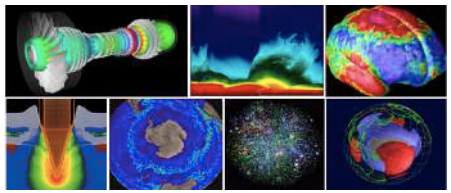
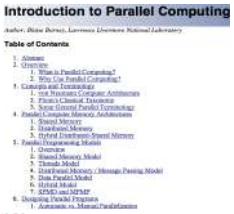
Tính toán song song và phân tán

PGS.TS. Trần Văn Lăng
langtv@vast.vn

Tài liệu: Introduction to Parallel Computing

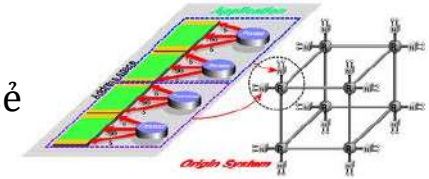
Blaise Barney, Lawrence Livermore National Laboratory
https://computing.llnl.gov/tutorials/parallel_comp/





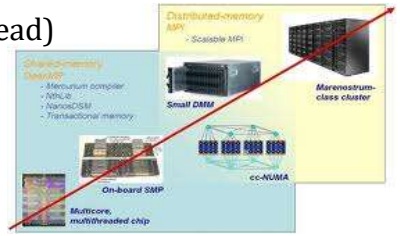
4. Mô hình lập trình song song

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

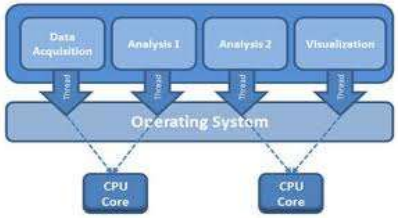


4.1 Tổng quan

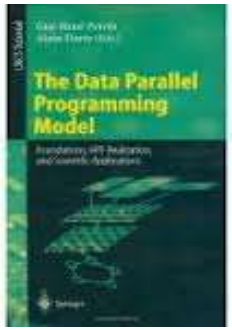
- Một số mô hình lập trình song song (Parallel Programming Model) được sử dụng phổ biến:
  - Shared Memory (không có thread)
  - Threads (luồng)
  - Distributed Memory/Message Passing



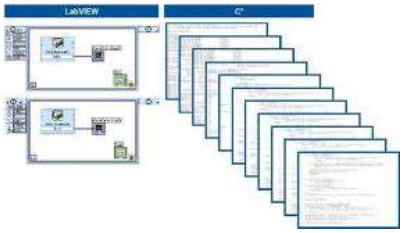
- Data Parallel
- Hybrid
- Single Program Multiple Data (SPMD)
- Multiple Program Multiple Data (MPMD)



- 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 chỉ ra một loại máy tính hay kiến trúc bộ nhớ cụ thể khi cần hiện thực



- 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.



Ví dụ

- Mô hình Shared memory 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 sẽ đơn lẻ (global address space). Vì vậy có thể tiếp cận như là "virtual shared memory" với KSR.

Kendall Square Research

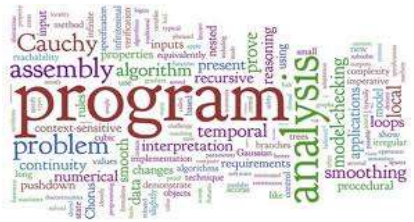
- Là một công ty siêu máy tính đặt ở Quảng trường Kendall gần MIT, được thành lập từ 1986.
- Hệ thống KSR chạy trên một phiên bản đặt biệt của hệ điều hành OSF/2 (một biến thể của Unix) với chương trình được biên dịch bởi trình biên dịch Greehills C và FORTRAN.
- Bên cạnh các ứng dụng khoa học truyền thống, KDR còn liên kết với Oracle trong việc cung cấp ứng dụng thương mại được song song hóa với Oracle PRDBMS.

- Kiến trúc KSR như là Cache Only Memory Architecture (COMA)
- COMA là tổ chức bộ nhớ máy tính để dùng như là một multiprocessor (Distributed Memory) mà trong đó mỗi local memory ở mỗi node được dùng như bộ nhớ đệm (cache) – điều này tương phản với tổ chức NUMA, trong đó local memory dùng như là bộ nhớ chính thực sự.

- 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 cá máy.
  - Tuy nhiên, có thể sử dụng MPI để gửi và nhận thông điệp như một mạng các máy tính distributed memory.

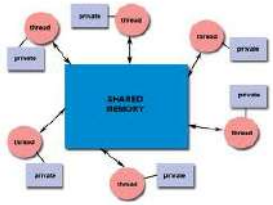
Sử dụng mô hình nào ???

- 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.

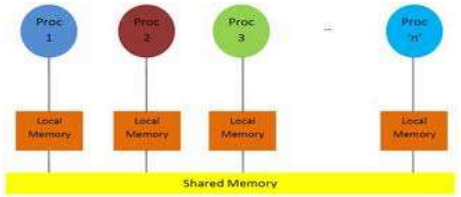


4.2 Mô hình Shared Memory (không có threads)

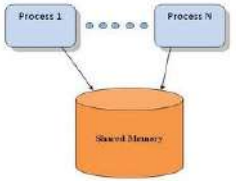
- Trong mô hình lập trình này các task chia sẻ không gian địa chỉ chung mà các task này đọc ghi bất đồng bộ.
- Các cơ chế khác nhau như locks/semaphores có thể sử dụng để kiểm soát việc truy cập đến shared memory.



- 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.



- 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

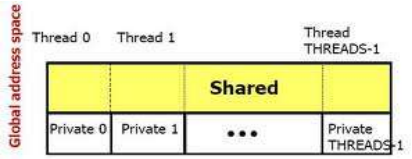


Việc hiện thực

- 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.

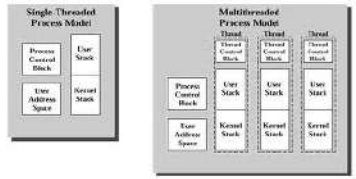
4.3 Mô hình Thread

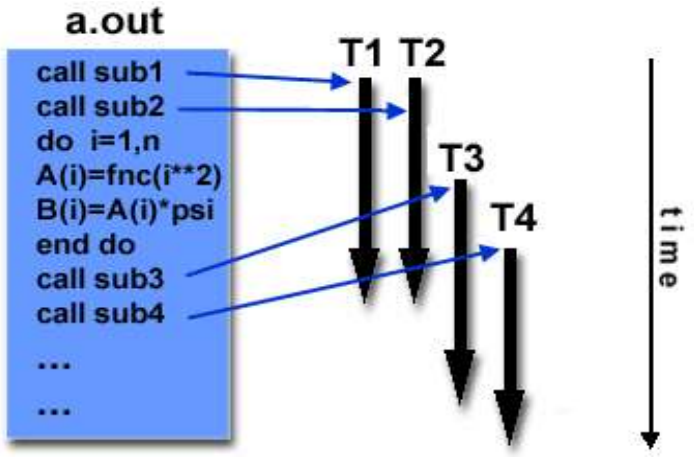
- Là mô hình thuộc loại lập trình với shared memory.
- Trong mô hình thread của chương trình song song; một chương trình có nhiều tiến trình có thể có nhiều cách để thực thi đồng thời.



- 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 đòi hỏi cần thiết
  - Ngoài ra, 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.

  - 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 sẽ không gian bộ nhớ với chương trình chính.

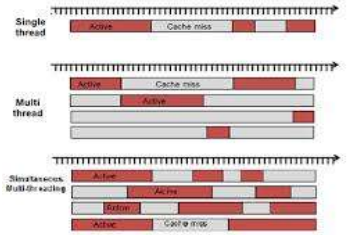




- Rõ ràng nhất đó là mô tả công việc của mỗi thread như là một chương trình con trong chương trình chính.
  - Mọi thread có thể thực hiện bất kỳ chương trình con nào tại cùng thời điểm với các thread khác.

- 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.

- 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.



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

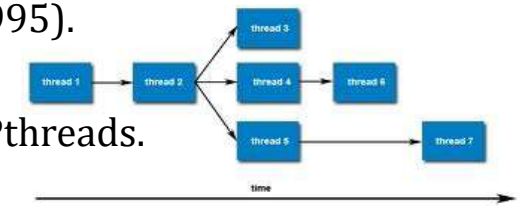
- Trong cả hai trường hợp, người lập trình phải thể hiện sự song song trong chương trình.

- 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)

- 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

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.



- 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

OpenMP

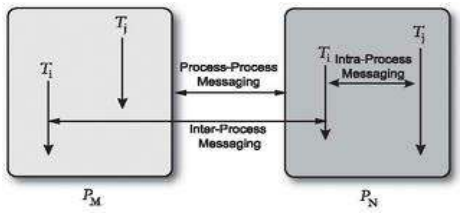
- Sử dụng các chỉ thị trực tiếp trong chương trình; có thể viết theo kiểu tuần tự.
- Được xây dựng bởi một nhóm các nhà sản xuất máy tính lớn
- API của OpenMP FORTRAN được phát hành vào 28/10/1997; của C/C++ vào cuối năm 1998.

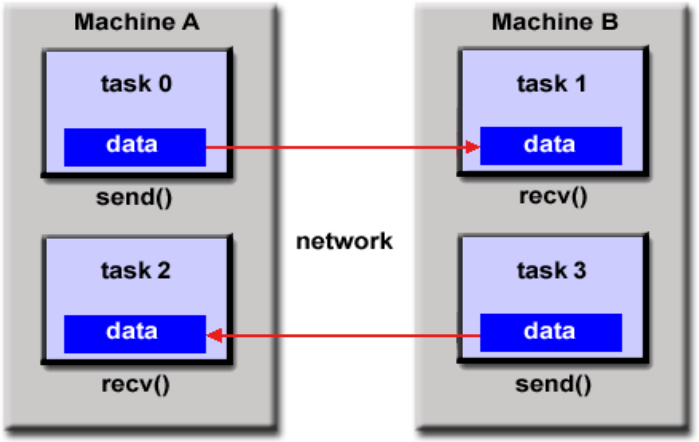
- Thỏa tính portable/multi-platform, bao gồm platform Unix và Windows NT
- Chương trình có thể viết bằng C/C++ và FORTRAN
- Dễ dàng sử dụng
- OpenMP: computing.llnl.gov/tutorials/openMP

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

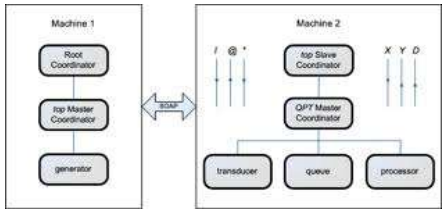
4.4 Mô hình phân tán/chuyển thông điệp

- 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.



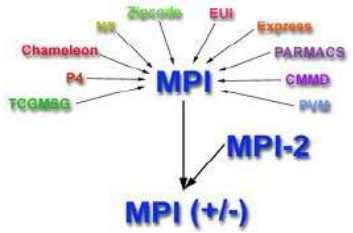


– Task trao đổi dữ liệu thông qua việc truyền thông bởi gửi và nhận thông điệp.
– Việc truyền dữ liệu thường đòi hỏi hoạt động hợp tác được thực hiện bởi mỗi tiến trình.
  • Chẳng hạn, thao tác gửi phải có một thao tác nhận phù hợp.



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.



- Từ những năm 1980 đã có một loạt các thư viện chuyển thông điệp khác nhau đáng 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.

- 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/

- 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.

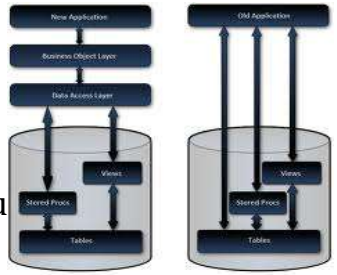
CS WMPI
High-Performance Message Passing Middleware

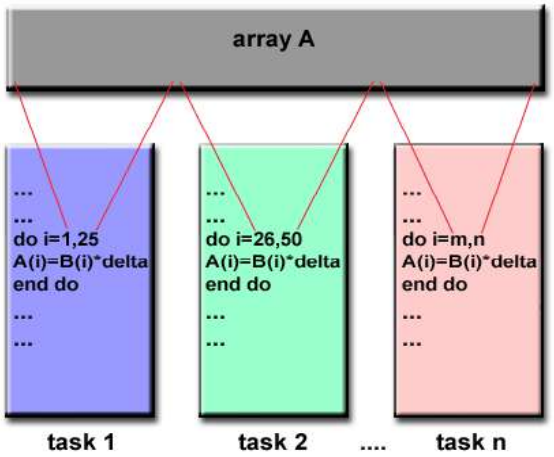
- Hiện thực MPI tồn tại hầu như trong tất cả các platform tính toán song song thông dụng.
- Cũng lưu ý rằng, tất các các chương trình đều bao hàm mọi thứ có trong cả MPI1 và MPI2.
- MPI: computing.llnl.gov/tutorials/mpi

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 mảnh hay một khối lập phương.

– 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".





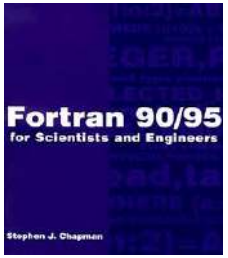
- Trên kiến trúc shared memory, tất cả các task có thể có quyền truy cập đến cấu trúc dữ liệu thông qua bộ nhớ toàn cục.
- Trên kiến trúc distributed memory, cấu trúc dữ liệu được phân ra và thường trú như những khúc, những khoang (chunk) trong bộ nhớ địa phương của mỗi task.

Việc hiện thực

- Lập trình với mô hình song song dữ liệu thường được thực hiện bằng cách viết một chương trình với một cấu trúc song song dữ liệu (data parallel construct).
- Những cấu trúc này có thể gọi đến thư viện chương trình con song song hoặc những chỉ thị biên dịch được chấp nhận bởi trình biên dịch song song.

FORTRAN 90, 95 (F90, F95)

- Chuẩn ISO/ANSI mở rộng từ FORTRAN 77
  - Chứa mọi thứ có trong FORTRAN 77
  - Bổ sung thêm tập hợp ký tự như là định dạng mới.
  - Bổ sung vào cấu trúc chương trình và thêm những lệnh mới.
  - Thêm biến, hành vi và đối số



  - Thêm con trỏ và cấp phát bộ nhớ động.
  - Thêm việc xử lý mảng (mảng được xử lý như đối tượng).
  - Thêm chức năng đệ quy và các hàm cấp thấp.
  - Và có nhiều chức năng mới so với FORTRAN 77
- Việc hiện thực có hiệu lực trên hầu hết các platform song song phổ biến.

High Performance Fortran (HPF)

- Mở rộng từ FORTRAN 90 để hỗ trợ lập trình song song dữ liệu.
- Có mọi thứ của FORTRAN 90
- Các chỉ thị để trình biên dịch biết các phân phối dữ liệu được thêm vào.

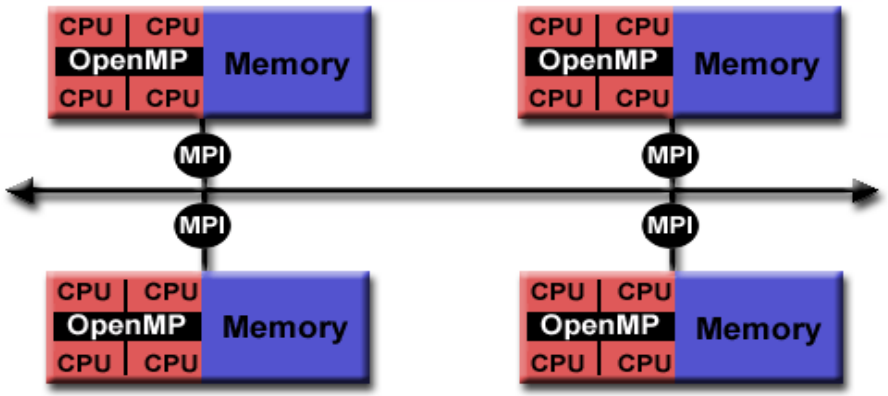
- Nâng cao việc tối ưu khi sinh ra mã máy.
- Cấu trúc song song dữ liệu được thêm vào.
- Trình biên dịch HPF compilers tương đối phổ thông vào những năm 90, nhưng bây giờ không còn được dùng thông dụng.

Sử dụng cho Distributed Memory

- Có thể dùng thư viện MPI trong việc phân phối dữ liệu.
- Việc chuyển thông điệp được ẩn phía sau.

4.6 Hybrid Model

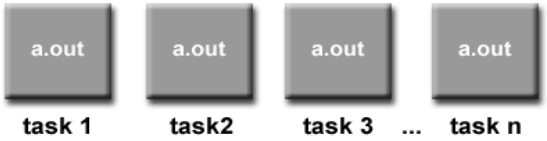
- Hybrid model (mô hình lai) là sự kết hợp các mô hình mô tả trước đây.
- Chẳng hạn, hiện nay thường có sự kết hợp giữa mô hình chuyển thông điệp (MPI) với mô hình threads (OpenMP).
  - Thread thực hiện các tính toán dùng dữ liệu địa phương của node tính toán.
  - Khi truyền dữ liệu giữa các tiến trình trên những node khác nhau dùng MPI.



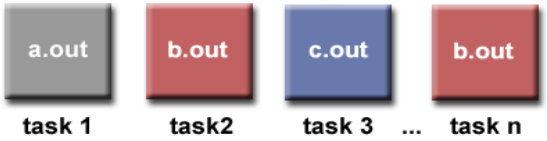
- Ví dụ tương tự và ngày càng phổ biến khác của mô hình lai là dùng lập trình MPI với GPU (Graphics Processing Unit).
  - GPU thực hiện các tính toán mạnh dùng dữ liệu địa phương trên node.
  - Truyền thông giữa các tiến trình trên các node khác nhau dùng MPI.

4.7 Mô hình SPMD và MPMD

- SPMD (Single Program Multiple Data)

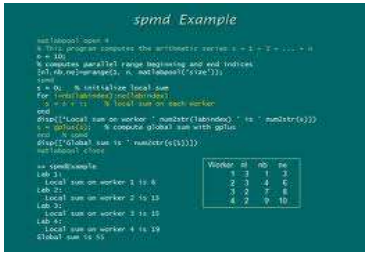


- MPMD (Multiple Program Multiple Data)

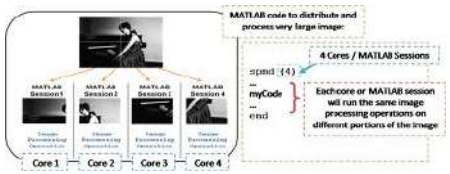


Single Program Multiple Data (SPMD)

- SPMD thực sự là một mô hình lập trình cấp cao, có thể được xây dựng dựa trên tổ hợp tùy ý của các mô hình lập trình đã đề cập trước đó.



- Single Program: tất cả các task thi hành bản sao của cùng chương trình một cách đồng thời.
  - Chương trình này có thể là thread, message passing, data parallel hay hybrid.



- Multiple Data: các task có thể sử dụng dữ liệu khác nhau.
- Chương trình SPMD thường có nhiều phần khác nhau, căn cứ và một số điều kiện để có những phần thực hiện trên các task tương ứng.
  - Như vậy, các task không nhất thiết phải thi hành toàn bộ chương trình, mà chỉ cần một phần của nó.

- Mô hình SPMD dùng lập trình chuyển thông điệp hoặc lai ghép.
- Đây là mô hình lập trình song song phổ biến nhất cho các cluster với nhiều node.

Multiple Program Multiple Data (MPMD)

- Cũng giống như SPMD, MPMD thực sự là mô hình lập trình cấp cao.
- Multiple Program: các task có thể thi hành những chương trình khác nhau một cách đồng thời.
  - Các chương trình này có thể là threads, message passing, data parallel, hoặc hybrid.

Multiple Program Multiple Data (MPMD)

- Multiple data: tất cả các task có thể dùng dữ liệu khác nhau.
- Ứng dụng MPMD không phổ biến như ứng dụng SPMD, nhưng có thể tốt hơn trong một số loại bài toán.
- Đặc biệt đối với loại bài toán được phân ly thành các task theo chức năng (functional decomposition).