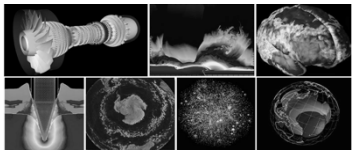
Tính toán song song và phân tán

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



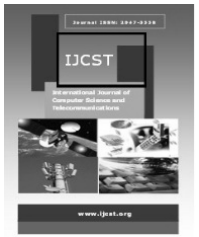
8. PVM trong việc lập trình song song

1. Môi trường truyền thông điệp
2. Hệ thống truyền thông điệp
3. PVM – Parallel Virtual Machine
4. Kiến trúc PVM
5. Cài đặt PVM
6. Sử dụng PVM
7. Lập trình trong PVM

8.1 Môi trường truyền thông điệp

- Để thực hiện tính toán song song và phân tán, cần có môi trường truyền thông điệp với 3 yếu tố:
  - Multiple processors (Cho các trạm làm việc)
  - Network (Liên kết giữa các trạm)
  - Môi trường tạo và quản lý việc xử lý song song
    - Hệ điều hành
    - Môi trường giao tiếp (PVM, MPI, ...)
    - Thư viện truyền thông điệp

- Để viết chương trình song song:
  - Phân ly thuật giải hoặc dữ liệu thành các phần riêng.
  - Phân bổ những phần công việc này như các task làm việc đồng thời trên các bộ xử lý.
  - Hợp tác và trao đổi giữa các bộ xử lý.



- Để hiện thực một chương trình song song, có thể sử dụng, hoặc:
  - Một ngôn ngữ song song chuyên biệt
  - Ngôn ngữ cấp cao với các cú pháp và từ khóa liên quan đến song song.
  - Ngôn ngữ cấp cao thông dụng với các hàm thư viện liên quan đến song song.

- Theo ba cách tiếp cận trên:
  - *occam* là ngôn ngữ lập trình song song chuyên biệt, dùng trên máy gọi là *transputer*
  - Một vài ngôn ngữ xử lý song song cấp cao như **CC++** (Compositional **C++**); FM (FORTRAN M), FORTRAN 90, HPF, HPC, ...

- Sử dụng những hàm thư viện về truyền thông điệp (chẳng hạn PVM và MPI) với ngôn ngữ C/C++, FORTRAN.

8.2 Hệ thống truyền thông điệp

- Hệ thống truyền thông điệp tạo ra môi trường cho phép người lập trình cài đặt chương trình tính toán song song.
- Môi trường cài đặt này có thể hoạt động trên nhiều chủng loại máy tính khác nhau (máy PC với bộ xử lý thuộc họ Intel, các kiến trúc Sparc, Alpha, HP, ...)

- Hầu hết các ứng dụng song song đều được cài đặt trên hệ điều hành UNIX như Solaris, AIX, Linux, ...
- Chính vì vậy, các máy với những hệ điều hành này đều có thể tạo ra hệ thống truyền thông điệp.
- Có hai hệ thống chuyển thông điệp phổ biến:
  - Hệ thống PVM (*Parallel Virtual Machine*)
  - Môi trường MPI *(Message-Passing Interface)*

8.3 Parallel Virtual Machine

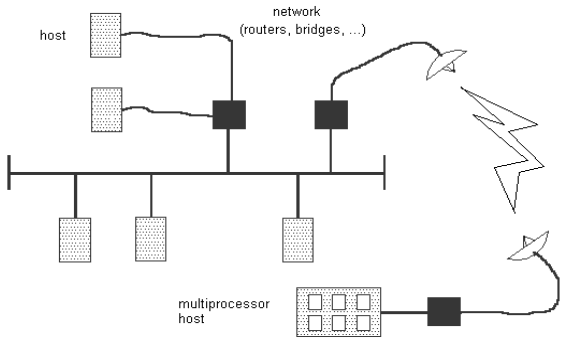
- PVM – Parallel Virtual Machine (*máy ảo song song*) được dùng để chỉ một máy tính logic có bộ nhớ phân tán
- PVM cung cấp các thủ tục để khởi tạo các task trên máy ảo (*virtual machine*) và cho phép các task này trao đổi với nhau.
- *Task* trên hệ thống PVM được coi là một đơn vị tính toán, có ý nghĩa như một *UNIX process*.

8.4 Kiến trúc PVM

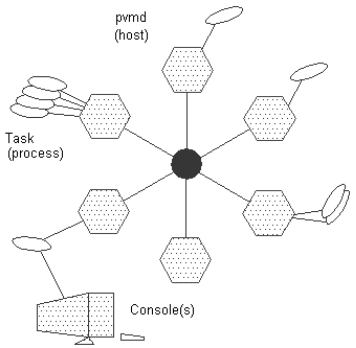
- Ứng dụng trên PVM có thể viết bằng ngôn ngữ C/C ++ hoặc FORTRAN 77.
- Thuật giải có thể song song hóa bằng cách dùng các cấu trúc truyền thông điệp với các hàm thư viện như **`pvm_send(),pvm_recv()`** để gởi và nhận dữ liệu.

Các hàm nầy là một bộ phận thứ hai của PVM, bên cạnh *pvmd* như là một PVM *daemon process*

Kiến trúc vật lý của PVM



Mô hình logic của PVM



8.5 Cài đặt PVM

- Có nhiều tập tin dưới dạng nén khác nhau của PVM, chúng ta có thể sử dụng tập tin **pvm3.4.6.tgz (**http://www.netlib.org/pvm3/pvm3.4.6.tgz**)**
- Đây là bản mới nhất được cập nhập vào 02/02/2009,
- Hiện nay PVM đã ổn định nên không có phiên bản mới hơn.

PVM trên Internet

- Giới thiệu về pvm3 và tải các tiện ích và tập tin pvm3.4.6.tgz để sử dụng (http://www.netlib.org/pvm3/)
- C++ Interface to PVM: http://www.informatik.uni-stuttgart.de/ipvr/bv/cppvm/
- Về XPVM: http://www.netlib.org/utk/icl/xpvm/xpvm.html

- Thông thường PVM được cài đặt để nhiều người cùng sử dụng, hoặc cho nhiều đề án khác nhau của cùng một người,
- Trong cả hai trường hợp PVM đều có mục tiêu sử dụng chung.

- Giả sử cần install PVM trên user có tên gọi lang của hệ điều hành LINUX
- Các bước sau đây cần tiến hành (giả sử tập tin **pvm3.4.6.tgz** đã có trên **$HOME**

```
$ tar xvfz pvm3.4.6.tgz
```

- Khi đó trên **$HOME** có thư mục **pvm3**

- Giả sử đang sử dụng Bash shell, cần đặt đường dẫn và biến môi trường sau đây trong tập tin **$HOME/.bashrc** (*Trường hợp cần dùng chung, đặt trong tập tin* **/etc/profile**).

```
$ export PVM_ROOT=$HOME/pvm3
$ export PVM_ARCH=LINUX
$ PATH=$PATH:$PVM_ROOT/lib
```

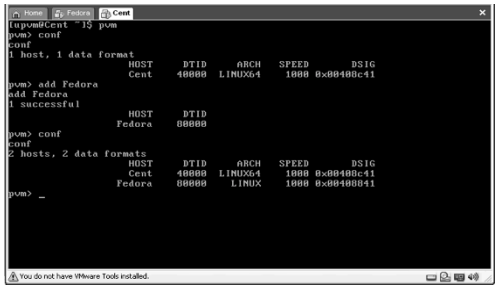

- Sau khi biến môi trường và đường dẫn đã được kích hoạt, biên dịch PVM bằng các lệnh để cài đặt PVM lên máy:

```
$ cd $HOME
$ make
```

8.6 Sử dụng PVM

- Trước hết phải kích hoạt để PVM làm việc

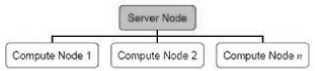
```
$ pvm
pvm>
```



- Tại dấu nhắc **pvm**, có thể thực hiện các lệnh như

```
pvm> add
pvm> delete
pvm> conf
```

- Để thêm, xóa, coi cấu hình của hệ thống máy ảo.



- Có thể quay về dấu nhắc UNIX để làm việc bằng lệnh **quit**

```
pvm>quit
$
```

- Hoặc để thoát ra khỏi, dùng lệnh

```
pvm>halt
$
```

Lưu ý

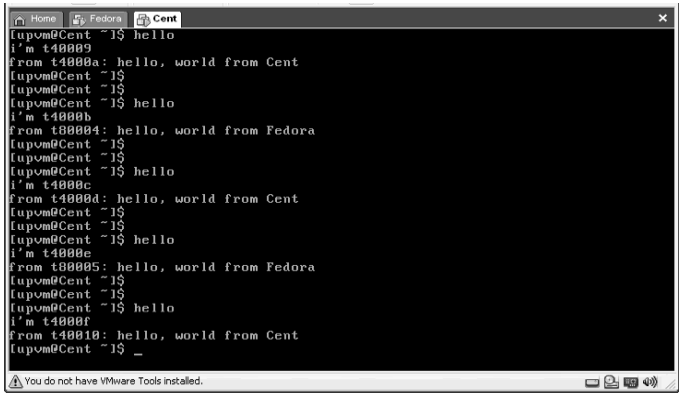
- PVM daemon hoạt động theo cơ chế remote shell, vì vậy cần phải có hostname của các máy PVM trong tập tin **/etc/hosts.equiv**. Hoặc trong các tập tin **$HOME/.rhosts.**
- Các chương trình thi hành bằng lệnh **pvm_spawn()** phải được chỉ đường dẫn tuyệt đối, hoặc được lưu trữ trong thư mục **$PVM_ROOT/bin/$PVM_ARCH**.

8.7 Ví dụ trong PVM

- Thi hành chương trình hello.c trong **$PVM_ROOT/examples**.
- Thực hiện các lệnh
  - **cd $HOME/folder**
  - **$aimk hello hello_other**
  - **$./hello**

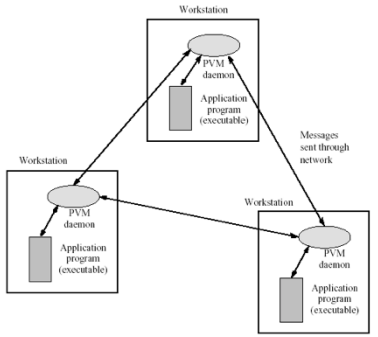
Để thực hiện, có makefile.aimk như sau:

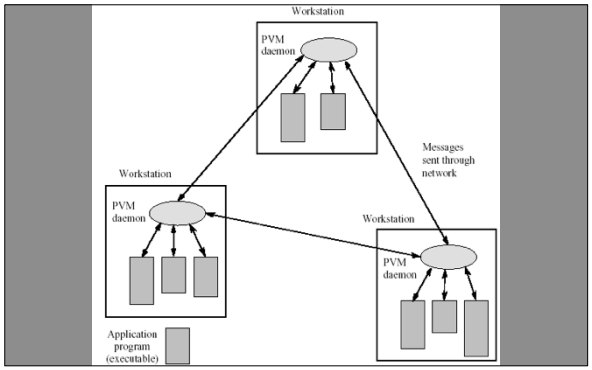
```
SDIR   =  $(HOME)/folder
XDIR   =  $(PVM_ROOT)/bin/$(PVM_ARCH)
INC=   $(ARCHCFLAGS) -I$(PVM_ROOT)/include
LIB=   -L$(PVM_ROOT)/lib/$(PVM_ARCH) -lpvm3
PROGS  =  hello hello_other
default:  hello
hello: $(SDIR)/hello.c
   $(CC) -o $@ $(SDIR)/$@.c $(INC) $(LIB)
hello_other:  $(SDIR)/hello_other.c
   $(CC) -o $@ $(SDIR)/$@.c $(INC) $(LIB)
   mv $@ $(XDIR)
```

8.8 Lập trình với PVM

- Thực chất của việc Parallel Programming trên distributed system là truyền thông điệp.
- Để chương trình thi hành trên virtual machine (VM), cần có pvmd (PVM daemon) hoạt động trên các node (các workstation) của VM này.
- Các application program (Executable) được nạp vào các workstation



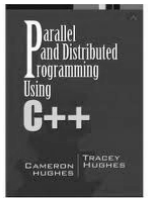


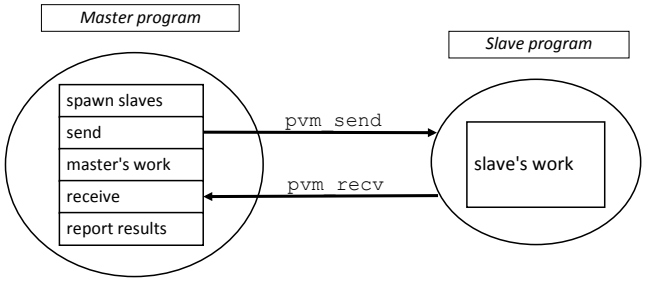
Mô hình lập trình

- Trong PVM có 2 mô hình lập trình thông dụng:
  - Mô hình master-slave
  - Mô hình task-to-task

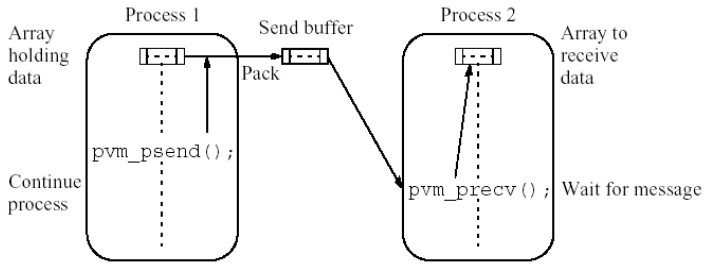
a) Mô hình master- slave

- Trong mô hình thiết kế *master-slave*, một máy chủ điều khiển sự hoạt động của các máy còn lại như là các *slave* thông qua các task ID (*identify*)





Hàm gởi và nhận cơ bản



- Trong mô hình master-slave, chương trình master phát sinh và điều khiển một vài chương trình slave để thực hiện các tính toán.
- PVM không có một hạn chế gì trên mô hình nầy, bất kỳ một task PVM nào cũng có thể đóng vai trò của một master.

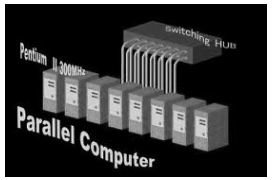
- Mô hình master-slave là một mô hình thuận tiện để minh họa thuật giải.
- Chương trình master gọi **`pvm_spawn()`** để thi hành một số các chương trình slave trên các máy khác trong hệ thống PVM.


```
int numt = pvm_spawn(char *task, char **argv, int
flag, char *where, int ntask, int *tids)
```

- task: tên tập tin thi hành có trên host mà nó thi hành. Tên này có thể là tập tin trong vùng tìm kiếm của PVM, hoặc chỉ các absolute path.
  - By default, PVM looks for executable in **`$PVM_ROOT/bin/$PVM_ARCH/`**

```
int numt = pvm_spawn(char *task, char **argv, int
flag, char *where, int ntask, int *tids)
```

- argv: các argument của tập tin thi hành. Nếu tập tin thi hành không có argument, tham số này có giá trị là **NULL**.




```
int numt = pvm_spawn(char *task, char **argv,
int flag, char *where, int ntask, int *tids)
```

- flag: có thể là tổng của các giá trị:
  - PvmTaskDefault 0: PVM can choose any machine to start task
  - PvmTaskHost 1: "where" specifies a particular host
  - PvmTaskArch 2: "where" specifies a type of architecture
  - PvmTaskDebug 4: Start up processes under debugger
  - PvmTaskTrace 8: Processes will generate PVM trace data
  - PvmMppFront 16: Start process on MPP front-end
  - PvmHostCompl 32: Use complement host set


```
int numt = pvm_spawn(char *task, char **argv, int
flag, char *where, int ntask, int *tids)
```

- where: depending on value of flag, can specify a hostname or an architecture.
  - This argument can also be used to spcify a custom working directory for each given spawn command. For example,

  "mar.lang.ac.vn:/home/lang/project"


```
int numt = pvm_spawn(char *task, char **argv, int
flag, char *where, int ntask, int *tids)
```

- ntask: number of copies of executable to start up
- tids: integer array of length at least ntask, storing the TID of PVM processes started by this pvm_spawn call