Biên dịch và chạy chương trình trên CCS

1. Thông tin chung về hệ thống

Hiện tại, tài nguyên tính toán của **Trung tâm Khoa học tính toán (CCS)** bao gồm 09 máy tính (node1, .., node9), mỗi máy có 4 processors. Các máy từ node1 đến node7 có 4GB bộ nhớ trong, node8 có 16 GB bộ nhớ trong, node9 có 32 GB bộ nhớ trong. Máy node7 có dung lượng đĩa lớn nhất, khoảng 4.3 TB, dùng để chứa dữ liệu, chương trình của người sử dụng (/home/). Thư mục này được được share cho tất cả các nodes thông qua NFS. Vì vậy node7 được sử dụng làm máy **master**. Ngoài ra, trên mỗi node còn có vùng đĩa cục bộ /temp0/ dùng để lưu dữ liệu tạm thời trong quá trình xử lý của các users và tăng hiệu năng đối với một số loại chương trình liên quan đến đọc/ghi trên ổ đĩa cứng, các files trong thư mục này sẽ tự động xóa sau 15 ngày kể từ khi tạo ra.

Tất cả 9 nodes của hệ thống được cài đặt hệ điều hành linux debian và cấu hình thành một PC cluster với thư viện xử lý song song MPITCH2 và Bộ quản lý chương trình PBS Torque. Người sử dụng có thể truy cập và khai thác các tài nguyên hệ thông thông qua máy Master (node7) với tên internet toàn cầu là: **ccs1.hnue.edu.vn** thông qua giao thức **ssh**. Hệ thống đã cài đặt các trình biên dịch cho các ngôn ngữ lập trình thông dụng như C, C++, java, Fortran, matlab, v.v.

Hệ thống máy tính của Trung tâm thường dùng để chạy các chương trình lớn, yêu cầu nhiều bộ xử lý và bộ nhớ. Sau khi truy cập hệ thống, người sử dụng có thể viết (hoặc gửi chương trình từ máy tính cá nhân của mình lên), biên dịch và chạy chương trình trên hệ thống của Trung tâm. Có 2 loại chương trình: single-node program (chương trình đơn, chạy trên 1 bộ xử lý) và multi-node program (chương trình song song); và có 2 cách thực hiện chương trình trên hệ thống PC cluster của CCS: chạy trực tiếp trên nền hệ điều hành và chạy thông qua Bộ quản lý chương trình Torque.

2. Chương trình chạy trên 1 bộ xử lý và chương trình song song

Không phải cứ đưa chương trình lên hệ thống PC cluster là nó đều chạy song song cả. Thông thường chúng ta viết chương trình thì chương trình đó chạy trên 01 bộ xử lý. Ví dụ bạn có thể dùng chương trình soạn thảo (**vim** hoặc **mcedit**) viết chương trình c++ chạy trên 01 bộ xử lý, file chương trình có tên là *hello.cpp* và có nội dung dưới đây:

```
#include <iostream>
using namespace std;
int main () {
   cout << "Hello World" << endl;
   return 0;
}
```

Sau đó biên dịch và chạy như sau:

c++ hello.cpp –o hello

./hello

Trên màn hình sẽ hiện lên kết quả thực hiện là thông báo `Hello World.`

Muốn viết chương trình song song, bạn phải dùng thư viện MPI và trình biên dịch mpic++ (hoặc mpicc, mpif77, mpif90 tùy theo ngôn ngữ lập trình bạn thạo). Ví dụ, bạn soạn file chương trình c++ có tên là *mpi_hello.cpp* và nội dung như sau:

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char ** argv) {

     int size,rank;
     int length;
     char name[80];
     MPI_Status status;
     int i;

     MPI_Init(&argc, &argv);
     // note that argc and argv are passed by address

     MPI_Comm_rank(MPI_COMM_WORLD,&rank);
     MPI_Comm_size(MPI_COMM_WORLD,&size);
     MPI_Get_processor_name(name,&length);

     if (rank==0) {
     // server commands
            printf("Hello MPI from the server process!\n");
            for (i=1;i<size;i++) {

     MPI_Recv(name,80,MPI_CHAR,i,999,MPI_COMM_WORLD,&status);
                    printf("Hello MPI!\n");
                    printf(" mesg from %d of %d on %s\n",i,size,name);
            }
     } else {
            // client commands
            MPI_Send(name,80,MPI_CHAR,0,999,MPI_COMM_WORLD);
     }

     MPI_Finalize();
}
```

Sau đó, biên dịch bằng trình biên dịch:

mpic++ mpi_hello.cpp –o mpi_hello

Khi này, file chương trình thực hiện được có tên là *mpi_hello* (không có phần đuôi) đã được tạo ra (có thể xem bằng lệnh **ls** của linux). Bình thường, chúng ta thực hiện chương trình này bằng lệnh của MPI như sau:

mpiexec –np <số bộ xử lý> mpi_hello

Tuy nhiên, đê thực hiện chương trình có sử dụng nhiều bộ xử lý như **mpi_hello** trên hệ thống PC cluster của Trung tâm, chúng ta phải submit chương trình thông qua Bộ quản lý chương trình PBS Torque với 3 bước như sau:

Bước1: Tạo file PBS script *mpi_hello.script* có nội dung đơn giản như sau:

```
#!/bin/sh
/usr/local/bin/mpiexec /home/hoanpt/mpi/mpi_hello
```

(Chú ý, phải đánh đường dẫn đầy đủ đến các files **mpiexec**, `/usr/local/bin/`, và **mpi_hello,** `/home/hoanpt/mpi/` trong script, bạn phải thay đường dẫn `/home/hoanpt/mpi/` bằng đường dẫn đến file chương trình của bạn)

Bước 2: Submit file PBS script thông qua lệnh **qsub** của PBS như sau:

qsub -q l1 -l nodes=2:ppn=4 mpi_hello.script

Ở đây, tham số **-q l1** để yêu cầu PBS thực hiện chương trình trên hàng đợi chương trình có tên là **l1** (thông tin về các hàng đợi chương trình trên hệ thống của Trung tâm xem phần dưới), tham số -l nodes=2::ppn=4 để yêu cầu PBS thực hiện chương trình trên 2 nodes và mỗi node 4 bộ xử lý.

Khi thực hiện lệnh trên, PBS sẽ tự động cấp phát bộ xử lý và bộ nhớ còn rỗi để thực hiện chương trình theo yêu cầu mổ tả trong file *mpi_hello.script* hoặc trên chính dòng lệnh **qsub** (được chỉ ra trong các tham số như **–l**, **-q**, v.v.). Sau khi thực hiện chương trình, PBS sẽ sinh ra 2 files mới: *mpi_hello.script.oXX,* chứa kết quả thực hiện chương trình (nếu không có lỗi) và *mpi_hello.script.eXX,* chứa thông báo lỗi (nếu có) khi thực hiện chương trình (ở đây *XX* là một số nguyên do PBS sinh ra trong quá trình quản lý các chương trình, có thể biết các số này bằng lệnh **qstat**).

Bước 3: Đọc kết quả thực hiện chương trình: Kết quả và thông báo lỗi nếu có sẽ được PBS ghi vào các files *mpi_hello.script.oXX* và *mpi_hello.script.eXX* như đã nó ở trên. Dùng lệnh **cat** của linux để xem:

cat mpi_hello.script.oXX

Kết quả in ra sẽ là như sau (có thể khi chạy chương trình ở thời điểm khác nhau thì thông tin về nodes sẽ khác nhau):

```
Hello MPI from the server process!
Hello MPI!
 mesg from 1 of 8 on node4
Hello MPI!
 mesg from 2 of 8 on node4
Hello MPI!
 mesg from 3 of 8 on node4
Hello MPI!
 mesg from 4 of 8 on node3
```

```
Hello MPI!
 mesg from 5 of 8 on node3
Hello MPI!
 mesg from 6 of 8 on node3
Hello MPI!
 mesg from 7 of 8 on node3
```

3. Chạy chương trình trên hệ điều hành hay thông qua Bộ quản lý chương trình PBS Torque

Ở trên chúng ta đã thấy có 2 cách thực hiện chương trình: chạy tương tác trên nền hệ điều hành (ví dụ **./hello**) hoặc thông qua Bộ quản lý chương trình PBS Torque (**qsub mpi_hello.script**). Nếu chương trình của bạn nhỏ (yêu cầu 1 bộ xử lý, chạy trong thời gian ngắn, và yêu cầu ít bộ nhớ) thì có thể chạy trực tiếp trên hệ điều hành như ví dụ **hello** ở trên. Nếu chương trình lớn, đòi hỏi nhiều bộ xử lý, thời gian thực hiện lâu thì phải thực hiện thông quả Bộ quản lý chương trình PBS Torque như ví dụ **mpi_hello** ở trên. Chúng tôi khuyến cáo tất cả các chương trình chạy trên Trung tâm Khoa học tính toán đều nên chạy thông qua PBS Torque (chú ý rằng tất cả chương trình, kể cả chương trình chạy trên 01 bộ xử lý, đều có thể thực hiện thông qua PBS) để các tài nguyên của Trung tâm được khai thác hiệu quả và để PBS Torque phục vụ các chương trình của tất cả người sử dụng một cách tối ưu.

Để thực hiện một chương trình thông qua Bộ quản lý chương trình PBS, chúng ta làm theo 3 bước như trong ví dụ **mpi_hello** ở trên, cụ thể:

Bước 1: Chuẩn bị file PBS script, trong đó mô tả các yêu cầu về số node, số bộ xử lý, số lượng bộ nhớ, thời gian, v.v. để PBS cấp phát tài nguyên khi thực hiện chương trình

Bước 2: submit file PBS script cho PBS thực hiện, trong đó có các tham số cung cấp thêm yêu cầu ngoài thông tin mô tả trong file script.

Bước 3: Đọc các file kết quả hoặc file thông báo lỗi.

Lấy ví dụ khác, để chạy chương trình **hello** trong ví dụ đầu tiên thông qua Bộ quản lý chương trình PBS, sau khi biên dịch (bằng c++) ra file **hello**, thay vì thực hiện trên nền hệ điều hành linux (**./hello**), chúng ta làm như sau:

- ***Bước 1***: Tạo file PBS script có tên là *hello.script* có nội dung như sau:

  ```
  #!/bin/sh
  /home/hoanpt/mpi/hello
  ```

  (Chú ý, phải đánh đường dẫn đầy đủ đến file **hello,** `/home/hoanpt/mpi/` trong script, bạn phải thay đường dẫn này bằng đường dẫn đến file chương trình của bạn, để biết xem bạn đang trong thư mục nào, dùng lệnh **pwd** của linux)

- ***Bước 2:*** Thực hiện lệnh

qsub –l nodes=node2:ppn=4 hello.script

- ***Bước 3:*** Đọc kết quả (dùng lệnh **qstat** hoặc **ls** để biết các đuôi *XX* của các file *hello.script.oXX*, *hello.script.eXX* do PBS sinh ra)

  cat hello.script.oXX

  Khi đó, nội dung kết quả ghi trong file *hello.script.oXX* sẽ là

  ```
  Hello World
  ```

Trong ví dụ này, chương trình **hello** là chương trình được viết chạy trên 1 bộ xử lý, chúng ta đã yêu cầu PBS cấp tài nguyên là node2, số bộ xử lý là 4 (bước 2), nhưng cuối cùng thì chương trình chỉ thực hiện trên 1 bộ xử lý. Vì vậy khi submit jobs thông qua PBS, chúng ta cần cố gắng yêu cầu PBS cấp vừa đủ tài nguyên cho chương trình của mình. Để PBS quản lý và cấp phát tài nguyên hợp lý, chúng tôi cũng đã tạo sẵn một số hàng đợi (**queue**) của PBS Torque với các tài nguyên mặc định như sau:

l1: gồm các node1::ppn=4 + 1:node2:ppn=4+1:node3:ppn=4+1:node4:ppn=4+1:node5:ppn=4+1:node6:ppn=4

l9: gồm các nodes từ 1 đến 9

p8: gồm chỉ node8:ppn=4

p9: gồm chỉ node9:ppn=4

Các bạn có thể submit các jobs vào hàng đợi phù hợp để PBS Torque khai thác tài nguyên và quản lý chương trình hiệu quả nhất.

4. Tài liệu tham khảo thêm:

- Hướng dẫn về PBS scripts:
  http://www.hpcc.nectec.or.th/wiki/index.php/PBS_Job_Submission

  http://www.hpc.cam.ac.uk/user/jobs.html

- MPI:

  ftp://ftp.mcs.anl.gov/pub/mpi/mpich2-doc-user.pdf