

## Tính toán song song và phân tán

PGS.TS. Trần Văn Lăng

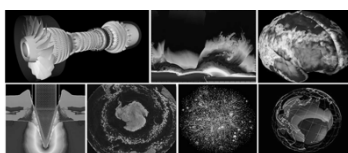
tvlang@vast-hcm.ac.vn

lang@lhu.edu.vn

**Tài liệu: Introduction to Parallel Computing**

Blaise Barney, Lawrence Livermore National Laboratory

[https://computing.llnl.gov/tutorials/parallel\\_comp/](https://computing.llnl.gov/tutorials/parallel_comp/)



Introduction to Parallel Computing

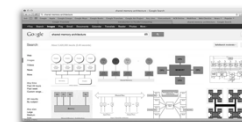
Table of Contents	
1.	Abstract
2.	Chapter 1
3.	Chapter 2
4.	Chapter 3
5.	Chapter 4
6.	Chapter 5
7.	Chapter 6
8.	Chapter 7
9.	Chapter 8
10.	Chapter 9
11.	Chapter 10
12.	Chapter 11
13.	Chapter 12
14.	Chapter 13
15.	Chapter 14
16.	Chapter 15
17.	Chapter 16
18.	Chapter 17
19.	Chapter 18
20.	Chapter 19
21.	Chapter 20
22.	Chapter 21
23.	Chapter 22
24.	Chapter 23
25.	Chapter 24
26.	Chapter 25
27.	Chapter 26
28.	Chapter 27
29.	Chapter 28
30.	Chapter 29
31.	Chapter 30
32.	Chapter 31
33.	Chapter 32
34.	Chapter 33
35.	Chapter 34
36.	Chapter 35
37.	Chapter 36
38.	Chapter 37
39.	Chapter 38
40.	Chapter 39
41.	Chapter 40
42.	Chapter 41
43.	Chapter 42
44.	Chapter 43
45.	Chapter 44
46.	Chapter 45
47.	Chapter 46
48.	Chapter 47
49.	Chapter 48
50.	Chapter 49
51.	Chapter 50
52.	Chapter 51
53.	Chapter 52
54.	Chapter 53
55.	Chapter 54
56.	Chapter 55
57.	Chapter 56
58.	Chapter 57
59.	Chapter 58
60.	Chapter 59
61.	Chapter 60
62.	Chapter 61
63.	Chapter 62
64.	Chapter 63
65.	Chapter 64
66.	Chapter 65
67.	Chapter 66
68.	Chapter 67
69.	Chapter 68
70.	Chapter 69
71.	Chapter 70
72.	Chapter 71
73.	Chapter 72
74.	Chapter 73
75.	Chapter 74
76.	Chapter 75
77.	Chapter 76
78.	Chapter 77
79.	Chapter 78
80.	Chapter 79
81.	Chapter 80
82.	Chapter 81
83.	Chapter 82
84.	Chapter 83
85.	Chapter 84
86.	Chapter 85
87.	Chapter 86
88.	Chapter 87
89.	Chapter 88
90.	Chapter 89
91.	Chapter 90
92.	Chapter 91
93.	Chapter 92
94.	Chapter 93
95.	Chapter 94
96.	Chapter 95
97.	Chapter 96
98.	Chapter 97
99.	Chapter 98
100.	Chapter 99
101.	Chapter 100

Dr. Tran Van Lang, Assoc. Prof. in Computer Science

1

## Nội dung

1. Tổng quan
2. Khái niệm và thuật ngữ
3. Kiến trúc bộ nhớ của máy tính song song
4. Mô hình lập trình song song
5. Thiết kế chương trình song song
6. Ứng dụng

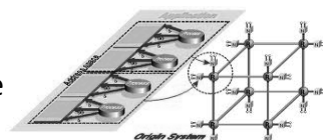


Dr. Tran Van Lang, Assoc. Prof. in Computer Science

2

## 4. Mô hình lập trình song song

1. Tổng quan
2. Mô hình bộ nhớ chia sẻ
3. Mô hình Thread
4. Mô hình phân tán/chuyển thông điệp
5. Mô hình song song dữ liệu
6. Mô hình lai
7. SPMD và MPMD



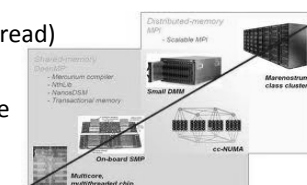
Dr. Tran Van Lang, Assoc. Prof. in Computer Science

3

## 4.1 Tổng quan

- Có một số mô hình lập trình song song (Parallel Programming Model) được sử dụng phổ biến:

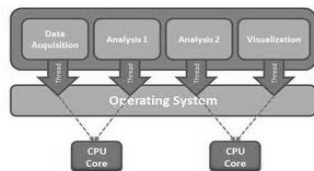
- Shared Memory (không có thread)
- Threads (luồng)
- Distributed Memory/Message Passing



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

4

- Data Parallel
- Hybrid
- Single Program Multiple Data (SPMD)
- Multiple Program Multiple Data (MPMD)



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

5

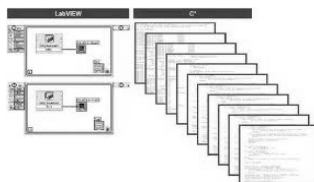
- Các mô hình lập trình song song tồn tại như là một sự trừu tượng hóa trên phần cứng và kiến trúc bộ nhớ.
- Những mô hình này không thể ch ra một loại máy tính hay kiến trúc bộ nhớ cụ thể khi cần hiện thực



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

6

- Trong thực tế, bất kỳ mô hình nào trong số những mô hình này cũng có thể hiện thực trên bất kỳ phần cứng nằm bên dưới.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

7

## Ví dụ



- Mô hình Shared memory model trên máy tính Distributed memory: sử dụng hệ thống shared memory KSR (Kendall Square Research)
  - Bộ nhớ máy tính được phân phối một cách vật lý thông qua các máy được nối mạng.
  - Nhưng người dùng nhìn thấy như là một bộ nhớ chia sẻ đơn lẻ (global address space). Vì vậy có thể tiếp cận như là "virtual shared memory" với KSR.

Dr. Tran Van Lang, Assoc. Prof. in Computer Science

8



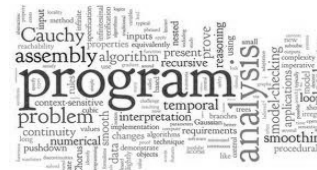
- Mô hình Distributed memory trên máy Shared memory: dùng hệ thống distributed memory Message Passing Interface (MPI)
  - SGI Origin 2000 sử dụng loại CC-NUMA của kiến trúc bộ nhớ chia sẻ, mà ở đó mỗi task truy cập một cách trực tiếp đến không gian địa chỉ chung (global address space) lan rộng trên tất cả các máy.
  - Tuy nhiên, có thể sử dụng MPI để gửi và nhận thông điệp như một mạng các máy tính distributed memory.

Dr. Tran Van Lang, Assoc. Prof. in Computer Science

9

#### • Sử dụng mô hình nào:

- Thường là một sự tổ hợp của những lựa chọn mang tính cá nhân
- Không có mô hình tốt nhất, mặc dù có những mô hình khi hiện thực chắc chắn tốt hơn mô hình khác.

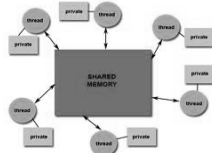


Dr. Tran Van Lang, Assoc. Prof. in Computer Science

10

## 4.2 Mô hình Shared Memory (không có threads)

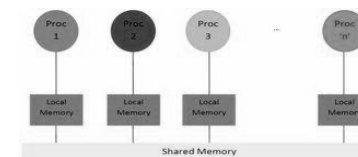
- Trong mô hình lập trình này các task chia sẻ không gian địa chỉ chung mà các task này đọc ghi bất đồng bộ.
- Các cơ chế khác nhau như locks/semaphores có thể sử dụng để kiểm soát việc truy cập đến shared memory.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

11

- Lợi thế của mô hình này là người lập trình không cần chỉ định việc truyền dữ liệu giữa các task; chương trình được phát triển thường được đơn giản hóa.

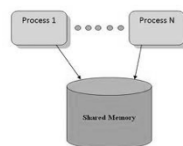


Dr. Tran Van Lang, Assoc. Prof. in Computer Science

12

- Một nhược điểm quan trọng:

- Khó để giữ lại được tính nguyên thủy của dữ liệu khi mà nhiều bộ xử lý dùng cùng dữ liệu này
- Việc kiểm soát dữ liệu một cách địa phương là khó khăn đối với người lập trình trình độ trung bình



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

13

## Việc hiện thực

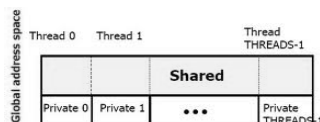
- Sử dụng các trình biên dịch có sẵn của hệ thống. Chẳng hạn, trên máy SMP độc lập, đây là vấn đề không phức tạp.
- Dùng trên máy distributed shared memory, chẳng hạn như SGI Origin, bộ nhớ được phân phối một cách vật lý xuyên qua mạng các máy, nhưng phần cứng và phần mềm được đặc tả toàn cục.

Dr. Tran Van Lang, Assoc. Prof. in Computer Science

14

## 4.3 Mô hình Thread

- Là mô hình thuộc loại lập trình với shared memory.
- Trong mô hình thread của chương trình song song, một tiến trình có thể có nhiều con đường, thực thi đồng thời.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

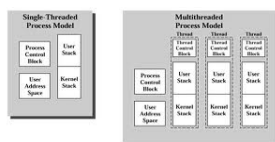
15

- Đơn giản nhất để mô tả thread đó khái niệm một chương trình bao gồm nhiều chương trình con:
  - Chương trình chính (vd: a.out) được lập lịch để chạy trên máy với những yêu cầu cần thiết
  - Chương trình chính thực hiện nối tiếp một vài công việc và lập lịch để cho các task (các thread) thực thi một cách đồng thời.

Dr. Tran Van Lang, Assoc. Prof. in Computer Science

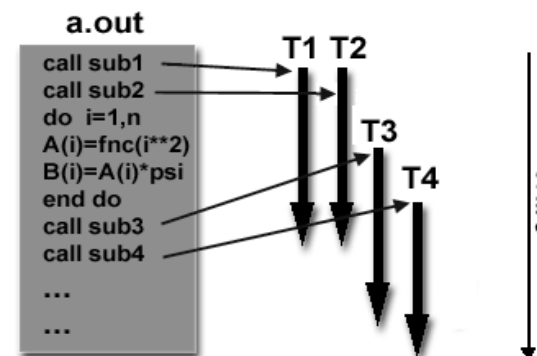
16

- Mỗi thread có thể có dữ liệu địa phương, nhưng cũng có thể chia sẻ toàn bộ dữ liệu của chương trình chính.
  - Điều này giúp tiết kiệm chi phí liên quan đến việc sử dụng lại tài nguyên cho mỗi thread.
  - Mỗi thread có lợi ích là vẫn chia sẻ không gian bộ nhớ với chương trình chính.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

17



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

18

- Rõ ràng nhất đó là mô tả công việc của mỗi thread như là một chương trình con trong chương trình chính.

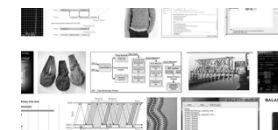
- Mọi thread có thể thực hiện bất kỳ chương trình con nào tại cùng thời điểm với các thread khác.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

19

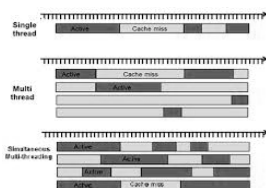
- Các thread giao tiếp với các thread khác thông qua bộ nhớ chung.
- Điều này đòi hỏi phải xây dựng sự đồng bộ để đảm bảo rằng nhiều hơn một thread không cập nhật cùng một địa chỉ chung vào bất cứ lúc nào.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

20

- Thread có thể đến và đi, nhưng chương trình chính vẫn hiện diện để cung cấp các tài nguyên chia sẻ cần thiết cho đến khi ứng dụng hoàn tất.

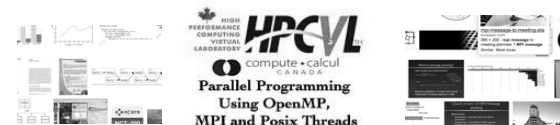


Dr. Tran Van Lang, Assoc. Prof. in Computer Science

21

## Việc hiện thực

- Để lập trình, thread được hiện thực theo một trong hai cách:
  - Sử dụng hàm thư viện được gọi là từ bên trong một chương trình.
  - Sử dụng các chỉ thị trực tiếp trong chương trình



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

22

- Trong cả hai trường hợp, người lập trình phải thể hiện sự song song trong chương trình.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

23

- Việc hiện thực thread không phải là một cách tiếp cận mới. Bởi các nhà sản xuất phần cứng đã tích hợp các phiên bản thread độc quyền riêng của họ.
  - Những phiên bản này khác hoàn toàn với phiên bản của các nhà sản xuất khác.
  - Điều này làm khó khăn cho người lập trình khi viết những ứng dụng thread mang tính cơ động cao (không phụ thuộc phần cứng)

Dr. Tran Van Lang, Assoc. Prof. in Computer Science

24

- Việc nỗ lực để chuẩn hóa các thread không đạt kết quả.
- Ngày nay có 2 phiên bản hiện thực thread hoàn toàn khác nhau:
  - POSIX Threads
  - OpenMP

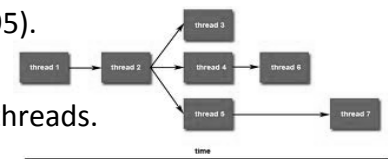


Dr. Tran Van Lang, Assoc. Prof. in Computer Science

25

## POSIX Thread

- Dùng các hàm thư viện; đòi hỏi phải song song hóa
- Được đặc tả bởi IEEE POSIX 1003.1c standard (1995).
- Chỉ dùng ngôn ngữ C.
- Thường được gọi là Pthreads.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

26

- Hầu hết các nhà sản xuất phần cứng hiện nay cung cấp Pthreads riêng của họ.
- Việc song song hóa rất rõ ràng, nên đòi hỏi người lập trình phải đặc biệt chú ý đến từng chi tiết khi viết chương trình.
- POSIX Threads: [computing.llnl.gov/tutorials/pthreads](http://computing.llnl.gov/tutorials/pthreads)

Dr. Tran Van Lang, Assoc. Prof. in Computer Science

27

## OpenMP

- Sử dụng các chỉ thị trực tiếp trong chương trình; có thể viết theo kiểu tuần tự.
- Được xây dựng bởi một nhóm các nhà sản xuất máy tính lớn
- API của OpenMP FORTRAN được phát hành vào 28/10/1997; của C/C++ vào cuối năm 1998.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

28

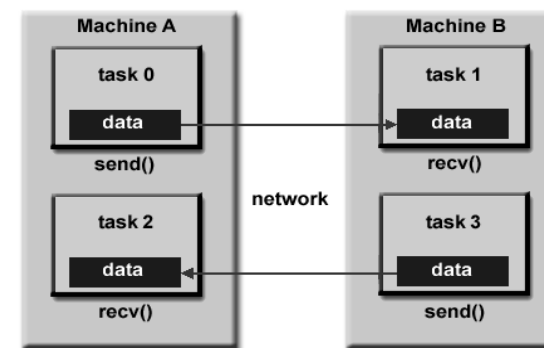
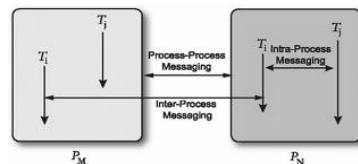
- Thỏa tính portable/multi-platform, bao gồm platform Unix và Windows NT
- Chương trình có thể viết bằng C/C++ và FORTRAN
- Dễ dàng sử dụng
- OpenMP: [computing.llnl.gov/tutorials/openMP](http://computing.llnl.gov/tutorials/openMP)

- Hãng Microsoft cũng có phiên bản Thread của riêng.
- Không liên quan gì đến UNIX POSIX chuẩn hay OpenMP.



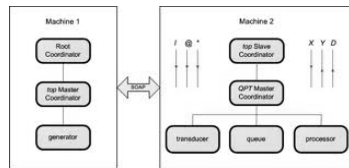
#### 4.4 Mô hình phân tán/chuyển thông điệp

- Mô hình Distributed Memory/Message Passing có những đặc tính như:
  - Tập hợp các task sử dụng bộ nhớ địa phương trong suốt quá trình tính toán. Nhiều task có thể nằm trên một máy vật lý và/hoặc trên một số bất kỳ các máy.





- Task trao đổi dữ liệu thông qua việc truyền thông bởi gửi và nhận thông điệp.
- Việc truyền dữ liệu thường đòi hỏi hoạt động hợp tác được thực hiện bởi mỗi tiến trình.
  - Chẳng hạn, thao tác gửi phải có một thao tác nhận phù hợp.

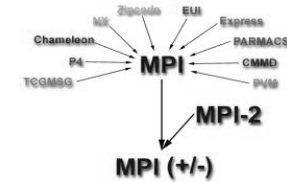


Dr. Tran Van Lang, Assoc. Prof. in Computer Science

33

## Việc hiện thực

- Sử dụng các hàm thư viện có sẵn để hiện thực thao tác chuyển thông điệp.
- Các hàm này được gọi trong chương trình, người lập trình chịu trách nhiệm xác định sự song song của thuật giải.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

34

- Từ những năm 1980 đã có một loạt các thư viện chuyển thông điệp khác nhau đánh kể.
- Điều này gây khó khăn cho người lập trình khi muốn phát triển những ứng dụng linh động trên các môi trường khác nhau.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

35

- Năm 1992, MPI Forum được thành lập với mục đích chính là thiết lập giao diện chuẩn cho việc hiện thực chuyển thông điệp.
- Phần 1 của **Message Passing Interface (MPI)** ra đời vào 1994. Phần 2 (MPI-2) vào 1996.
- Những đặc tả MPI có tại:  
<http://www-unix.mcs.anl.gov/mpi/>

Dr. Tran Van Lang, Assoc. Prof. in Computer Science

36

- MPI hiện nay là một chuẩn công nghiệp không chính thức (is now the "de facto" industry standard) trong việc chuyển thông điệp,
- Thay thế hầu như tất cả các hiện thực chuyển thông điệp khác để tạo ra ứng dụng.

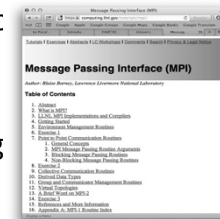


**CSWMPI**  
High-Performance Message  
Passing Middleware

Dr. Tran Van Lang, Assoc. Prof. in Computer Science

37

- Hiện thực MPI tồn tại hầu như trong tất cả các platform tính toán song song thông dụng.
- Cũng lưu ý rằng, tất cả các chương trình đều bao hàm mọi thứ có trong MPI1 và MPI2.
- MPI: [computing.llnl.gov/tutorials/m](http://computing.llnl.gov/tutorials/m)



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

38

## 4.5 Mô hình Data Parallel

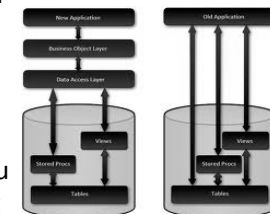
- Data Parallel Model (Mô hình song song dữ liệu) có những đặc điểm như sau:
  - Hầu hết các công việc song song tập trung vào việc thực hiện các thao tác trên một tập hợp dữ liệu. Tập hợp dữ liệu này thường được tổ chức thành một cấu trúc chung như một mảng hay một khối lập phương.



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

39

- Tập hợp các task làm việc cùng với nhau trên cùng cấu trúc dữ liệu. Tuy nhiên, mỗi task làm việc trên một phần khác nhau của cùng cấu trúc dữ liệu.
- Task thực hiện thao tác giống nhau trên phần công việc của nó. Chẳng hạn, "thêm 4 vào mỗi phần tử của mảng".



Dr. Tran Van Lang, Assoc. Prof. in Computer Science

40