) một mật mã duy nhất.



1/1/2015

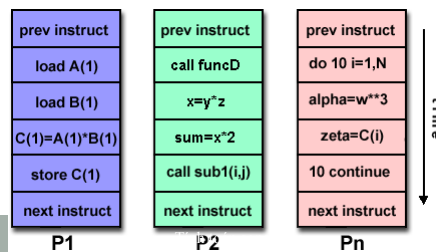
Tính toán song song

26



Multiple Instruction, Multiple Data (MIMD)

- ❑ Hiện nay phổ biến nhất trong máy tính song song. Các mô hình máy tính hiện đại nhất thuộc loại này.
- ❑ Đa lệnh: mỗi bộ vi xử lý có thể thực thi một luồng câu lệnh khác nhau
- ❑ Đa dữ liệu: mỗi bộ vi xử lý có thể được làm việc với một luồng dữ liệu khác nhau
- ❑ Thực thi có thể là đồng bộ hoặc không đồng bộ, xác định hoặc không xác định
- ❑ Ví dụ: hầu hết các siêu máy tính hiện nay, mạng máy tính song song dạng “lưới” và các máy tính SMP đa bộ vi xử lý – bao gồm cả một số loại máy tính cá nhân.



1/1/2015

27



Một số thuật ngữ song song

Cũng giống như các lĩnh vực khác khác, tính toán song song cũng có “thuật ngữ” riêng. Một số thuật ngữ thường được sử dụng gắn với tính toán song song được liệt kê bên dưới.

- ❑ **Tác vụ (Task)**
 - ❑ Một phần logic riêng rẽ của công việc tính toán. Một tác vụ thường là một chương trình hoặc tập các lệnh giống chương trình mà được thực thi bởi một bộ vi xử lý.
- ❑ **Tác vụ song song (Parallel Task)**
 - ❑ Một tác vụ có thể được thực thi bởi nhiều bộ vi xử lý một cách an toàn (cho kết quả chính xác)
- ❑ **Thực thi tuần tự (Serial Execution)**
 - ❑ Thực thi tuần tự một chương trình, một câu lệnh chỉ thực thi tại một thời điểm. Ý nghĩa đơn giản nhất ở đây là việc thực hiện các công việc trên một máy tính đơn bộ vi xử lý.
 - ❑ Tuy nhiên, hầu như tất cả các tác vụ song song sẽ có một số phần của chương trình song song mà phải thực hiện tuần tự.

1/1/2015

Tính toán song song

28



Một số thuật ngữ song song

❑ Thực thi song song (Parallel Execution)

- ❑ Thực hiện một chương trình bởi nhiều tác vụ, với mỗi tác vụ có thể thực thi cùng hoặc khác câu lệnh tại cùng một thời điểm.

❑ Bộ nhớ chia sẻ (Shared Memory)

- ❑ Theo quan điểm hẹp về phần cứng, mô tả một kiến trúc của máy tính mà tất cả các bộ vi xử lý có truy cập trực tiếp tới bộ nhớ vật lý chung.
- ❑ Theo quan điểm lập trình, nó mô tả một mô hình ở đó các tác vụ song song có cùng một "hình ảnh" của bộ nhớ và có thể đánh địa chỉ trực tiếp và truy cập tới cùng vị trí bộ nhớ logic.

❑ Bộ nhớ phân tán (Distributed Memory)

- ❑ Trong phần cứng, đề cập tới mạng máy tính truy cập bộ nhớ dựa trên cơ sở không dùng chung bộ nhớ vật lý.
- ❑ Trong mô hình lập trình, về logic các tác vụ này chỉ có thể "nhìn thấy" bộ nhớ của máy cục bộ và phải sử dụng các giao tiếp để truy cập bộ nhớ trên các máy khác mà ở đó các tác vụ khác đang được thực hiện.

1/1/2015

Tính toán song song

29



Một số thuật ngữ song song

❑ Truyền thông (Communications)

- ❑ Các tác vụ song song thường cần trao đổi dữ liệu. Có nhiều cách có thể được thực hiện, như qua bộ nhớ chia sẻ bus hoặc qua mạng. Việc trao đổi dữ liệu thường được gọi là truyền thông, dù chúng thực hiện bằng bất kỳ phương thức nào.

❑ Đồng bộ (Synchronization)

- ❑ Phối hợp các tác vụ song song theo thời gian thực, thường hay được thực hiện bằng các truyền thông. Thường được thiết lập bằng việc thiết lập một điểm đồng bộ cho ứng dụng mà một tác vụ có dừng lại đợi cho đến khi các vụ khác đạt tới cùng điểm tới hạn hoặc điểm logic tương đương.
- ❑ Đồng bộ hoá thường liên quan đến chờ đợi ít nhất một tác vụ, và do đó có thể gây ra thời gian thực hiện của ứng dụng song song tăng lên.

1/1/2015

Tính toán song song

30



Một số thuật ngữ song song

❑ Tính hạt (Granularity)

- ❑ Trong tính toán song song, tính hạt là thước đo chất lượng của tỷ lệ tính toán với giao tiếp.
- ❑ **Hạt thô (Coarse):** số lượng tương đối lớn công việc tính toán được thực hiện giữa các sự kiện truyền thông
- ❑ **Hạt tinh (Fine):** số lượng tương đối nhỏ công việc tính toán được thực hiện giữa các sự kiện giao tiếp

❑ Tốc độ quan sát (Observed Speedup)

- ❑ Tốc độ quan sát của code đã được song song hoá, được định nghĩa là:
Thời gian thực hiện tuần tự
Thời gian tính toán song song
- ❑ Đây là một chỉ số đơn giản nhất và được sử dụng rộng rãi được sử dụng để đo hiệu năng của một chương trình song song.

1/1/2015

Tính toán song song

31



Một số thuật ngữ song song

❑ Chi phí cho tính toán song song (Parallel Overhead)

- ❑ Lượng thời gian cần thiết để phối hợp các tác vụ song song, ngược lại với thời gian để làm công việc hữu ích nào đó. Parallel Overhead có thể bao gồm các hệ số như sau:
 - ❑ Thời gian khởi tạo tác vụ
 - ❑ Đồng bộ hoá
 - ❑ Giao tiếp dữ liệu
 - ❑ Chi phí phần mềm ngầm định bởi các trình biên dịch song song, các thư viện, các công cụ, hệ điều hành, v.v.
 - ❑ Thời gian kết thúc tác vụ

❑ Massively Parallel

- ❑ Đề cập tới phần cứng cho một hệ thống song song - có nhiều bộ vi xử lý. để thực hiện phối hợp song song.
- ❑ Tham khảo: [https://en.wikipedia.org/wiki/Massively_parallel_\(computing\)](https://en.wikipedia.org/wiki/Massively_parallel_(computing))

1/1/2015

Tính toán song song

32



Một số thuật ngữ song song

❑ Khả năng mở rộng (Scalability)

- ❑ Đề cập đến một hệ thống song song (phần cứng và/hoặc phần mềm) có khả năng chứng minh được sự gia tăng thêm nhiều bộ vi xử lý sẽ tương ứng tỷ lệ tốc độ tính toán song song. Các yếu tố góp phần vào khả năng mở rộng bao gồm:
 - ❑ Phần cứng – đặc biệt là băng thông bộ nhớ - cpu và mạng truyền thông
 - ❑ Thuật toán
 - ❑ Liên quan đến chi phí cho tính toán song song
 - ❑ Các đặc điểm của ứng dụng cụ thể và cách thức lập trình



CÁC KIẾN TRÚC BỘ NHỚ MÁY TÍNH SONG SONG

Parallel Computer Memory Architectures



Các kiến trúc bộ nhớ

- ❑ Bộ nhớ chia sẻ - Shared Memory
- ❑ Bộ nhớ phân tán - Distributed Memory
- ❑ Lai bộ nhớ chia sẻ và phân tán - Hybrid Distributed-Shared Memory

1/1/2015

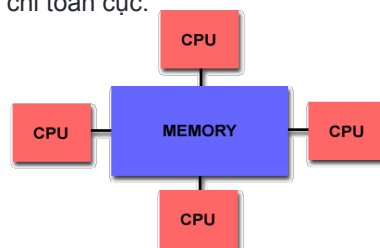
Tinh toán song song

35



Bộ nhớ chia sẻ

- ❑ Các máy tính song song với bộ nhớ chia sẻ rất đa dạng, nhưng chung nhất là khả năng tất cả các bộ xử lý truy cập vào bộ nhớ giống như là không gian địa chỉ toàn cục.



- ❑ Nhiều bộ xử lý có thể thao tác độc lập nhưng chia sẻ cùng các tài nguyên bộ nhớ.
- ❑ Những thay đổi trong một vị trí của bộ nhớ bị ảnh hưởng bởi một bộ xử lý thì các bộ xử lý khác có thể nhìn thấy.
- ❑ Các máy chia sẻ bộ nhớ có thể được chia thành hai nhóm chính dựa trên thời gian truy cập: **UMA** và **NUMA**.

1/1/2015

Tinh toán song song

36



Bộ nhớ chia sẻ : UMA với NUMA

- ❑ Uniform Memory Access (UMA):
 - ❑ Ngày nay hầu hết là các máy đa bộ vi xử lý đối xứng - Symmetric Multiprocessor (SMP)
 - ❑ Các bộ vi xử lý giống hệt nhau
 - ❑ Bình đẳng truy cập và thời gian truy cập tới bộ nhớ
 - ❑ Đôi khi được gọi CC-UMA - Cache Coherent UMA. Cache coherent có nghĩa nếu một bộ vi xử lý cập nhật một vị trí trong bộ nhớ chia sẻ thì tất cả các bộ nhớ khác sẽ biết về cập nhật này. Cache coherency được thực hiện ở cấp độ phần cứng.
- ❑ Non-Uniform Memory Access (NUMA):
 - ❑ Thường được làm bởi liên kết hai hoặc nhiều SMP
 - ❑ Một SMP có thể truy cập trực tiếp bộ nhớ của một SMP khác
 - ❑ Không phải tất cả các bộ xử lý có thời gian truy cập bằng nhau với tất cả bộ nhớ.
 - ❑ Bộ nhớ truy cập qua liên kết thì chậm hơn
 - ❑ Nếu cache coherent được duy trì thì cũng có thể được gọi là CC-NUMA - Cache Coherent NUMA

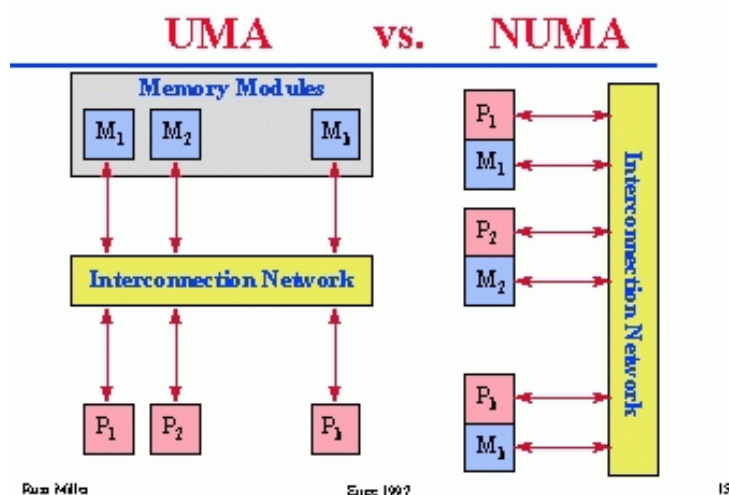
1/1/2015

Tinh toán song song

37



Bộ nhớ chia sẻ : UMA với NUMA



1/1/2015

Tinh toán song song

38



Bộ nhớ chia sẻ: ưu và nhược điểm

□ Ưu điểm:

- Không gian địa chỉ toàn cục cung cấp ở một khía cạnh thân thiện người sử dụng lập trình với bộ nhớ
- Chia sẻ dữ liệu giữa các tác vụ là nhanh chóng và đồng bộ nhờ khoảng cách gần nhau giữa bộ nhớ tới các CPU

□ Nhược điểm:

- Thiếu sự mở rộng giữa bộ nhớ và các CPU. Thêm nhiều CPU về phương diện hình học có thể tăng lưu lượng truyền giữa bộ nhớ chia sẻ và CPU, và các hệ thống gắn kết cache, tăng lưu lượng truy cập liên quan đến quản lý với cache/bộ nhớ.
- Lập trình viên có trách nhiệm đồng bộ hoá giữa các cấu trúc để đảm bảo truy cập "đúng" bộ nhớ toàn cục.
- Chi phí: càng trở nên khó khăn và tốn kém để thiết kế và sản xuất ra các máy chia sẻ bộ nhớ với việc ngày càng tăng số bộ vi xử lý.

1/1/2015

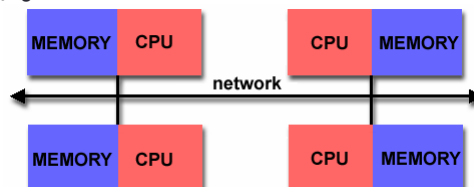
Tinh toán song song

39



Bộ nhớ phân tán

- Giống như hệ thống bộ nhớ chia sẻ, hệ thống bộ nhớ phân tán rất đa dạng nhưng chúng có những đặc điểm chung: Các hệ thống chia sẻ bộ nhớ yêu cầu một mạng lưới truyền thông để kết nối bộ nhớ của các bộ xử lý.
- Các bộ xử lý có bộ nhớ cục bộ riêng. Địa chỉ bộ nhớ trong một bộ xử lý không ánh xạ tới địa chỉ của bộ nhớ khác vì vậy không có khái niệm về không gian địa chỉ toàn cục trên tất cả các bộ xử lý.
- Vì mỗi bộ xử lý có bộ nhớ cục bộ riêng, nên các thao tác là độc lập. Các thay đổi được thực hiện ở bộ nhớ cục bộ mà không ảnh hưởng tới bộ nhớ trên các bộ xử lý khác. Do đó khái niệm về cache coherency không được áp dụng ở đây.
- Khi một bộ xử lý cần truy cập dữ liệu ở một bộ xử lý khác, thường là nhiệm vụ của lập trình viên cần định nghĩa tường minh cách thực hiện và khi nào dữ liệu được trao đổi. Đồng bộ hoá giữa các tác vụ là trách nhiệm của người lập trình.
- Cấu trúc mạng được sử dụng để truyền dữ liệu rất đa dạng, mặc dù vậy để đơn giản có thể dùng mạng Ethernet.



1/1/2015

Tinh toán song song

40



Bộ nhớ phân tán: ưu và nhược điểm

□ Ưu điểm

- Bộ nhớ có khả năng mở rộng với nhiều bộ vi xử lý. Tăng số lượng bộ vi xử lý và kích thước bộ nhớ tăng tương ứng
- Mỗi bộ vi xử lý có thể truy cập nhanh bộ nhớ riêng của nó mà không có sự can thiệp và không có các chi phí phát sinh xảy ra khi có gắng duy trì liên kết bộ nhớ cache.
- Hiệu quả chi phí: có thể sử dụng linh hoạt với các bộ xử lý có thể dễ dàng thay đổi và mạng máy tính.

□ Nhược điểm

- Lập trình viên phải đảm nhiệm nhiều việc liên quan đến giao tiếp giữa các bộ xử lý.
- Có thể sẽ gặp khó khăn để ánh xạ tới các cấu trúc dữ liệu tồn tại với cơ sở bộ nhớ toàn cục và với cách tổ chức trên bộ nhớ này.
- Không đồng bộ thời gian truy cập bộ nhớ (NUMA)

1/1/2015

Tinh toán song song

41



Lai bộ nhớ chia sẻ và phân tán

Tóm tắt một vài đặc điểm chính của bộ nhớ chia sẻ và phân tán

So sánh các kiến trúc bộ nhớ chia sẻ và phân tán			
Kiến trúc	CC-UMA	CC-NUMA	Phân tán
Ví dụ	SMPs Sun Vexx DEC/Compaq SGI Challenge IBM POWER3	Bull NovaScale SGI Origin Sequent HP Exemplar DEC/Compaq IBM POWER4 (MCM)	Cray T3E Maspar IBM SP2 IBM BlueGene
Giao tiếp	MPI Threads OpenMP shmem	MPI Threads OpenMP shmem	MPI
Khả năng mở rộng	đến 10 bộ xử lý	đến 100 bộ xử lý	đến 1000 bộ xử lý
Các hạn chế	Bảng thông Memory-CPU	Bảng thông Memory-CPU Không đồng bộ thời gian truy cập	Quản trị hệ thống Khó khăn trong phát triển và bảo trì chương trình
Phần mềm có sẵn	đến 1000s ISVs	many 1000s ISVs	100s ISVs

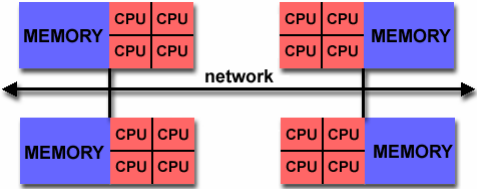
1/1/2015

Tinh toán song song

42

Lai bộ nhớ phân tán và chia sẻ

- Các máy tính lớn nhất và nhanh nhất hiện nay đều sử dụng cả hai kiến trúc bộ nhớ này




- Thành phần bộ nhớ chia sẻ thường là một máy SMP với cache coherent. Các bộ xử lý trên SMP có thể đánh địa chỉ bộ nhớ máy tính như toàn cục.
- Thành phần trên bộ nhớ phân tán là mạng máy tính của nhiều SMP. Các SMP chỉ biết về bộ nhớ riêng của chúng, không phải bộ nhớ trên các SMP khác. Do đó các giao tiếp mạng là cần thiết để trao đổi dữ liệu từ một SMP này tới SMP khác.
- Xu hướng hiện nay dường như chỉ ra rằng kiểu kiến trúc bộ nhớ sẽ tiếp tục chiếm ưu thế và tăng khả năng tính toán trong tương lai.
- Các ưu và nhược điểm: kế thừa các ưu nhược điểm của cả hai hệ thống bộ nhớ phân tán và chia sẻ.

1/1/2015 Tính toán song song 43

CÁC MÔ HÌNH LẬP TRÌNH SONG SONG

Parallel Programming Models


1/1/2015 Tính toán song song 44



Nội dung

- ☐ Tổng quan
- ☐ Mô hình chia sẻ bộ nhớ
- ☐ Mô hình luồng (Thread)
- ☐ Mô hình gửi thông điệp (MPI)
- ☐ Mô hình song song dữ liệu
- ☐ Các mô hình khác

1/1/2015 Tính toán song song 45



Tổng quan

- ☐ Có nhiều mô hình lập trình song song thường sử dụng:
 - ☐ Chia sẻ bộ nhớ
 - ☐ Luồng
 - ☐ Gửi thông điệp
 - ☐ Song song dữ liệu
 - ☐ Lai các mô hình
- ☐ Các mô hình lập trình song song tồn tại như một sự trừu tượng hoá trên các kiến trúc phần cứng và phần mềm.

1/1/2015 Tính toán song song 46



Tổng quan

- ❑ Mặc dù có vẻ không rõ ràng, các mô hình này không cụ thể cho một kiểu kiến trúc máy hay bộ nhớ nào đặc biệt. Thực tế với bất kỳ mô hình nào (về lý thuyết) đều có thể áp dụng cho bất kỳ kiến trúc phần cứng.
- ❑ Mô hình chia sẻ bộ nhớ trên máy tính có bộ nhớ phân tán: theo cách tiếp cận **Kendall Square Research (KSR) ALLCACHE**.
 - ❑ Bộ nhớ của máy là phân tán về vật lý nhưng được sử dụng như một bộ nhớ chia sẻ đơn. Thường phương pháp này gọi là "bộ nhớ chia sẻ ảo" hay "virtual shared memory".
 - ❑ Chú ý: mặc dù KSR không còn dùng trong thương mại nhưng cũng không có lý do gì việc triển khai một hệ thống tương tự bởi một nhà cung cấp khác trong tương lai
- ❑ Mô hình gửi thông điệp trên các máy chia sẻ bộ nhớ: MPI trên SGI Origin.
 - ❑ **SGI Origin** sử dụng kiểu **CC-NUMA** của kiến trúc chia sẻ bộ nhớ, ở đó mỗi tác vụ có quyền truy cập vào bộ nhớ toàn cục. Khả năng gửi và nhận thông điệp với MPI được thực hiện thông qua mạng các máy có bộ nhớ phân tán. Tuy nhiên chức năng gửi thông điệp không được thực thi nhưng nó vẫn được sử dụng trên hệ thống này.

1/1/2015

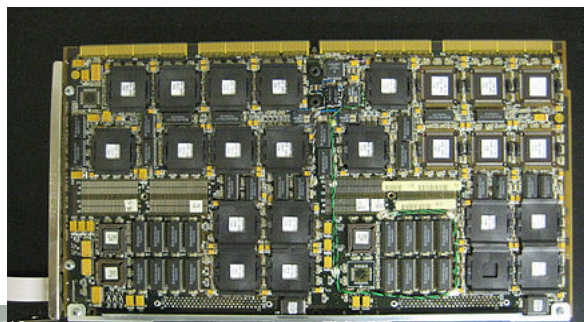
Tinh toán song song

47



Tổng quan

- ❑ Việc lựa chọn mô hình nào sử dụng thường là sự kết hợp cái có sẵn và lựa chọn cá nhân. **Không có mô hình "tốt nhất"**, mặc dù vậy chắc chắn có một vài mô hình là tốt hơn những mô hình khác.
- ❑ Các phần sau mô tả mỗi mô hình được đề cập ở trên và thảo luận một số dự án triển khai thực tế của chúng.



KSR1

1/1/2015

Tinh toán song song

48



Mô hình chia sẻ bộ nhớ

- ❑ Trong mô hình lập trình chia sẻ bộ nhớ, các tác vụ chia sẻ không gian địa chỉ chung mà chúng có thể đọc/ghi không đồng bộ.
- ❑ Các kỹ thuật khác nhau như lock/semaphore có thể được sử dụng để điều khiển truy cập tới vùng bộ nhớ chia sẻ.
- ❑ Một thuận lợi của mô hình này từ khía cạnh của lập trình viên là không có khái niệm về “quyền sở hữu” dữ liệu vì vậy không cần phải xác định một cách rõ ràng về giao tiếp dữ liệu giữa các tác vụ. Vì vậy, phát triển chương trình thường có thể được đơn giản hoá.
- ❑ Một nhược điểm quan trọng về hiệu suất, nó trở nên khó khăn trong việc hiểu và quản lý dữ liệu cục bộ.

1/1/2015

Tinh toán song song

49



Mô hình chia sẻ bộ nhớ: cách thực hiện

- ❑ Trên các nền tảng bộ nhớ chia sẻ, các trình biên dịch gốc dịch các biến sử dụng của người dùng thành địa chỉ bộ nhớ thực tế, địa chỉ bộ nhớ này là toàn cục.
- ❑ Hiện nay không còn triển khai trên nền tảng bộ nhớ phân tán. Tuy nhiên như đã đề cập ở phần tổng quan, cách tiếp cận KSR ALLCACHE đã cung cấp một cách nhìn về chia sẻ bộ nhớ của dữ liệu bộ nhớ vật lý của các máy này.

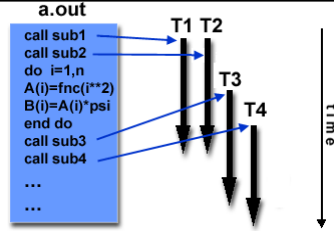
1/1/2015

Tinh toán song song

50




Mô hình luồng (Thread)



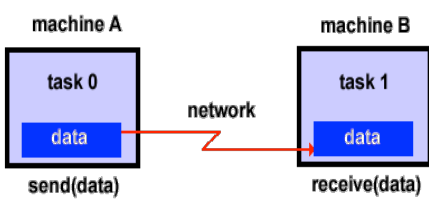
- ❑ Trong lập trình song song theo mô hình thread, một xử lý đơn có thể chuyển thành đa xử lý bằng cách thực thi đồng thời theo các cách khác nhau.
- ❑ Đơn giản nhất để mô tả mô hình thread, chúng ta phân tích ví dụ trên với một số thủ tục trong chương trình:
 - ❑ Chương trình chính **a.out** được lập lịch để chạy trên hệ điều hành. a.out nạp và yêu cầu hệ thống và các tài nguyên cần thiết để chạy chương trình.
 - ❑ a.out thực thi một dãy các công việc, và rồi **tạo ra một dãy các tác vụ (các thread)** mà có thể được lập lịch và chạy đồng thời bởi hệ điều hành.
 - ❑ **Mỗi thread có dữ liệu cục bộ**, nhưng cũng **chia sẻ toàn bộ tài nguyên của a.out**. Điều này tiết kiệm các chi phí liên quan đến tái tạo tài nguyên của chương trình cho mỗi thread. Mỗi thread cũng có quyền với bộ nhớ toàn cục vì nó chia sẻ không gian bộ nhớ của a.out.
 - ❑ Công việc của thread có thể được miêu tả tốt như một chương trình con trong chương trình chính. Bất kỳ thread nào cũng có thể thực thi bất kỳ chương trình con tại cùng thời điểm như các thread khác.
 - ❑ **Các thread giao tiếp** với nhau qua **bộ nhớ toàn cục** (cập nhật vị trí địa chỉ). Điều này đòi hỏi các cấu trúc đồng bộ để đảm bảo rằng khi có nhiều hơn một thread thì sẽ không cập nhật lên cùng địa chỉ toàn cục tại cùng thời điểm.
 - ❑ Các thread có thể chạy luân phiên nhau, nhưng a.out vẫn giữ lại trạng thái hiện tại để cung cấp các tài nguyên chia sẻ khác cho đến khi ứng dụng kết thúc.
- ❑ Các thread thường được kết hợp với các kiến trúc bộ nhớ chia sẻ và hệ điều hành.

1/1/2015
Tính toán song song
51



Thực thi các mô hình Thread

- ❑ Từ khía cạnh lập trình, để thực thi thread thường bao gồm:
 - ❑ Một thư viện các chương trình con mà được gọi từ bên trong mã nguồn song song.
 - ❑ Một tập các chỉ thị biên dịch được nhúng vào cả hai mã nguồn tuần tự hoặc song song.
- ❑ Trong cả hai trường hợp, lập trình viên phải có trách nhiệm xác định thành phần nào cần xử lý xong.
- ❑ Thực thi thread không phải là mới trong tính toán máy tính. Trước đây, các nhà cung cấp phần cứng đã thực thi các phiên bản thread riêng cho phần cứng của họ. Việc thực thi này cơ bản là khác nhau và nó gây khó khăn cho các lập trình viên phát triển ứng dụng thread để chạy trên các nền tảng phần cứng khác nhau.
- ❑ Có hai chuẩn khác nhau trong việc thực thi thread: **POSIX Threads** và **OpenMP**.



1/1/2015
Tính toán song song
52



Các mô hình thread: POSIX Thread

❑ POSIX Threads

- ❑ Thư viện cơ sở, yêu cầu viết mã song song
- ❑ Đặc tả theo chuẩn IEEE POSIX 1003.1c (1995).
- ❑ Hỗ trợ ngôn ngữ C
- ❑ Thường được gọi là Pthread.
- ❑ Hầu hết các nhà cung cấp phần cứng hiện nay đều tích hợp Pthreads
- ❑ Lập trình Song song hoá rất khác biệt, đòi lập trình viên chú ý đến từng chi tiết.

1/1/2015

Tính toán song song

53



Threads Model: OpenMP

❑ OpenMP

- ❑ Dựa trên chỉ thị biên dịch (compiler directive); có thể sử dụng mã tuần tự
- ❑ Được định nghĩa và phối hợp bởi một nhóm các nhà cung cấp phần cứng máy tính và phần mềm. OpenMP Fortran API được phát hành vào tháng 28 tháng 10, 1997. C/C++ API được phát hành vào cuối năm 1998.
- ❑ Portable/multi-platform, bao gồm các nền tảng Unix và Windows NT
- ❑ Thực thi là có sẵn với C/C++ và Fortran
- ❑ Có thể sử dụng dễ dàng và đơn giản – nhằm mục đích “tăng nhanh các ứng dụng song song”
- ❑ Microsoft thực thi riêng cho thread, không liên quan đến chuẩn UNIX POSIX hoặc OpenMP.

1/1/2015

Tính toán song song

54



Mô hình Message Passing

- ❑ Mô hình message passing có một số các đặc điểm sau:
 - ❑ Một tập các tác vụ sử dụng bộ nhớ cục bộ riêng của chúng khi tính toán. Nhiều tác vụ có thể cư trú trên cùng một máy tính hoặc qua nhiều máy tính.
 - ❑ Các tác vụ trao đổi dữ liệu thông qua truyền thông gửi và nhận message.
 - ❑ Truyền dữ liệu thường đòi hỏi các hoạt động phối hợp khi được thực thi bởi mỗi xử lý. Ví dụ, thao tác gửi phải được so khớp với thao tác nhận.

1/1/2015

Tính toán song song

55



Thực thi mô hình Message Passing: MPI

- ❑ Từ khía cạnh lập trình, thực thi message passing thường bao gồm một thư viện các chương trình còn được nhúng vào trong mã nguồn. Các lập trình viên có trách nhiệm xác định tất cả các xử lý song song.
- ❑ Trước đây, từ năm 1980 đã có một loạt các thư viện khác nhau. Các thực thi khác nhau đáng kể và gây khó khăn cho các lập trình viên phát triển các ứng dụng linh hoạt với các nền tảng phần cứng.
- ❑ 1992, Diễn đàn MPI được thành lập với mục tiêu chung duy nhất là xây dựng một chuẩn giao diện cho thực thi message passing.
- ❑ Phần 1 của **Message Passing Interface (MPI)** được phát hành năm 1994. Phần 2 (MPI-2) được phát hành 1996. Cả hai đặc tả này có sẵn tại địa chỉ: www.mcs.anl.gov/Projects/mpl/standard.html.

1/1/2015

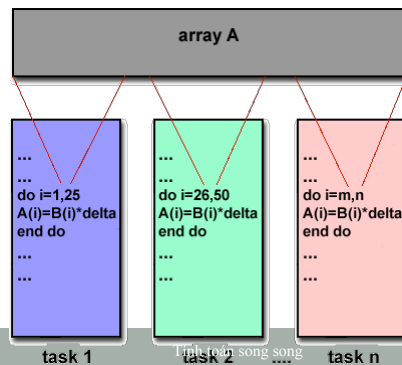
Tính toán song song

56



Thực thi mô hình Message Passing: MPI

- ❑ Hiện nay MPI là một chuẩn công nghiệp, nằm trong chuẩn "**de facto**" cho kết nối giữa các nút chạy một chương trình song song trên bộ nhớ phân tán.
- ❑ Với kiến trúc bộ nhớ chia sẻ, thực thi MPI thường không sử dụng giao tiếp các tác vụ qua mạng mà thay vào đó chúng sử dụng bộ nhớ chia sẻ (bản sao bộ nhớ) vì các lý do hiệu năng.
- ❑ Tập MPI thực thi bao gồm thư viện các thủ tục sao cho có thể gọi được từ các chương trình Fortran, C, C++ hay Ada



1/1/2015

task 1

Tinh toán song song

task n

57



Mô hình song song dữ liệu - Data Parallel

- ❑ Mô hình song song dữ liệu thể hiện qua các đặc điểm sau:
 - ❑ Hầu hết các công việc song song tập trung vào thực hiện các thao tác trên một bộ dữ liệu. Bộ dữ liệu này thường được tổ chức thành một cấu trúc chung như mảng hoặc khối (block).
 - ❑ Một tập các tác vụ làm việc trên cùng cấu trúc dữ liệu, tuy nhiên mỗi tác vụ làm việc trên các phần khác nhau của cùng cấu trúc dữ liệu.
 - ❑ Các tác vụ thực hiện cùng hành động trên phần vùng công việc của chúng, ví dụ "thêm 4 tới mỗi phần tử của mảng".
- ❑ Trong kiến trúc chia sẻ bộ nhớ, tất cả các tác vụ phải truy cập tới cấu trúc dữ liệu thông qua bộ nhớ toàn cục. Trên kiến trúc phân tán, cấu trúc dữ liệu được chia nhỏ và cứ trú như các "khối" trong bộ nhớ cục bộ của mỗi tác vụ.

1/1/2015

Tinh toán song song

58



Thực thi mô hình song song dữ liệu

- ❑ Lập trình với mô hình song song dữ liệu thường được thực hiện bằng cách viết một chương trình với các cấu trúc dữ liệu song song. Các cấu trúc này có thể gọi thư viện thủ tục song song dữ liệu hoặc nhận biết bởi các chỉ thị biên dịch của trình biên dịch song song dữ liệu.
- ❑ **Fortran 90 và 95 (F90, F95):** chuẩn mở rộng ISO/ANSI của Fortran 77.
 - ❑ Chứa mọi thứ của Fortran 77
 - ❑ Định dạng mã nguồn mới; bổ sung bộ ký tự
 - ❑ Bổ sung cấu trúc chương trình và các câu lệnh
 - ❑ Thêm biến, phương thức và các đối số
 - ❑ Kiểu con trỏ và cấp phát bộ nhớ động
 - ❑ Xử lý với dữ liệu kiểu mảng (mảng coi như các đối tượng)
 - ❑ Định quy và thêm các hàm
 - ❑ Và nhiều các đặc điểm mới khác
- ❑ Thực thi đều có sẵn trên hầu hết các nền tảng song song phổ biến nhất hiện nay

1/1/2015

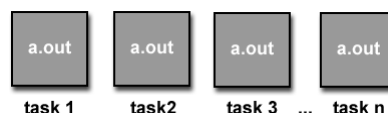
Tinh toán song song

59



Triển khai mô hình song song dữ liệu

- ❑ **High Performance Fortran (HPF):** Phần mở rộng Fortran 90 hỗ trợ lập trình song song dữ liệu.
 - ❑ Có mọi thứ trong Fortran 90
 - ❑ Thêm các chỉ thị biên dịch để điều hướng phân phối dữ liệu
 - ❑ Thêm phần định giá để có thể cải thiện tối ưu các mã lệnh
 - ❑ Thêm các cấu trúc dữ liệu song song (hiện nay nằm trong Fortran 95)
 - ❑ Thực thi đã có sẵn trên hầu hết các nền tảng song song phổ biến hiện nay.
- ❑ **Chỉ thị biên dịch (Compiler Directive):** Cho phép các lập trình viên chỉ định sự phân bố và sắp xếp dữ liệu. Hiện có sẵn trên hầu hết các nền tảng.
- ❑ Thực thi bộ nhớ phân tán theo mô hình này thường có trình biên dịch chuyển đổi chương trình thành code chuẩn kết hợp với việc gọi thư viện message passing (thường MPI) để phân phối dữ liệu cho tất cả các tiến trình. Tất cả các message được thực hiện “trong suốt” với lập trình viên.



1/1/2015

Tinh toán song song

60



Các mô hình khác

- ❑ Có các mô hình lập trình song song khác vẫn đang tiếp tục phát triển dù cho thế giới có sự thay đổi phần cứng và phần mềm máy tính.
- ❑ Có ba mô hình khác thông dụng được đề cập ở đây.
 - ❑ Dạng lai - Hybrid
 - ❑ Single Program Multiple Data
 - ❑ Multiple Program Multiple Data

1/1/2015

Tính toán song song

61



Hybryd

- ❑ Trong mô hình này, hai hay nhiều mô hình lập trình song song được kết hợp với nhau.
- ❑ Hiện nay, một ví dụ phổ biến của mô hình hybrid là kết hợp của mô hình MPI với mô hình thread (POSIX threads) hoặc mô hình chia sẻ bộ nhớ (OpenMP). Mô hình hybrid tận dụng môi trường phần cứng ngày càng phổ biến của các mạng máy tính được kết nối theo chuẩn SMP
- ❑ Một ví dụ phổ biến khác của mô hình hybrid là kết hợp dữ liệu song song với MPI. Như đã đề cập trong mô hình dữ liệu song song phần trước, thực thi dữ liệu song song (F90, HPF) trên kiến trúc bộ nhớ phân tán sử dụng MPI để truyền dữ liệu giữa các tác vụ và trong suốt với lập trình viên

1/1/2015

Tính toán song song

62



Single Program Multiple Data (SPMD)

- ❑ Single Program Multiple Data (SPMD):
- ❑ SPMD thực sự là mô hình lập trình “cấp cao” mà có thể xây dựng dựa trên việc kết hợp các mô hình lập trình song song đã đề cập trước đây.
- ❑ Một chương trình đơn được thực thi đồng thời bởi tất cả các tác vụ cùng một lúc.
- ❑ Tại mọi thời điểm, các tác vụ có thể được thực thi giống hoặc khác nhau các chỉ thị lệnh trong cùng chương trình.
- ❑ Các chương trình SPMD thường có logic lập trình cần thiết để cho phép các tác vụ khác nhau được phân nhánh hoặc thực thi có điều kiện một phần chương trình mà chúng được thiết kế ban đầu. Các tác vụ không nhất thiết phải thực thi toàn bộ - có thể chỉ một phần của nó.
- ❑ Tất cả các tác vụ có thể sử dụng dữ liệu khác nhau.

1/1/2015

Tính toán song song

63



Multiple Program Multiple Data (MPMD)

- ❑ Multiple Program Multiple Data (MPMD):
- ❑ Giống như SPMD, MPMD thực sự là một mô hình lập trình “mức cao” mà có thể được xây dựng dựa trên việc kết hợp các mô hình lập trình song song được đề cập trước đây.
- ❑ Các kiểu ứng dụng MPMD có nhiều tệp đối tượng thực thi (các chương trình). Khi một ứng dụng đang chạy song song, mỗi tác vụ có thể được thực thi giống hoặc khác nhau chương trình giống các tác vụ khác.
- ❑ Tất cả tác vụ có thể sử dụng dữ liệu khác nhau

1/1/2015

Tính toán song song


64



THIẾT KẾ CHƯƠNG TRÌNH SONG SONG

Designing Parallel Programs

1/1/2015 Tính toán song song 65



Nội dung

- ☐ Song song hoá tự động và thủ công
- ☐ Hiểu bài toán và chương trình
- ☐ Phân rã (Partitioning)
- ☐ Truyền thông (Communication)
- ☐ Đồng bộ (Synchronization)
- ☐ Các phụ thuộc dữ liệu (Data Dependencies)
- ☐ Cân bằng tải (Load Balancing)
- ☐ Tính hạt (Granularity)
- ☐ Đầu vào/Đầu ra (I/O)
- ☐ Các giới hạn và chi phí của lập trình song song
- ☐ Phân tích hiệu suất và hiệu chỉnh

1/1/2015 Tính toán song song 66



Nội dung

- ❑ Song song hoá tự động và thủ công
- ❑ Hiểu bài toán và chương trình
- ❑ Phân rã (Partitioning)
- ❑ Truyền thông (Communication)
- ❑ Đồng bộ (Synchronization)
- ❑ Các phụ thuộc dữ liệu (Data Dependencies)
- ❑ Cân bằng tải (Load Balancing)
- ❑ Tính hạt (Granularity)
- ❑ Đầu vào / Đầu ra
- ❑ Các giới hạn và chi phí của lập trình song song
- ❑ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

Tinh toán song song

67




-
- ❑ Thiết kế và phát triển các chương trình song song có đặc trưng là một *quá trình rất thủ công*. Lập trình viên thường chịu trách nhiệm cho cả hai việc: xác định và thực thi phần song song.
 - ❑ Code lập trình song song thường mất thời gian, phức tạp, dễ gặp lỗi và xử lý *lặp đi lặp lại*.
 - ❑ Vài năm trở lại đây, có rất nhiều các công cụ có sẵn trợ giúp các lập trình viên chuyển đổi chương trình tuần tự sang song song. Hầu hết các kiểu công cụ này là một trình biên dịch song song hoặc bộ tiền xử lý song song.

1/1/2015


Tinh toán song song

68



- ❑ Một trình biên dịch song song hoá thường làm việc theo 2 cách khác nhau:
 - ❑ Tự động toàn bộ
 - ❑ Trình biên dịch phân tích mã nguồn và nhận diện các thành phần cho sự song song.
 - ❑ Phân tích này bao gồm nhận diện các yếu tố cản trở sự song song và có thể là cả chi phí mà trong đó có hoặc không song song sẽ cải thiện hiệu năng tính toán.
 - ❑ Vòng lặp (do, for) là mục tiêu thường xuyên nhất cho song song hoá tự động.
 - ❑ Điều hướng bởi lập trình viên
 - ❑ Sử dụng "trình biên dịch điều hướng" hoặc các cờ biên dịch, lập trình viên có thể yêu cầu tường minh trình biên dịch thực hiện song song code.
 - ❑ Cũng có thể sử dụng kết hợp ở một mức độ nhất định với song song hoá tự động.

1/1/2015 Tính toán song song 69



- ❑ Nếu bạn đang bắt đầu với code tuần tự đang có và có thời gian hoặc ngân sách hạn chế thì song song hoá tự động có thể là một phương án. Tuy nhiên, có một số cảnh báo quan trọng khi áp dụng song song tự động:
 - ❑ Đưa ra kết quả sai
 - ❑ Hiệu năng thực sự có thể suy giảm
 - ❑ Ít linh hoạt hơn so với song song hoá thủ công
 - ❑ Giới hạn trong phần nhỏ của code (chủ yếu là các vòng lặp)
 - ❑ Có thể không thực hiện song song nếu phân tích bài toán có nhiều trở ngại hoặc code quá phức tạp
 - ❑ Hầu hết các công cụ tính toán song song tự động là cho Fortran
- ❑ Phần này áp dụng các phương pháp thủ công để viết mã song song.

1/1/2015 Tính toán song song 70



Nội dung

- ☐ Song song hoá tự động và thủ công
- ☐ Hiểu bài toán và chương trình
- ☐ Phân rã (Partitioning)
- ☐ Truyền thông (Communication)
- ☐ Đồng bộ (Synchronization)
- ☐ Các phụ thuộc dữ liệu (Data Dependencies)
- ☐ Cân bằng tải (Load Balancing)
- ☐ Tính hạt (Granularity)
- ☐ Đầu vào / Đầu ra
- ☐ Các giới hạn và chi phí của lập trình song song
- ☐ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

Tính toán song song

71



-
- ☐ Rõ ràng, bước đầu tiên trong phát triển phần mềm song song là hiểu bài toán mà bạn muốn giải quyết bằng song song. Nếu bạn đang bắt đầu bằng chương trình tuần tự, cũng cần thiết phải hiểu những đoạn code đã có.
 - ☐ Trước khi dành thời gian để thử phát triển một giải pháp song song cho một bài toán, cần xác định có hay không khả năng bài toán có thể giải quyết bằng song song.

1/1/2015

Tính toán song song

72



Ví dụ về bài toán song song

Tính toán năng lượng tiềm năng cho mỗi cấu trúc độc lập của một phân tử. Khi hoàn thành, tìm năng lượng tối thiểu cho mỗi cấu trúc đó.

- Bài toán này có thể được giải bằng song song. Mỗi phân tử được xác định là độc lập. Tính toán năng lượng tối thiểu cho mỗi cấu trúc cũng là một bài toán song song.

1/1/2015

Tính toán song song

73



Ví dụ về bài toán không song song

Tính dãy Fibonacci (1,1,2,3,5,8,13,21,...) theo công thức:

$$F(k+2) = F(k+1) + F(k)$$

- Đây là bài toán không thể song song vì việc tính toán dãy Fibonacci như trên đòi hỏi các tính toán phụ thuộc hơn là chỉ động lập trên một biểu thức. Tính toán giá trị của $k+2$ phụ thuộc vào $k+1$ và k . Ba biểu thức này không thể được tính toán độc lập và do đó nó không thể song song

1/1/2015

Tính toán song song

74



Xác định những «điểm nóng» của chương trình

- ❑ Nhận biết những phần công việc thực sự cần được giải quyết.
- ❑ Xem xét ở nhiều khía cạnh và sử dụng các công cụ phân tích hiệu năng có thể trợ giúp
- ❑ Tập trung vào những điểm nóng song song và bỏ qua những phần của chương trình mà chiếm dụng ít CPU.

1/1/2015

Tinh toán song song

75



Xác định các điểm thắt trong chương trình (*bottleneck*)

- ❑ Các phần chương trình song song không cân đối sẽ làm chậm hoặc gây ra việc song song bị chặn hoặc bị trì hoãn, ví dụ như vào/ra - I/O thường làm chương trình chạy chậm lại.
- ❑ Có thể cấu trúc lại chương trình hoặc dùng các thuật toán khác để giảm hoặc loại bỏ những phần chậm không cần thiết.

1/1/2015

Tinh toán song song

76



Các mối quan tâm khác

- ❑ Xác định những cản trở cho xử lý song song. Một cản trở phổ biến là *sự phụ thuộc dữ liệu* như ví dụ dãy Fibonacci ở trên.
- ❑ Nghiên cứu các thuật toán khác nếu có thể. Đây có thể là mối quan tâm quan trọng nhất khi thiết kế một ứng dụng song song.

1/1/2015

Tinh toán song song

77



Nội dung

- ❑ Song song hoá tự động và thủ công
- ❑ Hiểu bài toán và chương trình
- ❑ Phân rã (Partitioning)
- ❑ Truyền thông (Communication)
- ❑ Đồng bộ (Synchronization)
- ❑ Các phụ thuộc dữ liệu (Data Dependencies)
- ❑ Cân bằng tải (Load Balancing)
- ❑ Tính hạt (Granularity)
- ❑ Đầu vào / Đầu ra
- ❑ Các giới hạn và chi phí của lập trình song song
- ❑ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

Tinh toán song song

78



- ❑ Bước đầu tiên trong thiết kế chương trình song song là ngắt bài toán thành các “khối” riêng biệt của công việc để có thể phân phối tới nhiều tác vụ. Công việc này gọi là “phân rã” hoặc “phân tách”.
- ❑ Có hai cách phân chia công việc tính toán theo các tác vụ song song:
 - ❑ **Phân rã theo miền dữ liệu**
 - và
 - ❑ **Phân rã theo chức năng**

1/1/2015

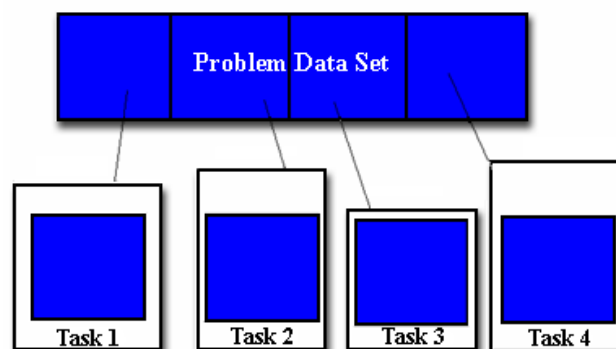
Tính toán song song

79



Phân rã theo miền dữ liệu

- ❑ Trong kiểu phân rã này, dữ liệu bài toán được phân rã. Mỗi tác vụ làm việc trên một phần của dữ liệu.



1/1/2015

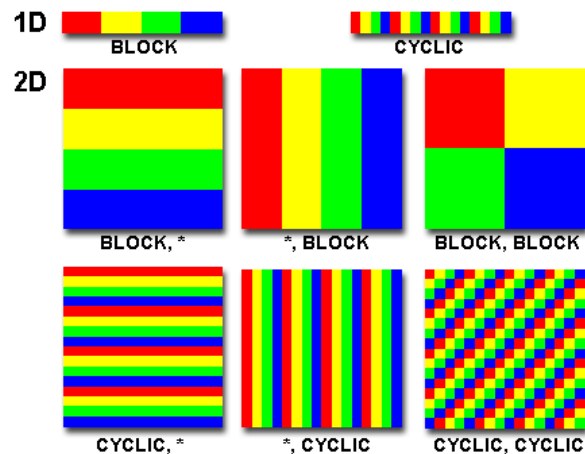
Tính toán song song

80



Phân vùng dữ liệu

- Có nhiều cách khác nhau để phân vùng dữ liệu



1/1/2015

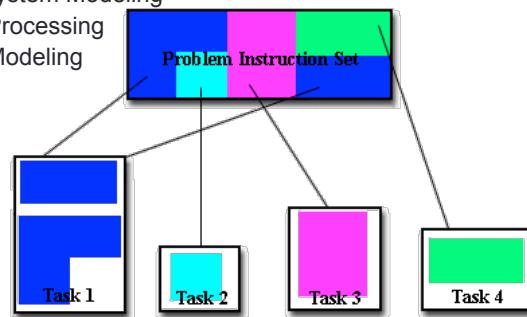
Tinh toán song song

81



Phân rã theo chức năng

- Trong cách tiếp cận này, tập trung vào tính toán sẽ được thực hiện, không quan tâm dữ liệu được thao tác. Bài toán được phân rã theo công việc phải được thực hiện. Mỗi tác vụ sau đó sẽ thực thi một phần của công việc tổng thể.
- Phân rã chức năng có thể thực thi tốt cho các bài toán có phân chia các tác vụ rõ ràng. Ví dụ như:
 - Mô hình hệ sinh thái - Ecosystem Modeling
 - Xử lý tín hiệu số - Signal Processing
 - Mô hình khí hậu - Climate Modeling



1/1/2015

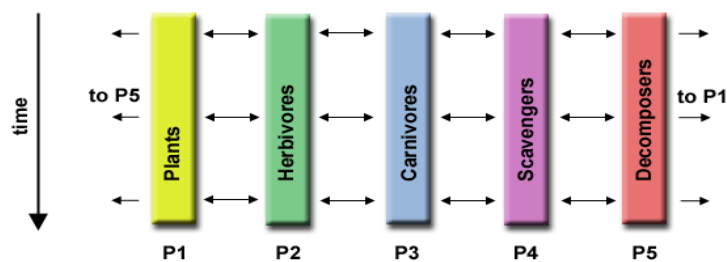
Tinh toán song song

82



Mô hình hệ sinh thái

- Mỗi chương trình sẽ tính toán quần thể của một nhóm nào đó mà sự phát triển mỗi nhóm phụ thuộc vào hàng xóm của chúng. Theo thời gian, mỗi tiến trình tính toán trạng thái hiện tại của chúng, rồi những thay đổi thông tin với các quần thể lân cận. Tất cả các tác vụ được tiến tới tính toán trạng thái ở thời gian kế tiếp.



1/1/2015

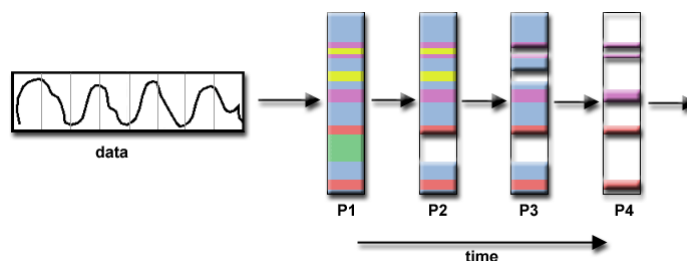
Tính toán song song

83



Xử lý tín hiệu số

- Một bộ dữ liệu tín hiệu âm thanh được truyền qua bốn bộ lọc tính toán. Mỗi bộ lọc là một xử lý riêng biệt. Dữ liệu sẽ chạy qua các bộ lọc từ thứ 1 đến thứ 2, thứ 3 và thứ tư.



1/1/2015

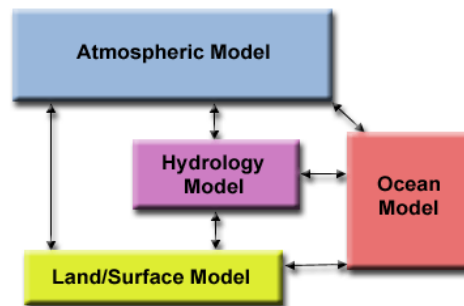
Tính toán song song

84



Mô hình khí hậu

- ❑ Mỗi thành phần của mô hình có thể chia thành các tác vụ riêng. Dấu mũi tên miêu tả cách trao đổi dữ liệu giữa các thành phần khi tính toán. Mô hình khí quyển tạo ra tốc độ gió được sử dụng trong mô hình đại dương, mô hình đại dương lại tạo ra dữ liệu nhiệt độ bề mặt biển mà được sử dụng trong mô hình khí quyển và tiếp tục như vậy



- ❑ Kết hợp hai kiểu phân rã bài toán là phổ biến và tự nhiên.

1/1/2015

Tính toán song song

85



Nội dung

- ❑ Song song hoá tự động và thủ công
- ❑ Hiểu bài toán và chương trình
- ❑ Phân rã (Partitioning)
- ❑ Truyền thông (Communication)
- ❑ Đồng bộ (Synchronization)
- ❑ Các phụ thuộc dữ liệu (Data Dependencies)
- ❑ Cân bằng tải (Load Balancing)
- ❑ Tính hạt (Granularity)
- ❑ Đầu vào / Đầu ra
- ❑ Các giới hạn và chi phí của lập trình song song
- ❑ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

Tính toán song song

86



Có cần truyền thông?

- ❑ Nhu cầu truyền thông giữa các tác vụ phụ thuộc vào bài toán của bạn
- ❑ **Không cần phải truyền thông**
 - ❑ Một vài kiểu bài toán có thể phân tách và thực thi song song mà hầu như không cần các tác vụ chia sẻ dữ liệu. Ví dụ tưởng tượng thao tác xử lý ảnh cần chuyển các pixel từ đen thành trắng thì chỉ cần đảo ngược màu sắc. Dữ liệu ảnh này có thể dễ dàng phân phối để xử lý song song nhiều tác vụ trên mỗi phần dữ liệu.
 - ❑ Những loại bài toán thường được gọi **song song đẹp (embarrassingly parallel)** bởi vì chúng thực hiện theo một “đường thẳng”, rất ít giao tiếp giữa các tác vụ được yêu cầu.
- ❑ **Cần phải truyền thông**
 - ❑ Hầu hết các ứng dụng song song thường không đơn giản và chúng yêu cầu các tác vụ chia sẻ dữ liệu với nhau. Ví dụ bài toán khuếch tán nhiệt 3-D yêu cầu một tác vụ phải biết nhiệt độ được tính toán bởi các tác vụ lân cận. Thay đổi dữ liệu hàng xóm có thể ảnh hưởng trực tiếp tới dữ liệu của tác vụ này.

1/1/2015

Tính toán song song

87



Các hệ số cần xem xét (1)

- ❑ Một số hệ số quan trọng để xem xét khi thiết kế truyền thông giữa các tác vụ trong chương trình.
- ❑ **Chi phí truyền thông**
 - ❑ Truyền thông giữa các tác vụ luôn được tính trong chi phí song song.
 - ❑ Các chu trình và tài nguyên của máy nên được sử dụng cho tính toán thay vì được sử dụng để đóng gói và truyền dữ liệu.
 - ❑ Truyền thông thường yêu cầu một vài kiểu đồng bộ giữa các tác vụ, mà có thể kết quả trong các tác vụ phải mất thời gian “đợi” thay vì làm việc.
 - ❑ Truyền thông cạnh tranh có thể bão hòa băng thông mạng, làm trầm trọng thêm vấn đề về hiệu năng.

1/1/2015

Tính toán song song

88



Các hệ số cần xem xét (2)

❑ Độ trễ và Băng thông

- ❑ **Độ trễ** là thời gian cần thiết để gửi một tin nhắn tối thiểu (0 byte) từ điểm A tới điểm B. Thường theo đơn vị micro giây.
- ❑ **Băng thông** là số lượng dữ liệu có thể truyền trên đơn vị thời gian. Thường biểu diễn bằng MB/giây.
- ❑ Gửi nhiều tin nhắn nhỏ có thể gây ra độ trễ chi phối chi phí truyền thông. Thường hiệu quả hơn đóng gói các gói tin nhỏ thành gói tin lớn hơn như vậy sẽ tăng hiệu quả băng thông truyền thông.

1/1/2015

Tinh toán song song

89



Các hệ số cần xem xét (3)

❑ Mức độ nhìn thấy của truyền thông

- ❑ Với mô hình Message Passing, các giao tiếp là rõ ràng và thường “nhìn thấy” và dưới điều khiển của lập trình viên.
- ❑ Với mô hình Data Parallel, các giao tiếp thường xảy ra trong suốt với lập trình viên, đặc biệt trên kiến trúc bộ nhớ phân tán. Lập trình viên thậm trí còn không thể biết chính xác cách nào giữa các tác vụ thực hiện giao tiếp với nhau như thế nào.

1/1/2015

Tinh toán song song

90



Các hệ số cần xem xét (4)

- ❑ **Truyền thông đồng bộ và không đồng bộ**
 - ❑ Truyền thông đồng bộ yêu cầu “bắt tay” giữa các tác vụ khi chia sẻ dữ liệu. Điều này có thể là chỉ ra rõ ràng trong code bởi lập trình viên, hoặc xảy ra ở mức thấp không được biết đến.
 - ❑ Truyền thông đồng bộ thường được gọi là truyền thông **khoá (blocking)** khi những công việc khác phải đợi cho đến khi các truyền thông hoàn thành.
 - ❑ Truyền thông không đồng bộ cho phép các tác vụ truyền dữ liệu độc lập với nhau. Ví dụ tác vụ 1 có thể chuẩn bị và gửi message cho tác vụ 2 và ngay lập tức bắt đầu làm công việc khác. Trong khi tác vụ 2 nhận được dữ liệu hay không không quan trọng.
 - ❑ Các giao tiếp không đồng bộ thường được gọi là truyền thông không khoá (**non-blocking**) khi các công việc khác có thể vẫn được làm khi các truyền thông đang diễn ra.
 - ❑ Các tính toán đan xen cùng với truyền thông mang lại lợi ích rất lớn cho truyền thông không đồng bộ.

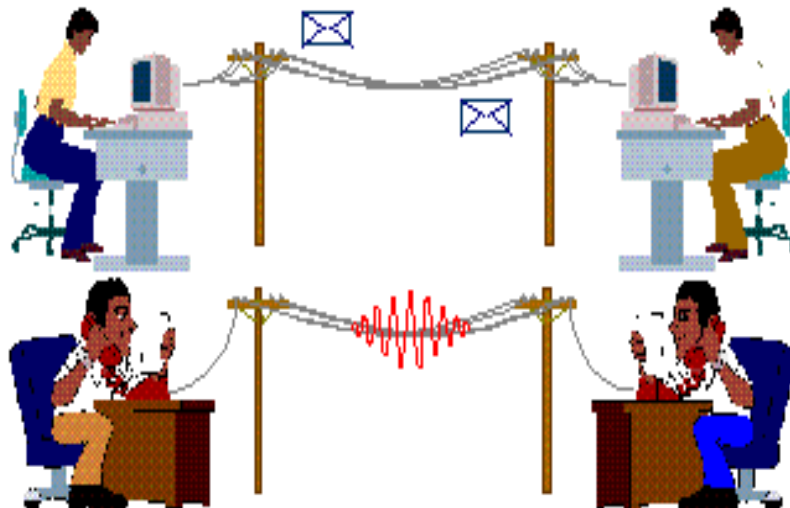
1/1/2015

Tính toán song song

91



Truyền thông đồng bộ & không đồng bộ



1/1/2015

Tính toán song song

92



Các hệ số cần xem xét (5)

- ❑ Phạm vi truyền thông
 - ❑ Biết được tác vụ nào phải giao tiếp với nhau là rất quan trọng trong thiết kế chương trình song song. Cả hai kiểu truyền thông mô tả bên dưới đều có thể thực thi đồng bộ hoặc không đồng bộ.
 - ❑ **Điểm tới điểm (Point-to-point)** – liên quan đến hai tác vụ với một tác vụ đóng vai trò người gửi/sản xuất dữ liệu, và tác vụ kia đóng vai trò là người nhận/người tiêu thụ.
 - ❑ **Tập hợp (Collective)** – liên quan đến chia sẻ giữa các tác vụ (nhiều hơn 2 tác vụ) mà thường được đặc tả như các thành viên trong một nhóm chung hay tập hợp.

1/1/2015

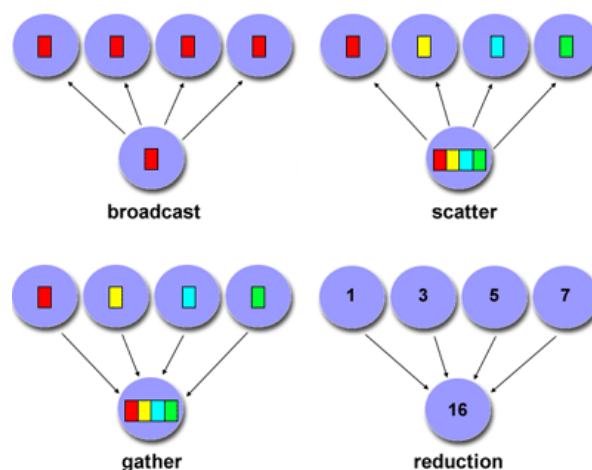
Tính toán song song

93



Các truyền thông dạng tập hợp

❑ Ví dụ



1/1/2015

Tính toán song song

94



Các hệ số cần xem xét (6)

- ❑ Hiệu quả truyền thông
 - ❑ Thường các lập trình viên sẽ lựa chọn các hệ số liên quan có thể ảnh hưởng tới hiệu suất truyền thông. Một trong số đó đề cập ở đây.
 - ❑ Mô hình thực thi nào nên được sử dụng? Sử dụng mô hình Message Pasing là một ví dụ, thực thi MPI có thể là nhanh hơn trên một nền tảng phần cứng hơn những thực thi khác.
 - ❑ Kiểu truyền thông nào nên được sử dụng? Như đã đề cập phần trước, truyền thông không đồng bộ có thể cải thiện hiệu suất tổng thể chương trình.
 - ❑ Phương tiện truyền thông mạng – Một vài nền tảng có thể cung cấp nhiều hơn một mạng cho các truyền thông. Cái nào là tốt nhất?

1/1/2015

Tính toán song song

95

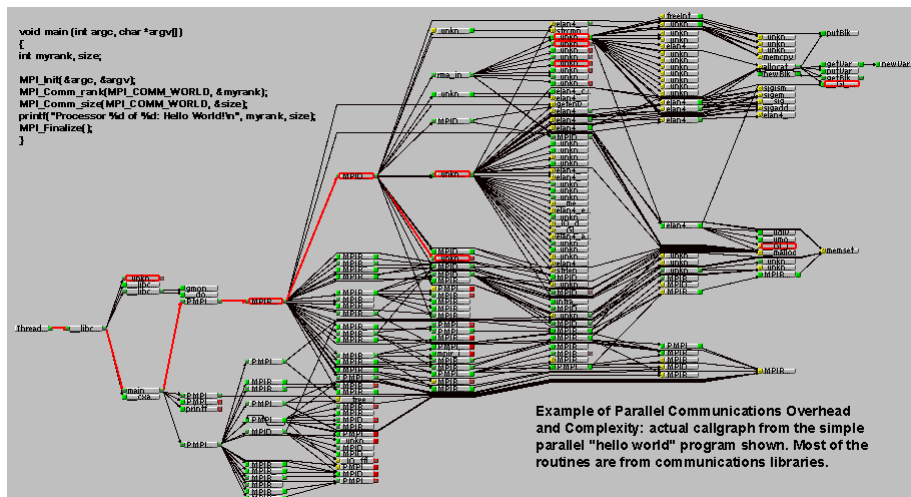


Các hệ số cần xem xét (7)

- ❑ Chi phí và độ phức tạp

```
void main (int argc, char *argv[])
{
    int myrank, size;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    printf("Processor %d of %d: Hello World!\n", myrank, size);
    MPI_Finalize();
}
```



Example of Parallel Communications Overhead and Complexity: actual callgraph from the simple parallel "hello world" program shown. Most of the routines are from communications libraries.

1/1/2015

Tính toán song song

96



Các hệ số cần xem xét (8)

- ❑ Cuối cùng, nhận ra rằng đây chỉ là một phần danh sách những điều cần xem xét !!! 😊

1/1/2015

Tinh toán song song

97



Nội dung

- ❑ Song song hoá tự động và thủ công
- ❑ Hiểu bài toán và chương trình
- ❑ Phân rã (Partitioning)
- ❑ Truyền thông (Communication)
- ❑ Đồng bộ (Synchronization)
- ❑ Các phụ thuộc dữ liệu (Data Dependencies)
- ❑ Cân bằng tải (Load Balancing)
- ❑ Tính hạt (Granularity)
- ❑ Đầu vào / Đầu ra
- ❑ Các giới hạn và chi phí của lập trình song song
- ❑ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

Tinh toán song song

98



Các kiểu đồng bộ dữ liệu

- ❑ **Rào chắn - Barrier**
 - ❑ Thường ám chỉ rằng các tác vụ có liên quan với nhau
 - ❑ Mỗi tác vụ thực thi công việc của chúng cho tới khi chạm tới ngưỡng. Rồi nó dừng lại hoặc bị "chặn lại".
 - ❑ Khi tác vụ sau cùng gặp barrier, tất cả các tác vụ là được đồng bộ.
 - ❑ Điều gì xảy ra ở đây. Thông thường một phần tuần tự của công việc phải được thực hiện. Trong các trường hợp khác, các tác vụ có thể giải phóng để tiếp tục công việc của chúng.
- ❑ **Khoá/Đèn tín hiệu - Lock /semaphore**
 - ❑ Có thể liên quan đến bất kỳ tác vụ nào
 - ❑ Thường được sử dụng để tuần tự (bảo vệ) truy cập tới dữ liệu toàn cục hoặc một phần của code. Chỉ một tác vụ tại một thời điểm có thể dùng (sở hữu) một lock/ semaphore/flag.
 - ❑ Tác vụ đầu tiên yêu cầu một khoá "thiết lập" nó. Tác vụ này có thể sau đó an toàn (tuần tự) truy cập dữ liệu hoặc code đã được bảo vệ.
 - ❑ Các tác vụ khác có thể cố gắng yêu cầu một khoá nhưng phải đợi cho đến khi tác vụ sử dụng khoá này giải phóng nó.
 - ❑ Có thể là blocking hoặc non-blocking
- ❑ **Các hoạt động truyền thông đồng bộ**
 - ❑ Liên quan chỉ những tác vụ thực hiện một thao tác truyền thông
 - ❑ Khi một tác vụ thực hiện một hoạt động truyền thông, một vài hình thức phối hợp được yêu cầu cùng với các tác vụ khác tham gia vào truyền thông. Ví dụ, trước khi một tác vụ có thể thực hiện thao tác gửi, trước tiên nó phải nhận một xác nhận từ tác vụ nhận rằng OK để gửi.

1/1/2015

Tinh toán song song

99



Nội dung

- ❑ Song song hoá tự động và thủ công
- ❑ Hiểu bài toán và chương trình
- ❑ Phân rã (Partitioning)
- ❑ Truyền thông (Communication)
- ❑ Đồng bộ (Synchronization)
- ❑ Các phụ thuộc dữ liệu (Data Dependencies)
- ❑ Cân bằng tải (Load Balancing)
- ❑ Tính hạt (Granularity)
- ❑ Đầu vào / Đầu ra
- ❑ Các giới hạn và chi phí của lập trình song song
- ❑ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

Tinh toán song song

100



Định nghĩa

- ❑ **Sự phụ thuộc** tồn tại giữa các câu lệnh chương trình khi thứ tự các câu lệnh thực thi ảnh hưởng tới kết quả chương trình.
- ❑ **Phụ thuộc dữ liệu (data dependence)** xảy ra khi các tác vụ khác nhau sử dụng cùng một hoặc nhiều vị trí lưu trữ.
- ❑ Tính phụ thuộc là rất quan trọng trong lập trình song song bởi vì chúng là một trong những cản trở chính trong xử lý song song.

1/1/2015

Tính toán song song

101



Ví dụ 1: Vòng lặp mang phụ thuộc dữ liệu

```
DO 500 J = MYSTART, MYEND
  A(J) = A(J-1) * 2.0500
CONTINUE
```

- ❑ Giá trị $A(J-1)$ phải được tính trước khi tính giá trị $A(J)$, do đó $A(J)$ thể hiện sự phụ thuộc dữ liệu vào $A(J-1)$. Xử lý song song bị ngăn cản.
- ❑ Nếu tác vụ 2 có $A(J)$ và tác vụ 1 có $A(J-1)$, tính toán đúng giá trị $A(j)$ cần thiết phải:
 - ❑ Kiến trúc bộ nhớ phân tán – tác vụ 2 phải chứa giá trị giá trị của $A(J-1)$ từ tác vụ 1 sau khi tác vụ 1 kết thúc tính toán của mình.
 - ❑ Kiến trúc bộ nhớ chia sẻ: tác vụ 2 phải đọc $A(J-1)$ sau khi tác vụ 1 cập nhật nó.

1/1/2015

Tính toán song song

102



Ví dụ 2: Vòng lặp độc lập phụ thuộc dữ liệu

tác vụ 1	tác vụ 2
-----	-----
$x = 2$	$x = 4$
.	.
.	.
$y = x**2$	$y = x**3$

- ❑ Như ví dụ trước tính song song bị cản trở. Giá trị của Y là phụ thuộc vào:
 - ❑ Kiến trúc bộ nhớ phân tán – khi giá trị của X được giao tiếp giữa các tác vụ.
 - ❑ Kiến trúc bộ nhớ chia sẻ - tác vụ sau cùng lưu trữ giá trị của X.
- ❑ Mặc dù nhận diện tất cả các phụ thuộc dữ liệu là quan trọng khi thiết kế chương trình song song, các phụ thuộc thực hiện vòng lặp là đặc biệt quan trọng vì các vòng lặp có thể là mục tiêu phổ biến nhất của những nỗ lực thực hiện song song.

1/1/2015

Tính toán song song

103



Làm thế nào để xử lý dữ liệu phụ thuộc?

- ❑ Các kiến trúc bộ nhớ phân tán – truyền dữ liệu theo yêu cầu tại các điểm đồng bộ.
- ❑ Các kiến trúc chia sẻ - đồng bộ các thao tác đọc/ghi giữa các tác vụ.

1/1/2015

Tính toán song song

104



Nội dung

- ❑ Song song hoá tự động và thủ công
- ❑ Hiểu bài toán và chương trình
- ❑ Phân rã (Partitioning)
- ❑ Truyền thông (Communication)
- ❑ Đồng bộ (Synchronization)
- ❑ Các phụ thuộc dữ liệu (Data Dependencies)
- ❑ Cân bằng tải (Load Balancing)
- ❑ Tính hạt (Granularity)
- ❑ Đầu vào / Đầu ra
- ❑ Các giới hạn và chi phí của lập trình song song
- ❑ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

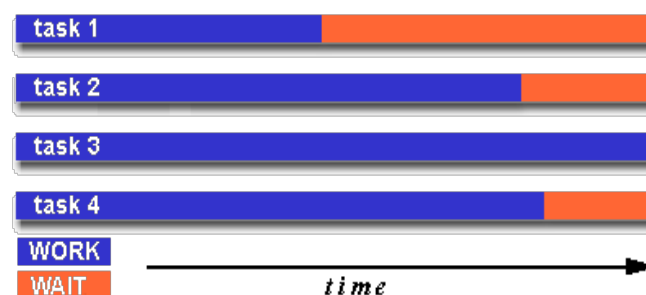
Tinh toán song song

105



Định nghĩa

- ❑ Cân bằng tải đề cập đến việc thực hiện phân phối công việc giữa các tác vụ để **tất cả** các tác vụ luôn “bận” trong **tất cả** thời gian. Nói cách khác là giảm thiểu thời gian rảnh rỗi giữa các tác vụ.
- ❑ Cân bằng tải rất quan trọng trong chương trình song song vì lý do hiệu suất. Ví dụ nếu các tác vụ phải chịu dừng lại ở một điểm đồng bộ, tác vụ chậm nhất sẽ xác định hiệu suất tổng thể.



1/1/2015

Tinh toán song song

106



Cách nào để đạt cân bằng tải? (1)

- ❑ **Bình đẳng phân chia công việc cho mỗi tác vụ nhận**
 - ❑ Với các thao tác mảng/ma trận ở đó mỗi tác vụ thực thi mỗi công việc tương tự nhau, phân phối đều dữ liệu đặt giữa các tác vụ.
 - ❑ Với các vòng lặp, các công việc được làm ở mỗi lần lặp là tương tự nhau, phân phối đồng đều số lần lặp cho các tác vụ.
 - ❑ Nếu một hệ thống các máy không đồng nhất với đặc điểm hiệu suất khác nhau được sử dụng, cần sử dụng một vài công cụ phân tích hiệu suất để phát hiện sự mất cân bằng tải. Hiệu chỉnh công việc cho phù hợp

1/1/2015

Tinh toán song song

107



Cách nào để đạt cân bằng tải? (2)

- ❑ **Sử dụng khởi gán công việc động**
 - ❑ Một số lớp bài toán dẫn đến sự mất cân bằng tải ngay cả khi dữ liệu là phân phối đồng đều giữa các tác vụ:
 - ❑ Các mảng thưa – một vài tác vụ sẽ có dữ liệu thực tế để làm việc trong khi những tác vụ khác hầu như "zeros".
 - ❑ Các phương pháp lưới thích ứng – một vài tác vụ có thể cần tinh chỉnh lưới của chúng trong khi những tác vụ khác thì không.
 - ❑ Các mô phỏng N-body - ở đó một số hạt có thể di chuyển tới/ từ miền tác vụ gốc tới các tác vụ khác. Ở đó các hạt thuộc sở hữu của một số tác vụ đòi hỏi làm nhiều công việc hơn những cái được sở hữu bởi các tác vụ khác.
 - ❑ Khi số lượng công việc của mỗi tác vụ là thay đổi hoặc không thể dự đoán được, sẽ hữu ích nếu dùng cách tiếp cận **lập lịch các công việc (scheduler - task pool)**. Với các tác vụ kết thúc công việc của nó, hàng đợi sẽ nhận một phần công việc mới.
 - ❑ Sẽ trở nên rất cần thiết thiết kế thuật toán tìm ra và điều khiển không cân bằng tải như chúng xảy ra động trong code

1/1/2015

Tinh toán song song

108



Nội dung

- ❑ Song song hoá tự động và thủ công
- ❑ Hiểu bài toán và chương trình
- ❑ Phân rã (Partitioning)
- ❑ Truyền thông (Communication)
- ❑ Đồng bộ (Synchronization)
- ❑ Các phụ thuộc dữ liệu (Data Dependencies)
- ❑ Cân bằng tải (Load Balancing)
- ❑ Tính hạt (Granularity)
- ❑ Đầu vào / Đầu ra
- ❑ Các giới hạn và chi phí của lập trình song song
- ❑ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

Tinh toán song song

109



Định nghĩa

- ❑ Tỷ lệ tính toán/truyền thông:
 - ❑ Trong tính toán song song, tính hạt là đại lượng đo chất lượng của tỷ lệ tính toán/truyền thông.
 - ❑ Giai đoạn tính toán thường tách biệt với giai đoạn truyền thông bằng các sự kiện đồng bộ.
- ❑ Song song hạt mịn
- ❑ Song song hạt thô

1/1/2015

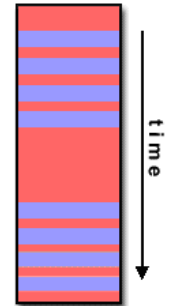
Tinh toán song song

110



Song song hạt mịn

- ❑ Số lượng tương đối nhỏ công việc tính toán được làm giữa các sự kiện truyền thông
- ❑ Tỷ lệ tính toán/truyền thông thấp
- ❑ Dễ dàng cân bằng tải
- ❑ Âm chỉ chi phí truyền thông cao hơn và ít có cơ hội để nâng cao hiệu suất
- ❑ Nếu tính hạt là quá mịn, có thể chi phí yêu cầu cho các giao tiếp và đồng bộ giữa các tác vụ mất nhiều thời gian hơn so với tính toán.



■ communication
■ computation

1/1/2015

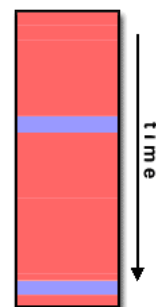
Tính toán song song

111



Song song hạt thô

- ❑ Số lượng tương đối lớn các công việc tính toán được thực hiện giữa các sự kiện truyền thông/đồng bộ
- ❑ Tỷ lệ tính toán cao/truyền thông
- ❑ Âm chỉ có nhiều cơ hội cho hiệu suất tăng
- ❑ Khó để cân bằng tải một cách hiệu quả



■ communication
■ computation

1/1/2015

Tính toán song song

112



Cái nào là tốt nhất?

- ❑ Tỷ lệ hạt hiệu quả nhất là phụ thuộc vào thuật toán và môi trường phần cứng mà nó chạy trên đó.
- ❑ Trong hầu hết các trường hợp, chi phí gắn với các truyền thông và đồng bộ là cao hơn so với tốc độ thực hiện thì sẽ rất thuận lợi để có được tính hạt thô.
- ❑ Song song hạt mịn có thể giúp giảm các chi phí do tải không cân bằng.

1/1/2015

Tính toán song song

113



Nội dung

- ❑ Song song hoá tự động và thủ công
- ❑ Hiểu bài toán và chương trình
- ❑ Phân rã (Partitioning)
- ❑ Truyền thông (Communication)
- ❑ Đồng bộ (Synchronization)
- ❑ Các phụ thuộc dữ liệu (Data Dependencies)
- ❑ Cân bằng tải (Load Balancing)
- ❑ Tính hạt (Granularity)
- ❑ Đầu vào/Đầu ra (I/O)
- ❑ Các giới hạn và chi phí của lập trình song song
- ❑ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

Tính toán song song

114



Vấn đề

- ❑ Các thao tác I/O thường được coi là cản trở xử lý song song
- ❑ Các hệ thống I/O song song vẫn yếu kém hoặc không có sẵn cho tất cả các nền tảng
- ❑ Trong môi trường mà các tác vụ có cùng không gian tệp, các thao tác ghi sẽ dẫn tới tệp bị ghi đè.
- ❑ Các thao tác đọc sẽ bị ảnh hưởng bởi khả năng phục vụ các file được điều khiển bởi nhiều yêu cầu đọc tại cùng thời điểm.
- ❑ I/O phải được giám sát qua mạng (NFS, không cục bộ) để tránh gây tắc nghẽn nghiêm trọng

1/1/2015

Tinh toán song song

115



Đã có

- ❑ Một vài hệ thống tập tin song song đã ra đời, ví dụ:
 - ❑ GPFS: General Parallel File System cho AIX (IBM)
 - ❑ Lustre: cho Linux clusters (Cluster File Systems, Inc.)
 - ❑ PVFS/PVFS2: Parallel Virtual File System cho Linux clusters (Clemson/Argonne/Ohio State/others)
 - ❑ PanFS: Panasas ActiveScale File System for Linux clusters (Panasas, Inc.)
 - ❑ HP SFS: HP StorageWorks Scalable File Share. Lustre dựa trên hệ thống song song tệp tin (Global File System choLinux) sản phẩm từ HP
- ❑ Đặc tả giao diện lập trình song song I/O cho MPI đã có sẵn từ năm 1996, một phần của MPI-2. Nhà cung cấp và các thực thi “miễn phí” hiện nay thường có sẵn.

1/1/2015

Tinh toán song song

116



Một vài tùy chọn

- ❑ Quy luật #1: Giảm tổng thể truy cập I/O càng nhiều càng tốt
- ❑ Cô lập I/O trong những phần công việc tuần tự, và rồi sử dụng các giao tiếp song song để phân phối dữ liệu cho các tác vụ song song. Ví dụ 1, tác vụ 1 có thể đọc một tệp tin đầu vào và sau đó truyền dữ liệu cần thiết cho các tác vụ khác. Tương tự như vậy, tác vụ 1 cũng có thể thực thi thao tác ghi dữ liệu cần thiết từ tất cả các tác vụ khác.
- ❑ Với bộ nhớ phân tán chia sẻ không gian tệp, thực thi I/O ở cục bộ, không chia sẻ không gian tệp. Ví dụ mỗi bộ xử lý có thể có không gian tệp */tmp*. Điều này thường hiệu quả hơn nhiều thực thi I/O thông qua mạng của một thư mục dùng chung.
- ❑ Tạo các tên tệp duy nhất cho mỗi tác vụ vào/ra tệp

1/1/2015

Tính toán song song

117




Nội dung

- ❑ Song song hoá tự động và thủ công
- ❑ Hiểu bài toán và chương trình
- ❑ Phân rã (Partitioning)
- ❑ Truyền thông (Communication)
- ❑ Đồng bộ (Synchronization)
- ❑ Các phụ thuộc dữ liệu (Data Dependencies)
- ❑ Cân bằng tải (Load Balancing)
- ❑ Tính hạt (Granularity)
- ❑ Đầu vào / Đầu ra
- ❑ Các giới hạn và chi phí của lập trình song song
- ❑ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

Tính toán song song

118

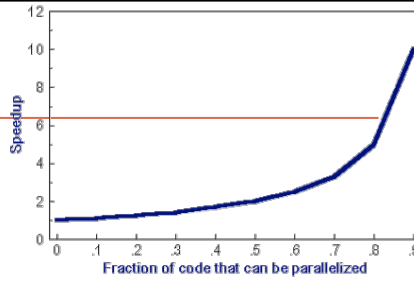


Luật Amdahl


■ Phát biểu luật *Amdahl*: khả năng tăng tốc độ chương trình được xác định bởi tỉ lệ code (P) có thể được song song:

$$\text{speedup} = \frac{1}{1 - P}$$

- ❑ Nếu tỉ lệ song song P = 0 thì tốc độ = 1. Nếu tất cả code là được song song, P = 1 thì tốc độ là vô hạn (theo lý thuyết).
- ❑ Nếu 50% của code được song song thì tốc độ tối đa = 2, nghĩa là chương trình chạy nhanh hơn 2 lần.



1/1/2015
Tính toán song song
119



Luật Amdahl

❑ Khi số bộ vi xử lý thực thi song song các phần công việc, mối quan hệ có thể được mô hình bởi

$$\text{speedup} = \frac{1}{\frac{P}{N} + S}$$

❑ Trong đó P = tỉ lệ code song song, N = số bộ vi xử lý và S = tỉ lệ code tuần tự

1/1/2015
Tính toán song song
120



Luật Amdahl

- Rõ ràng có những giới hạn khi mở rộng song song, ví dụ với $P = .50$, $.90$ và $.99$ (tỉ lệ 50%, 90% và 99% của phần code song song), thực nghiệm cho kết quả:

N	speedup		
	P = .50	P = .90	P = .99
10	1.82	5.26	9.17
100	1.98	9.17	50.25
1000	1.99	9.91	90.99
10000	1.99	9.91	99.02

1/1/2015

Tinh toán song song

121



Luật Amdahl

- Tuy nhiên một vài bài toán chứng minh việc tăng hiệu suất khi tăng kích thước bài toán, ví dụ:
 - **Các tính toán lưới 2D mất 85 giây chiếm 85%**
 - **Phần tuần tự mất 15 giây chiếm 15%**
- Chúng ta có thể tăng kích thước bài toán bằng việc tăng gấp đôi kích thước lưới. Thời gian thực thi:
 - **Các tính toán lưới 2D mất 680 giây chiếm 97.84%**
 - **Phần tuần tự mất 15 seconds chiếm 2.16%**
- Các bài toán tăng tỉ lệ phần trăm song song với kích thước của chúng có nhiều khả năng mở rộng hơn những bài toán cố định phần trăm thời gian song song.

1/1/2015

Tinh toán song song

122



Độ phức tạp

- ❑ Nói chung, các ứng dụng song song là phức tạp hơn nhiều so với các ứng dụng dạng tuần tự (xét về cùng loại ứng dụng và thuật toán). Không chỉ riêng nhiều luồng chỉ thị lệnh được thực thi tại một thời điểm mà còn có cả các luồng dữ liệu trao đổi giữa chúng.
- ❑ Các chi phí về độ phức tạp được đo bằng thời gian lập trình theo mọi khía cạnh của vòng đời phát triển phần mềm:
 - ❑ Thiết kế
 - ❑ Viết code
 - ❑ Gỡ rối
 - ❑ Điều chỉnh
 - ❑ Bảo trì
- ❑ Tuân thủ tốt phương pháp phát triển phần mềm là cần thiết khi làm việc với các ứng dụng song song.

1/1/2015

Tính toán song song

123



Tính tương thích

- ❑ Nhờ việc chuẩn hoá trong một số API như MPI, POSIX threads, HPF và OpenMP, các vấn đề tương thích của các chương trình song song không còn là vấn đề nghiêm trọng trong những năm qua. Tuy nhiên...
- ❑ Thường các vấn đề tương thích của một chương trình tuần tự kết hợp với chương trình song song. Ví dụ như nếu bạn sử dụng những “cải tiến” của nhà sản xuất cho Fortran, C hay C++, lúc này tính tương thích sẽ là một vấn đề.
- ❑ Mặc dù các chuẩn tồn tại trong nhiều API, nhưng việc triển khai sẽ khác nhau ở một số chi tiết, đôi khi đòi hỏi phải sửa đổi mã nguồn để thực hiện tương thích.
- ❑ Hệ điều hành có thể đóng một vai trò quan trọng trong vấn đề tính tương thích của mã nguồn.
- ❑ Các kiến trúc phần cứng có các đặc điểm rất khác nhau và có thể ảnh hưởng đến tính tương thích.

1/1/2015

Tính toán song song

124



Yêu cầu tài nguyên

- ❑ Mục đích chính của chương trình song song là giảm thời gian thực thi, tuy nhiên để thực hiện điều này, yêu cầu nhiều thời gian của CPU. Ví dụ code song song chạy 1 giờ trên 8 bộ xử lý thực chất nó sử dụng 8 giờ thời gian của CPU.
- ❑ Số lượng bộ nhớ đòi hỏi cho code song song có thể lớn hơn code tuần tự, do cần tái tạo dữ liệu và cho các chi phí kết hợp của các thư viện hỗ trợ song song và các hệ thống phụ trợ.
- ❑ Đối với các chương trình song song nhỏ, có thể về mặt hiệu suất nó kém hơn thực thi bằng tuần tự tương tự. Chi phí trên liên quan đến việc thiết lập môi trường song song, khởi tạo tác vụ, các truyền thông và kết thúc tác vụ có thể bao gồm một phần đáng kể thời gian thực thi

1/1/2015

Tinh toán song song

125



Khả năng mở rộng

- ❑ Khả năng mở rộng hiệu năng của một chương trình tính toán song song phụ thuộc vào nhiều yếu tố liên quan đến nhau. Không đơn giản là việc thêm nhiều máy tính thì sẽ cho kết quả tốt hơn.
- ❑ Một thuật toán có thể có những giới hạn vốn có để mở rộng. Ở một vài điểm, việc thêm tài nguyên gây ra hiệu suất giảm.
- ❑ Các yếu tố phần cứng đóng một vai trò quan trọng trong khả năng mở rộng, ví dụ:
 - ❑ Bảng thông bus của bộ nhớ và CPU trên các máy SMP
 - ❑ Bảng thông mạng truyền thông
 - ❑ Số lượng bộ nhớ có sẵn trên các máy hoặc thiết lập trên các máy
 - ❑ Tốc độ xử lý đồng hồ
- ❑ Các thư viện hỗ trợ song song và các phần mềm hệ thống phụ trợ có thể giới hạn khả năng mở rộng độc lập của ứng dụng.

1/1/2015

Tinh toán song song

126



Nội dung

- ❑ Song song hoá tự động và thủ công
- ❑ Hiểu bài toán và chương trình
- ❑ Phân rã (Partitioning)
- ❑ Truyền thông (Communication)
- ❑ Đồng bộ (Synchronization)
- ❑ Các phụ thuộc dữ liệu (Data Dependencies)
- ❑ Cân bằng tải (Load Balancing)
- ❑ Tính hạt (Granularity)
- ❑ Đầu vào / Đầu ra
- ❑ Các giới hạn và chi phí của lập trình song song
- ❑ Phân tích hiệu suất và hiệu chỉnh

1/1/2015

Tinh toán song song

127




-
- ❑ Gỡ rối, lần vết và phân tích thực thi chương trình song song là một thách thức đáng kể hơn so với các chương trình tuần tự.
 - ❑ Hiện nay có sẵn một số công cụ cho phép theo dõi thực thi và phân tích.
 - ❑ Hãy bắt đầu với tài liệu hướng dẫn công cụ phân tích hiệu năng: [Performance Analysis Tools Tutorial](#)
 - ❑ Công việc vẫn còn phải tiếp tục, đặc biệt là trong phần khả năng mở rộng hệ thống tính toán song song.

1/1/2015

Tinh toán song song


128



CÁC VÍ DỤ SONG SONG

Parallel Examples


1/1/2015 Tính toán song song 129



Nội dung

- ☐ Xử lý mảng
- ☐ Tính toán số PI
- ☐ Phương trình nhiệt đơn giản (Simple Heat Equation)
- ☐ Phương trình sóng 1-D (1-D Wave Equation)

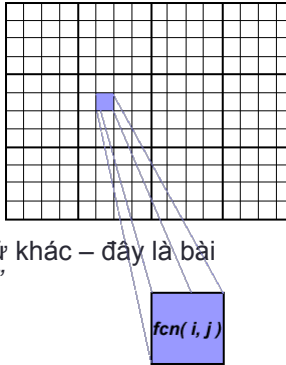
1/1/2015 Tính toán song song 130




Xử lý mảng

- ❑ Ví dụ này giới thiệu cách tính toán các phần tử mảng hai chiều (2-D) với việc tính toán trên mỗi phần tử mảng là độc lập với các phần tử mảng khác.
- ❑ Chương trình tuần tự này tính các phần tử theo cách tuần tự.
- ❑ Đoạn code tuần tự có thể mô tả như sau:


```
do j = 1,n
  do i = 1,n
    a(i,j) = fcn(i,j)
  end do
end do
```
- ❑ Tính toán các phần tử là độc lập với các phần tử khác – đây là bài toán *song song “đẹp”* hay *song song “hoàn hảo”*
- ❑ Bài toán này cần được nghiên cứu sâu hơn.



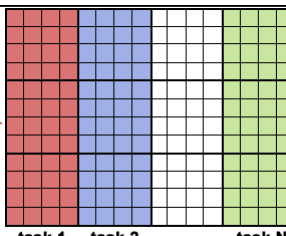
1/1/2015
Tính toán song song
131



Xử lý mảng: giải pháp 1

- ❑ Các phần tử mảng được chia thành các mảng con cho mỗi bộ xử lý riêng.
- ❑ Các tính toán là độc lập giữa các phần tử mảng nên không cần giao tiếp giữa các tác vụ.
- ❑ Phân chia các mảng con có thể theo số một số tiêu chí khác nhau ví dụ như theo bước nhảy trên số tác vụ. Phân chia theo đơn vị bước nhảy sẽ tối ưu được việc sử dụng cache/memory.
- ❑ Việc lựa chọn ngữ cảnh phân phối các mảng con phụ thuộc vào ngôn ngữ lập trình. Xem thêm [Block - Cyclic Distributions Diagram](#).
- ❑ Sau khi mảng bị chia, mỗi tác vụ thực hiện trên các phần của vòng lặp tương ứng với dữ liệu riêng. Ví dụ khối dữ liệu phân bố này được thực hiện:


```
do j = mystart, myend
  do i = 1,n
    a(i,j) = fcn(i,j)
  end do
end do
```
- ❑ Chú ý rằng các biến lặp vòng lặp for ngoài cùng khác với phần thuật toán tuần tự.



1/1/2015
Tính toán song song
132



Xử lý mảng: giải pháp 1

Phương pháp triển khai khả thi

- ❑ Thực thi theo mô hình SPMD.
- ❑ Tiến trình chủ (Master) khởi tạo mảng, gửi dữ liệu tới các tiến trình con (worker) và nhận kết quả.
- ❑ Các tiến trình worker nhận thông tin, thực thi dữ liệu chia sẻ tính toán và gửi kết quả về master.
- ❑ Thuật toán: **màu đỏ** là những phần thay đổi cho xử lý song song

1/1/2015

Tính toán song song

133



Xử lý mảng: giải pháp 1

Phương pháp triển khai khả thi

```

find out if I am MASTER or WORKER
if I am MASTER
    initialize the array
    send each WORKER info on part of array it owns
    send each WORKER its portion of initial array

    receive from each WORKER results
else if I am WORKER
    receive from MASTER info on part of array I own
    receive from MASTER my portion of initial array

    # calculate my portion of array
    do j = my first column, my last column
    do i = 1, n
        a(i, j) = fcn(i, j)
    end do
    end do

    send MASTER results
endif
  
```

1/1/2015

Tính toán song song

134



Xử lý mảng: giải pháp 2

Pool of Tasks

- ❑ Giải pháp 1 về xử lý mảng là giải pháp cân bằng tải tĩnh:
 - ❑ Mỗi tác vụ có một số lượng các công việc cố định phải làm
 - ❑ Thời gian nhàn rỗi sẽ đáng kể giữa các bộ xử lý nhanh với các bộ xử lý chạy chậm hơn do đó sẽ ảnh hưởng tới hiệu suất tổng thể.
- ❑ Cân bằng tải tĩnh thường không là mối quan tâm lớn nếu các tác vụ được thực thi cùng số lượng công việc trên các máy giống nhau..
- ❑ Nếu bạn gặp vấn đề với cân bằng tải (một vài công việc sẽ thực thi nhanh hơn những cái khác), sẽ có lợi nếu bạn sử dụng chiến lược “pool of tasks”

1/1/2015

Tinh toán song song

135



Xử lý mảng: giải pháp 2

Chiến lược Pool of Tasks

- ❑ Hai bộ xử lý được thực thi
- ❑ Tiến trình Master:
 - ❑ Giữ các tác vụ cho các tiến trình worker làm
 - ❑ Gửi worker tác vụ khi được yêu cầu
 - ❑ Thu thập kết quả từ các worker
- ❑ Tiến trình Worker: lặp lại các công việc sau
 - ❑ Nhận tác vụ từ tiến trình master
 - ❑ Thực hiện tính toán
 - ❑ Gửi các kết quả tới master
- ❑ Các tiến trình worker không biết trước thời gian chạy mà chúng sẽ xử lý và có bao nhiêu tác vụ chúng phải thực hiện.
- ❑ Cân bằng tải động xảy ra lúc chạy: các tác vụ nhanh hơn sẽ nhận thêm công việc để làm.
- ❑ Thuật giải: **màu đỏ** là những thay đổi cho phần xử lý song song.

1/1/2015

Tinh toán song song

136



Xử lý mảng: Giải pháp 2 cho chiến lược Pool of Tasks

```

find out if I am MASTER or WORKER

if I am MASTER

    do until no more jobs
        send to WORKER next job
        receive results from WORKER
    end do

    tell WORKER no more jobs

else if I am WORKER

    do until no more jobs
        receive from MASTER next job

        calculate array element:  $a(i,j) = fcn(i,j)$ 

        send results to MASTER
    end do

endif
  
```

1/1/2015

Tinh toán song song

137



Tính toán số PI

- ❑ Giá trị của PI có thể được tính theo các cách khác nhau. Hay xem xét một phương pháp sau để tính xấp xỉ số PI:
 - ❑ Một vòng tròn nội tiếp trong một hình vuông
 - ❑ Phát sinh ngẫu nhiên các điểm trong hình vuông
 - ❑ Xác định số điểm rơi trong hình vuông và các điểm rơi vào trong hình tròn
 - ❑ Đặt r bằng số điểm trong hình tròn chia cho số điểm trong hình vuông
 - ❑ $PI \sim 4 * r$
- ❑ Chú ý rằng càng phát sinh nhiều điểm, độ chính xác của số PI càng cao

1/1/2015

Tinh toán song song

138



Thảo luận

- ❑ Trong ví dụ về “pool of tasks”, mỗi tác vụ tính toán một phần tử riêng lẻ của mảng như một job. Tỷ lệ tính toán trên giao tiếp các tác vụ được coi là tính hạt mịn (finely granular)
- ❑ Các giải pháp tính hạt mịn phải chịu chi phí giao tiếp nhiều hơn để giảm bớt thời gian nhàn rỗi của tác vụ.
- ❑ Một giải pháp tối ưu hơn nữa có thể là phân phối nhiều công việc với mỗi job.

1/1/2015

Tính toán song song

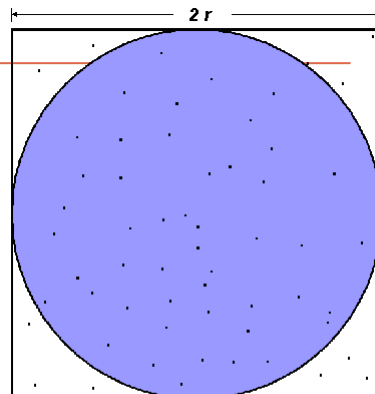
139



Thuật toán tuần tự

```

npoints = 10000
circle_count = 0
do j = 1,npoints
    generate 2 random numbers between
    0 and 1
    xcoordinate = random1 ;
    ycoordinate = random2
    if (xcoordinate, ycoordinate)
    inside circle then circle_count =
    circle_count + 1
end do
PI = 4.0*circle_count/npoints
  
```




$$\begin{aligned}
 A_S &= (2r)^2 = 4r^2 \\
 A_C &= \pi r^2 \\
 \pi &= 4 \times \frac{A_C}{A_S}
 \end{aligned}$$

- Chú ý: hầu hết thời gian chạy chương trình này dành cho vòng lặp
- Dẫn đến một giải pháp song song “đẹp”:
 - Khối lượng tính toán lớn
 - Tối thiểu giao tiếp
 - Tối thiểu I/O

1/1/2015

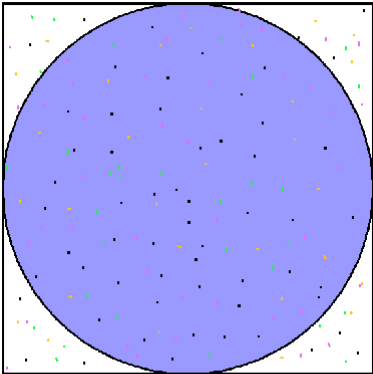
Tính toán song song

140



Tính toán số PI


Giải pháp song song



■ task 1
■ task 2
■ task 3
■ task 4

- ❑ Chiến thuật song song: ngắt vòng lặp thành các phần mà có thể được thực thi bởi các tác vụ khác nhau.
- ❑ Với tác vụ này số PI xấp xỉ:
 - ❑ Mỗi tác vụ thực hiện phần công việc lặp trong vòng một số lần
 - ❑ Mỗi tác vụ có thể làm phần việc của mình mà không cần yêu cầu thông tin từ các tác vụ khác.
 - ❑ Sử dụng mô hình SPMD. Một tác vụ đóng vai trò là master và thu thập kết quả.
- ❑ Thuật giải: **màu đỏ** là những thay đổi cho xử lý song song

1/1/2015
Tính toán song song
141



Tính toán số PI

Giải pháp song song

```

circle_count = 0

p = number of tasks
num = npoints/p

find out if I am MASTER or WORKER

do j = 1,num
  generate 2 random numbers between 0 and 1
  xcoordinate = random1 ; ycoordinate = random2
  if (xcoordinate, ycoordinate) inside circle
    then circle_count = circle_count + 1
  end do

if I am MASTER

  receive from WORKERS their circle_counts
  compute PI (use MASTER and WORKER calculations)

else if I am WORKER

  send to MASTER circle_count

endif

```

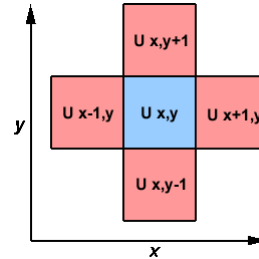
1/1/2015
Tính toán song song
142



Phương trình nhiệt đơn giản

Simple Heat Equation

- ❑ Hầu hết các bài toán trong tính toán song song đòi hỏi giao tiếp giữa các tác vụ. Một trong số các bài toán phổ biến là giao tiếp với các tác vụ “hàng xóm”.
- ❑ Phương trình nhiệt mô tả nhiệt độ thay đổi theo thời gian, khởi tạo phân bố nhiệt độ và các điều kiện biên.
- ❑ Một lược đồ hữu hạn phân biệt được sử dụng để giải quyết phương trình nhiệt số trên một vùng ô vuông.
- ❑ Khởi tạo nhiệt độ là 0 ở đường biên và cao ở giữa.
- ❑ Nhiệt độ đường biên được giữ là 0
- ❑ Bài toán hoàn toàn rõ ràng, mỗi bước thuật toán được sử dụng, các phần tử của một mảng 2 chiều đại diện cho nhiệt độ tại các điểm trên hình vuông.



1/1/2015

Tính toán song song

143



Phương trình nhiệt đơn giản

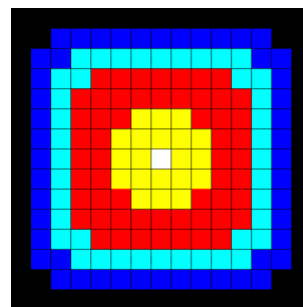
- ❑ Việc tính toán một phần tử phụ thuộc vào các giá trị của phần tử hàng xóm.

$$\begin{aligned}
 U_{x,y} &= U_{x,y} \\
 &+ C_x * (U_{x+1,y} + U_{x-1,y} - 2 * U_{xy}) \\
 &+ C_y * (U_{x,y+1} + U_{x,y-1} - 2 * U_{x,y})
 \end{aligned}$$

- ❑ Code chương trình tuần tự có thể:

```

do iy = 2, ny - 1
do ix = 2, nx - 1
  u2(ix, iy) =
    u1(ix, iy) +
    cx * (u1(ix+1,iy) + u1(ix-1,iy) - 2.*u1(ix,iy)) +
    cy * (u1(ix,iy+1) + u1(ix,iy-1) - 2.*u1(ix,iy))
end do
end do
  
```



1/1/2015

Tính toán song song

144




Giải pháp 1

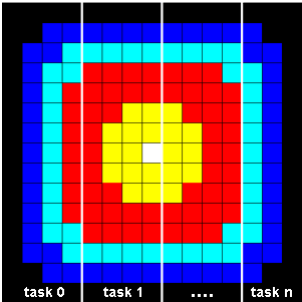


- ❑ Thực hiện theo mô hình SPMD
- ❑ Toàn bộ mảng được phân chia và phân phối như một mảng con tới tất cả các tác vụ. Mỗi tác vụ sở hữu riêng một phần của mảng toàn cục
- ❑ Xác định phụ thuộc dữ liệu
 - ❑ Các phần tử nội tại thuộc một tác vụ là độc lập với các tác vụ khác
 - ❑ Các phần tử biên là phụ thuộc vào dữ liệu của các tác vụ hàng xóm, đòi hỏi phải giao tiếp
- ❑ Tiến trình Master gửi thông tin khởi tạo cho các worker, kiểm tra độ hội tụ và thu thập dữ liệu
- ❑ Các tiến trình worker tính toán giải pháp, giao tiếp với các tiến trình hàng xóm nếu cần thiết
- ❑ Thuật giải: **màu đỏ** là những thay đổi cho xử lý song song

1/1/2015
Tính toán song song
145



Parallel Solution 1



```

find out if I am MASTER or WORKER

if I am MASTER
  initialize array
  send each WORKER starting info and subarray

  do until all WORKERS converge
    gather from all WORKERS convergence data
    broadcast to all WORKERS convergence signal
  end do

  receive results from each WORKER
else if I am WORKER
  receive from MASTER starting info and subarray

  do until solution converged
    update time
    send neighbors my border info
    receive from neighbors their border info

    update my portion of solution array

    determine if my solution has converged
    send MASTER convergence data
    receive from MASTER convergence signal
  end do

  send MASTER results
endif
  
```

1/1/2015
Tính toán song song
146



Giải pháp 2

Truyền thông và giao tiếp chồng lấn

- ❑ Trong giải pháp trước, giả định là các truyền thông chặn (blocking) được sử dụng bởi các tác vụ worker. Truyền thông chặn đợi xử lý truyền thông hoàn thành trước khi tiếp tục chỉ thị lệnh kế tiếp.
- ❑ Trong giải pháp trước, các tác vụ hàng xóm giao tiếp với dữ liệu biên, rồi mỗi tiến trình được cập nhật một phần mảng của nó.
- ❑ Số lần tính toán thường có thể được giảm bằng cách sử dụng truyền thông không chặn (non-blocking). Truyền thông không chặn cho phép công việc được thực thi khi các giao tiếp đang thực thi.
- ❑ Mỗi tác vụ có thể cập nhật dữ liệu bên trong của mảng hiện tại khi giao tiếp dữ liệu biên xảy ra, và cập nhật đường biên sau khi giao tiếp kết thúc.
- ❑ Thuật giải: **màu đỏ** là những thay đổi cho truyền thông không chặn.

1/1/2015

Tính toán song song

147



Giải pháp 2

Truyền thông và giao tiếp chồng lấn

```

find out if I am MASTER or WORKER

if I am MASTER
  initialize array
  send each WORKER starting info and subarray
  do until all WORKERS converge
    gather from all WORKERS convergence data
    broadcast to all WORKERS convergence signal
  end do
  receive results from each WORKER
else if I am WORKER
  receive from MASTER starting info and subarray
  do until solution converged
    update time

    non-blocking send neighbors my border info
    non-blocking receive neighbors border info

    update interior of my portion of solution array
    wait for non-blocking communication complete
    update border of my portion of solution array

    determine if my solution has converged
    send MASTER convergence data
    receive from MASTER convergence signal
  end do
  send MASTER results
endif

```

1/1/2015

Tính toán song song

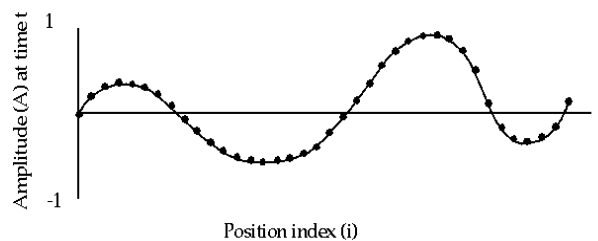
148



Phương trình sóng 1-D

1-D Wave Equation

- ❑ Trong ví dụ này, biên độ là đồng nhất, chuỗi đồ thị được tính sau một khoảng thời gian xác định.
- ❑ Việc tính toán liên quan tới:
 - ❑ Biên độ trên trục y
 - ❑ i như là chỉ số vị trí theo trục x
 - ❑ Các điểm nút đặt dọc theo đồ thị
 - ❑ Cập nhật biên độ theo thời gian không liên tục



1/1/2015

Tính toán sóng song

149



Phương trình sóng 1-D

- ❑ Biểu thức này giải quyết bài toán phương trình sóng 1-D, ở đây c là hằng số:

$$A(i, t+1) = (2.0 * A(i, t)) - A(i, t-1) + (c * (A(i-1, t) - (2.0 * A(i, t)) + A(i+1, t)))$$

- ❑ Chú ý: biên độ sẽ phụ thuộc vào bước thời gian trước đó ($t, t-1$) và các điểm hàng xóm ($i-1, i+1$). Sự phụ thuộc dữ liệu sẽ đòi hỏi một giải pháp song song sẽ liên quan đến các giao tiếp giữa các tác vụ.

1/1/2015

Tính toán sóng song

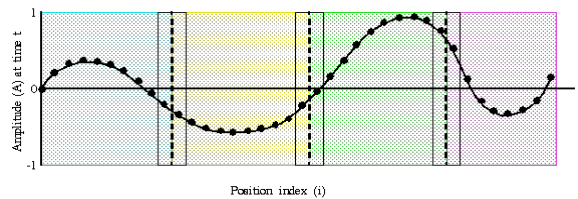
150



Phương trình sóng 1-D

Giải pháp

- ❑ Thực thi theo mô hình SPMD
- ❑ Toàn bộ mảng biên độ được phân chia và phân bố như các mảng con tới các tác vụ. Mỗi tác vụ sẽ sở hữu riêng một phần mảng toàn cục.
- ❑ Cân bằng tải: Tất cả các điểm yêu cầu số lượng công việc như nhau, như vậy số điểm nên được chia đều nhau
- ❑ Việc phân giải một khối cần phải phân chia công việc thành các tác vụ như chia khúc, cho phép mỗi tác vụ sở hữu hầu hết các điểm dữ liệu liền kề nhau.
- ❑ Các giao tiếp chỉ cần xảy ra ở các dữ liệu biên. Kích thước khối lớn hơn thì càng ít giao tiếp.



1/1/2015

Tính toán sóng song

151



Giải pháp phương trình sóng 1-D

```

find out number of tasks and task identities

#Identify left and right neighbors
left_neighbor = mytaskid - 1
right_neighbor = mytaskid + 1
if mytaskid = first then left_neighbor = last
if mytaskid = last then right_neighbor = first

find out if I am MASTER or WORKER
if I am MASTER
    initialize array
    send each WORKER starting info and subarray
else if I am WORKER
    receive starting info and subarray from MASTER
endif

#Update values for each point along string
#In this example the master participates in calculations
do t = 1, nsteps
    send left endpoint to left neighbor
    receive left endpoint from right neighbor
    send right endpoint to right neighbor
    receive right endpoint from left neighbor

#Update points along line
do i = 1, npoints
    newval(i) = (2.0 * values(i)) - oldval(i)
    + (sgtau * (values(i-1) - (2.0 * values(i)) + values(i+1)))
end do

end do

#Collect results and write to file
if I am MASTER
    receive results from each WORKER
    write results to file
else if I am WORKER
    send results to MASTER
endif
  
```

1/1/2015

Tính toán sóng song

152



Tài liệu tham khảo

- ❑ Seyed H. Roosta (2000). Parallel Processing and Parallel Algorithms Theory and Computation.
- ❑ Introduction to Parallel Computing,
https://computing.llnl.gov/tutorials/parallel_comp/
- ❑ Message Passing Interface (MPI),
<https://computing.llnl.gov/tutorials/mpi/>
- ❑ POSIX Threads Programming,
<https://computing.llnl.gov/tutorials/pthreads/>
- ❑ Các nội dung khác
<https://computing.llnl.gov/?set=training&page=index>