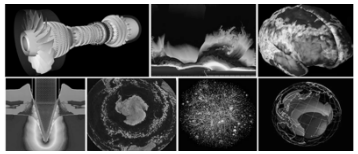
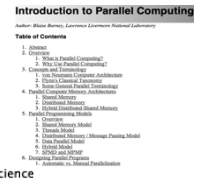
Tính toán song song và phân tán

PGS.TS. Trần Văn Lăng
tvlang@vast-hcm.ac.vn
lang@lhu.edu.vn

Tài liệu: **Introduction to Parallel Computing**

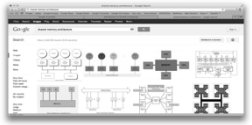
Blaise Barney, Lawrence Livermore National Laboratory
https://computing.llnl.gov/tutorials/parallel_comp/





5. Thiết kế chương trình song song

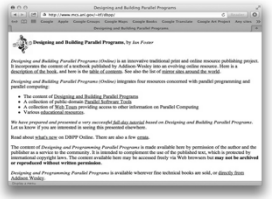
1. Song song tự động so với song song bằng tay
2. Hiểu được bài toán và chương trình
3. Phân hoạch
4. Truyền thông
5. Tích tụ
6. Ánh xạ



Tài liệu: Designing and Building Parallel Programs

Ian. Foster, Addison-Wesley, 1995, http://www.mcs.anl.gov/dbpp

6. Sự phụ thuộc dữ liệu
7. Cân bằng tải
8. Mức độ chi tiết
9. Nhập xuất
10. Chi phí của lập trình song song
11. Phân tích hiệu năng

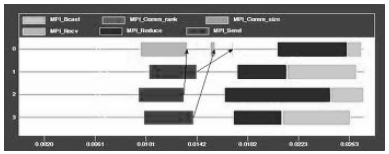


5.1 Việc song song hóa

- Thiết kế và phát triển các chương trình song song có đặc trưng đó là một quá trình rất thủ công.
- Người lập trình thường là phải chịu trách nhiệm cho cả việc nhận ra và hiện thực song song trong chương trình.

- Việc viết chương trình song song bằng tay thường mất nhiều thời gian, phức tạp hay bị lỗi
- Trong vài năm gần đây, có xu hướng phát triển công cụ hỗ trợ người lập trình chuyển đổi một chương trình tuần tự sang song song như là sự tiền xử lý

- Một trình biên dịch song song thường làm việc theo hai cách khác nhau:
  - Tự động hòa toàn
  - Theo sự hướng dẫn của người lập trình



Tự động hoàn toàn

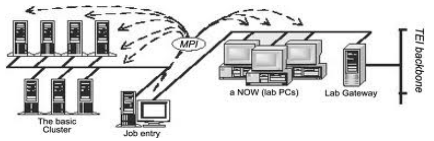
- Trình biên dịch phân tích mã nguồn và xác định cơ hội song song.
- Phân tích này bào gồm việc tìm ra các phần khó có cơ hội song song, cũng như những trọng số có thể để cải thiện hiệu năng.
- Các vòng lặp (do, for) là đối tượng xem xét thường xuyên nhất khi cần song song tự động.

Theo sự hướng dẫn

- Sử dụng trình biên dịch có hướng dẫn người lập trình chỉ một cách rõ ràng cách thức để song song một đoạn chương trình
- Cũng có thể sử dụng kết hợp với hệ thống tự động.

- Nếu chúng ta bắt đầu với một mã tuần tự với thời gian cũng như chi phí hạn chế, thì việc dùng hệ thống song song tự động là một chọn lựa.
- Tuy nhiên, cũng có một số khác biệt quan trọng khi áp dụng song song tự động:
  - Đưa ra kết quả sai
  - Hiệu năng bị gảm

  - Song song bằng thủ công linh hoạt hơn
  - Bị hạn chế trong một số đoạn lệnh (hầu hết là vòng lặp)
  - Đoạn mã có thể không cần song song.



5.2 Bài toán và chương trình song song

- Bước đầu tiên trong phát triển phần mềm song song là hiểu được bài toán muốn giải theo cách song song.
- Nếu có một chương trình tuần tự, cần phải hiểu rõ mã nguồn của nó.
- Trước khi dành thời gian để phát triển một giải pháp song song cho bài toán, hãy xác định bài toán có song song được hay không.

Bài toán song song được

- Ví dụ cần tính toán thế năng của hàng ngàn cấu trúc phân tử độc lập. Từ đó tìm thế năng cực tiểu.
  - Bài toán song song được bởi thế năng từng cấu trúc phân tử có thể tính độc lập.
  - Bài toán tìm cực tiểu cũng có thể song song được bằng cách xem xét từng khúc đã tính.

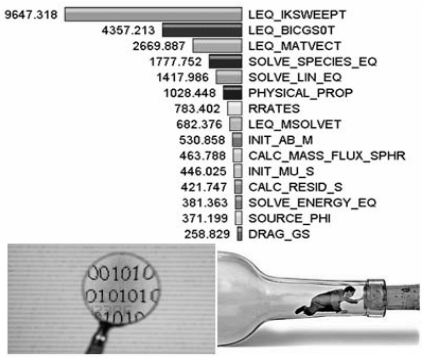
Bài toán không song song được

- Ví dụ, tính toán dãy số Finonacci (0, 1, 1, 2, 3, 5, 8, 13, 21, ...) theo công thức F(n) = F(n-1) + F(n-2)
- Đây là bài toán không song song được vì một phần tử của dãy số được tạo ra bị phụ thuộc vào các phần tử tạo ra trước đó.
  - Cụ thể, F(n) phụ thuộc vào F(n-1) và F(n-2)

Sau đó, nếu song song được

- Xác định các điểm nóng của chương trình
  - Ở đâu là công việc thực hiện
  - Trong các chương trình tính toán khoa học và kỹ thuật, công việc chính chỉ ở một vài nơi.
  - Sử dụng công cụ phân tích để hỗ trợ cho việc xác định này.
  - Bỏ qua những phần của chương trình mà ít sử dụng năng lực của CPU

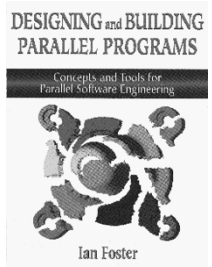
- Xác định điểm thắt cổ chai, điểm tắt nghẽ (***bottlenecks***) trong chương trình
  - Đó là những phần thực hiện với thời gian không cân đối với các phần khác. Từ đó làm phá hủy cơ hội song song. Chẳng hạn, việc nhập xuất thường diễn ra chậm.
  - Có thể cơ cấu lại chương trình, hoặc sử dụng thuật toán khác để loại bỏ vùng thực hiện chậm này.



- Xác định những yếu tố cản trở viêc song song. Đa phần sự cản trở này là do sự phụ thuộc dữ liệu (*data dependence*). Chẳng hạn, dãy số Fibonacci.
- Khảo sát các thuật toán khác nếu có thể. D9a6y là sự xem xét quan trọng nhất khi thiết kế ứng dụng song song.
- Tận dụng lợi thế của các phần mềm song song cũng như những thư viện toán học hiệu quả của các nhà cung cấp.

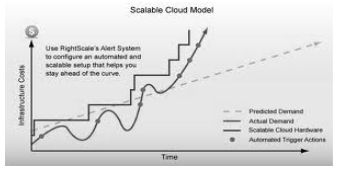
Đặc điểm của chương trình song song

- Đồng thời (concurrency)
- Tỷ lệ (scalability)
- Địa phương (locality)
- Mô đun (modularity)



- Tính đồng thời (concurrency), nhằm để có thể thực hiện trên nhiều bộ xử lý.
- Tính tỷ lệ hay co giản được, hay khả năng mở rộng (scalability-khả cỡ) thể hiện việc thuật giải có thể cài đặt mà không quan tâm đến số bộ xử lý mà chúng sẽ thi hành.

- Tính co giản thể hiện qua việc, khi số bộ xử lý tăng lên, số tiến trình trên một bộ xử lý sẽ thu ít lại, nhưng thuật giải chính nó lại không cần thay đổi.

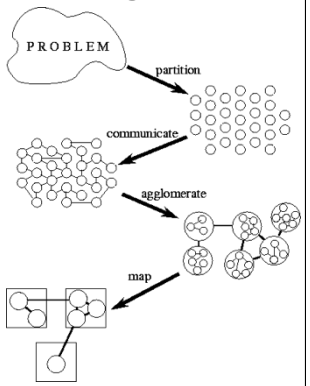


- Khi tính địa phương (locality) nhằm giảm chi phí thời gian của thuật giải.
  - Do khi đó việc truy cập đến local data (dữ liệu trên máy tại chỗ) xẩy ra thường xuyên hơn việc truy cập đến những remote data (dữ liệu ở xa)

- Tính mô đun hoá (modularity), hoàn toàn giống với sự mong đợi trong lập trình tuần tự.
  - Thực chất là sự phân hoạch những thực thể phức tạp thành các thành phần đơn giản hơn.

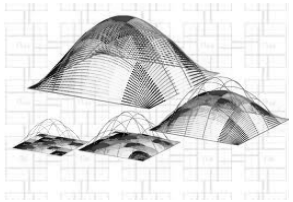
5.3 Phân chia (Partitioning)

- Có 4 giai đoạn cần thiết trong việc thiết kế một chương trình song song
  - Phân chia (Partitioning)
  - Giao tiếp (Communication)
  - Tích tụ (Agglomeration)
  - Ánh xạ (Mapping)



- Bước đầu tiên của việc thiết kế chương trình song song là tách bài toán thành ra các khúc, các phần công việc rời rạc để phân phối đến nhiều task.
- Công việc này được biết như là sự phân rã (decomposition) hoặc phân chia (partitioning).

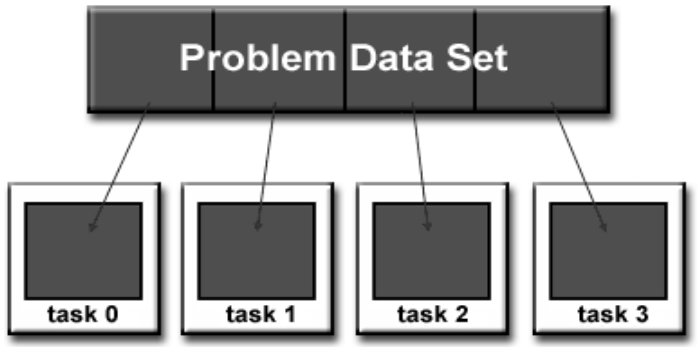
- Đây là giai đoạn tạo cơ hội song song,
- Khi phân rã, việc tính toán được thực hiện trên những vùng dữ liệu nhỏ hơn, và các hành động cũng được đưa về thành những tiến trình nhỏ hơn.



- Có 2 cách để phân chia công việc tính toán cho các task song song:
  - Phân rã theo miền (***domain decomposition***)
  - Phân rã theo chức năng (***functional decomposition***)

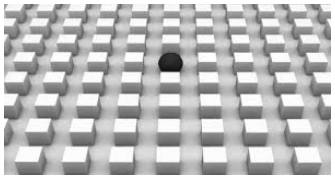
Phân rã theo miền

- Khi thiết kế chương trình, người lập trình trước hết thường quan tâm đến dữ liệu liên kết với bài toán
- Nên trong loại phân rã này, dữ liệu liên quan đến bài toán được phân rã; để rồi mỗi task song song làm việc trên một phần của dữ liệu.

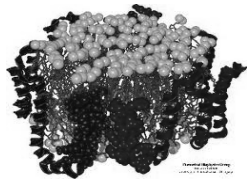


- Thông thường việc phân rã theo miền được căn cứ vào:
  - Dữ liệu đầu vào của chương trình,
  - Kết quả đầu ra được tính toán.
  - Những dữ liệu lưu giữ giá trị trung gian quá trình thực hiện.

- Phân rã theo dữ liệu là một phương pháp phân ly hiệu quả và được sử dụng nhiều nhất trong việc xác định tính đồng thời của thuật toán để có thể thao tác trên các cấu trúc dữ liệu lớn.



- Phương pháp này bao gồm 2 bước:
  - Dữ liệu trong bước tính toán sẽ được phân ra thành từng phần
  - Phần dữ liệu này sẽ được chuyển cho các task để thực thi.



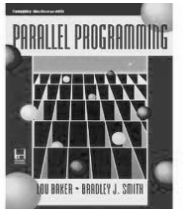
Ví dụ nhân ma trận

- Nhân 2 ma trận A và B, kết quả trả về là ma trận C. Giả sử ma trận A và B đều có kích thước n x n.
- Trước tiên phân từng ma trận thành 4 khối hay 4 ma trận con, bằng cách chia đôi các chiều của ma trận.
- Bốn ma trận con của ma trận kết quả C - mỗi phần có kích thước n/2 x n/2 - có thể được tính toán độc lập với nhau bởi 4 tiến trình.

- Bước 1: chia ma trận nhập và ma trận xuất thành 2 x 2 ma trận con.

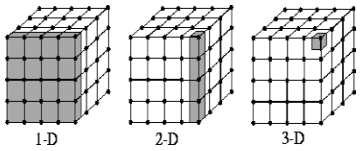
$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \times \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix} = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$$

- Bước 2: Sự phân chia việc nhân ma trận A thành 4 task dựa trên việc chia ma trận của bước 1:
  - Task 1: $C_{1,1} = A_{1,1}B_{1,1} + A_{1,2}B_{2,1}$
  - Task 2: $C_{1,2} = A_{1,1}B_{1,2} + A_{1,2}B_{22}$
  - Task 3: $C_{2,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$
  - Task 4: $C_{2,2} = A_{2,1}B_{1,2} + A_{2,2}B_{2,2}$

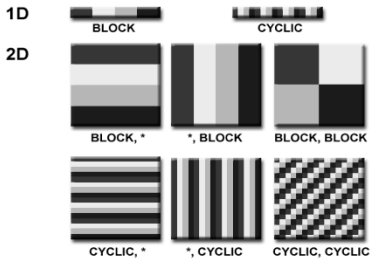


Ví du: chia miền trong lưới tính toán

- Phân rã miền cho bài toán đòi hỏi lưới 3 chiều.
- Khi đó miền phân chia là
  - 1-D, mỗi task bao gồm 5 x 5 điểm,
  - 2-D, mỗi task bao gồm 5 điểm,
  - 3-D, mỗi task chỉ có 1 điểm nút.

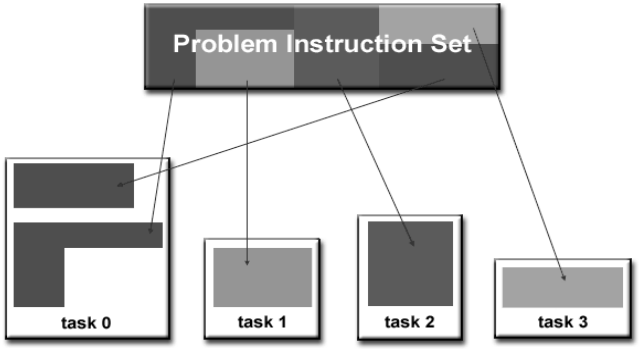


- Trường hợp 3-D được coi là mềm dẻo nhất và dễ dàng chấp nhận trong giai đoạn thiết kế



Phân rã theo chức năng

- Sự phân rã tốt ở đây không chỉ dừng lại ở việc phân rã dữ liệu tính toán, mà còn cả việc phân chia các thao tác tác động tới dữ liệu.
- Theo cách tiếp cận này, trọng tâm là những tính toán, không phải là dữ liệu.



Mô hình hệ sinh thái

- Mỗi chương trình tính toán quần thể của một nhóm nào đó, mà ở đó sự tăng trưởng của mỗi nhóm phụ thuộc vào nhóm lân cận.
- Như một sự tiến triển, mỗi tiến trình tính toán trạng thái hiện tại, rồi trao đổi thông tin với quần thể bên cạnh.
- Tất cả các task iến triển lên để tính toán trạng thái ở bước thời gian tiếp theo.

- Xét các nhóm: thực vật, động vật ăn cỏ, động vật ăn thịt, động vật ăn xác chết, sinh vật phân hủy