COMPUTACIÓ D'ALTES PRESTACIONS – PAC 2

Semestre Setembre 2015

- Presentació i objectius
- Enunciat de la pràctica
- Criteris d'avaluació
- Format de lliurament
- Data de lliurament

Presentació i objectius

Objectius

Aquesta pràctica té com a objectius introduir els conceptes bàsics de model de programació MPI i del seu entorn d'execució. Per a la realització es posaran en pràctica alguns dels conceptes presentats en aquesta assignatura.

Presentació de la pràctica

Cal lliurar els fitxers amb el codi font realitzat i un document amb les respostes a les preguntes formulades, l'avaluació de rendiment i els comentaris que considereu. Tota la codificació es realitza exclusivament en llenguatge C amb les extensions associades a MPI.

Restriccions de l'entorn

Tot i que el desenvolupament de la pràctica és molt més interessant utilitzant dotzenes de computadors, s'assumeix que inicialment tindreu accés a un nombre força reduït de computadors amb diversos nuclis per node.

Material per a la realització de la pràctica

En els servidors de la UOC teniu el programari necessari per executar MPI. De tota manera, si voleu fer el vostre desenvolupament i proves en el vostre sistema local haureu de tenir instal·lat algun programari que implementi MPI. Podeu utilitzar el que millor us sembli però us aconsellem OpenMPI o MPICH2, que és una distribució lliure, portable i molt popular. Podeu descarregar-la i trobar més informació sobre la instal·lació / utilització en la següent adreça:

http://www.mcs.anl.gov/research/projects/mpich2/

En qualsevol cas s'espera que realitzeu la vostra investigació i si cal que es debati en el fòrum de l'assignatura.

S'espera que es produeixi debat en el fòrum de l'assignatura per aprofundir en l'ús dels models de programació i els sistemes paral·lels proporcionats.

Enunciat de la pràctica

1. Entorn d'execució

La primera part d'aquesta pràctica consisteix a familiaritzar-se amb l'entorn i ser capaç de compilar i executar programes MPI.

Utilitzarem el següent programa MPI que implementa el típic programa que retorna "hello world" (hello.c):

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

int main(int argc, char **argv)
{
  MPI_Init(&argc, &argv);

  printf("Hello World!\n");

  MPI_Finalize();
}
```

podeu veure que cal incloure `"mpi.h"` i que és imprescindible començar el vostre programa MPI amb `MPI_Init` i al final acabar-lo amb `MPI_Finalize`. Aquestes crides s'encarreguen de treballar amb el runtime de MPI per a la creació i destrucció de processos MPI.

A continuació es presenten els passos necessaris per compilar i executar un simple "hello world" utilitzant MPI:

- Compilar el programa `hello.c` amb mpicc (més detalls a la documentació).
- Executar el programa MPI hello world mitjançant SGE. Un script d'exemple per a l'execució del programa hello utilitzant 8 processos MPI el podeu trobar a continuació:

```
#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N hello
#$ -o hello.out.$JOB_ID
#$ -e hello.err.$JOB_ID
#$ -pe orte 8

mpirun -np 8 ./hello
```

L'opció –pe orte 8 li indica a SGE que necessiteu 8 nuclis (que s'assignaran en més d'un node al clúster de la UOC). La comanda mpirun s'encarrega de cridar al runtime de MPI per realitzar l'execució del programa MPI. L'opció –np 8 li indica al runtime de MPI que utilitzi 8 processos per a l'execució del programa MPI i al final indiquem el nom del binari del programa. Podem estudiar altres opcions de mpirun.

Preguntes:

1.1. Quin és el resultat de l'execució del programa MPI? Per què?

1.2. Expliqueu què passa si utilitzeu l'opció –np 4 en canvi de –np 8 en el mpirun.

2. Processos MPI

A continuació introduïm un programa MPI que inclou crides bàsiques que trobaren en qualsevol programa MPI. Noteu que la variable "rank" és bàsica en l'execució de programes MPI ja que permet identificar el número de procés dins d'un comunicador (en aquest cas MPI_COMM_WORLD). Si us plau, consulteu els apunts per més detalls.

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

int main(int argc, char **argv)
{
  int rank, numprocs;
  char hostname[256];

  MPI_Init(&argc,&argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);

  gethostname(hostname,255);

  printf("Hello world! I am process number %d of %d MPI processes on host %s\n", rank,
numprocs, hostname);

  MPI_Finalize();

  return 0;
}
```

2.1 Quin és el resultat de l'execució del programa MPI? Per què?

3. Pas de missatges punt a punt

En aquest apartat us demanem el primer exercici de programació. Us facilitem un altre exemple d'aplicació MPI anomenat "hellompi.c" juntament amb l'anunciat de la PAC. Aquest programa està dissenyat per ser executat només amb 2 processos MPI i bàsicament s'encarrega de fer de pas de missatges entre els dos processos MPI.

3.1 Quina és la mida del missatge que es transfereix entre processos?

Es demana que realitzeu una variant del programa "hellompi.c" que faci que cadascun dels processos MPI faci l'enviament d'un missatge MPI a la resta de processos.

Un exemple de sortida amb 3 processos MPI es mostra a continuació:

```
Proc #1 sending message to Proc #0
Proc #1 sending message to Proc #2
Proc #1 received message from Proc #0
Proc #1 received message from Proc #2
Proc #0 sending message to Proc #1
Proc #0 sending message to Proc #2
Proc #0 received message from Proc #1
Proc #0 received message from Proc #2
Proc #2 sending message to Proc #0
Proc #2 sending message to Proc #1
Proc #2 received message from Proc #0
Proc #2 received message from Proc #1
```

3.2 Proporcioneu el codi del vostre programa.

4. Ordre de l'escriptura de la sortida estàndard

En aquest apartat es demana que implementeu un programa MPI que realitzi pas de missatges entre processos MPI en forma d'anell, és a dir, que cada procés MPI faci l'enviament d'un missatge al següent rank i en rebi un del rank anterior (excepte el primer i l'últim que han de tancar l'anell).

Així doncs, donats N processos MPI, s'espera que el procés MPI amb rank 0 faci l'enviament d'un missatge al procés amb rank 1, el procés amb rank 1 rebi el missatge el procés amb rank 0 i faci l'enviament d'un missatge al procés amb rank 2, i així fins arribar al procés amb rank N-1 que rebrà un missatge del procés N-2 i farà l'enviament d'un missatge al procés amb rank 0.

Podeu utilitzar com a referència la següent línia com a exemple per a la sortida del vostre programa.

```
Proc 2: received message from proc 1 sending to 3
```

4.1 Proporcioneu el codi del vostre programa.

4.2 En quin orde apareixen els missatges de sortida? Per què?

[La resta d'exercicis es proporcionaran amb la segona part de la PAC]

Criteris d'avaluació

Es valorarà el correcte ús del model de programació MPI. També es tindrà en compte la correcta programació, estructura i funcionament dels programes i la discussió dels mateixos.

Format de lliurament

Es crearà un fitxer en format PDF amb tota la informació.

Si els scripts o codis són llarg per afegir-los al document de l'entrega (no s'esperen que siguin massa sofisticats), podeu adjuntar-los amb la comanda següent:

```
$ tar cvf tot.tar fitxer1 fitxer2 ...
```

es crearà el fitxer "tot.tar" on s'hauran emmagatzemat els fitxers "fitxer1", fitxer2" i . . .

Per llistar la informació d'un fitxer `tar` es pot utilitzar la comanda següent:

```
$ tar tvf tot.tar
```

Per extraure la informació d'un fitxer `tar` es pot utilitzar:

```
$ tar xvf tot.tar
```

El nom del fitxer tindrà el format següent: "Cognom1Cognom2PAC2.pdf" (o *.tar). Els cognoms s'escriuran sense accents. Per exemple, l'estudiant Marta Vallès i Marfany utilitzarà el nom de fitxer següent: `VallesMarfanyPAC2.pdf (o *.tar)`

Data de lliurament

Dilluns 2 de Novembre de 2015.