COMPUTACIÓ D'ALTES PRESTACIONS – PAC 3

Semestre Setembre 2018

- 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 del 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 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 MPI

La aquesta part 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. **Quin és el resultat de l'execució del programa MPI? Per què?**
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;
}
```

Preguntes:

3. 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.

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
```

Preguntes:

4. 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
```

Preguntes:

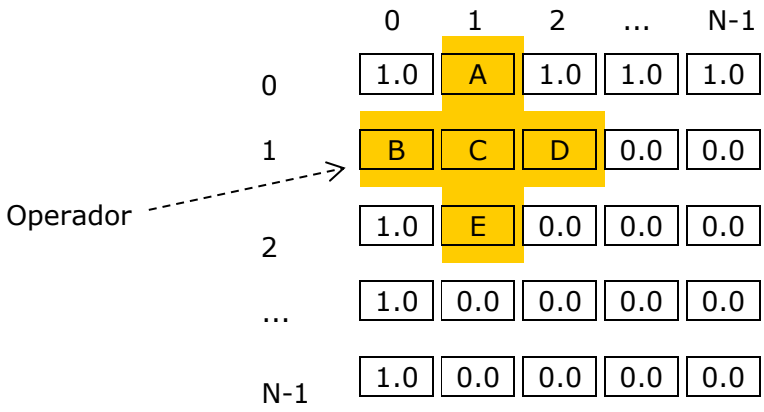
5. Proporcioneu el codi del vostre programa.

6. En quin orde apareixen els missatges de sortida? Per què?

5. Implementació d'un solver mitjançant MPI

Aquesta part d'aquesta PAC consisteix en implementar la versió paral·lela mitjançant MPI d'un programa seqüencial que implementa un solver. En primer lloc, el programa inicialitza una matriu (de tamany NxN) i després aplica un operador (també conegut com a *stencil*) sobre tota la matriu durant un determinat nombre d'iteracions (ITERS). Per a cada element, l'operador aplica la fórmula:

C '= 0,3 * (C + A + B + D + E), on C ' és el valor de C en la propera iteració.



Es proporciona un programa seqüencial de referència que implementa el solver (solver.c)

Nota: Per implementar aquest problema es recomana definir la matriu de tamany (N+2)x(N+2) per tal de poder aplicar l'operador sense sortir de rang - aquesta tècnica consisteix en envoltar la matriu (amb les també anomenades *halo zones*). També es proporciona un exemple addicional (stencil.c) que utilitza aquesta tècnica en un programa que implementa la propagació d'energia en un espai 2D. Podeu fer algunes proves i visualitzar el fitxer de sortida (heat.svg), per exemple:

```
./stencil 100 10 50
./stencil 100 10 150
./stencil 100 10 500
```

Preguntes:

7. **Implementeu el solver.c mitjançant MPI. Expliqueu les vostres decisions/observacions (per exemple, si heu fet servir crides MPI síncrones o asíncrones/per què).**
8. **Avalueu l'execució del vostre programa MPI en relació a la versió seqüencial. En particular, es demana que estudieu (i proporcioneu gràfiques) el speedup respecte el número de nuclis utilitzats, respecte el tamany de la matriu i respecte el número d'iteracions. Nota: podeu parametritzar tamany/iteracions.**

6. Eines d'avaluació de rendiment (TAU)

Aquest apartat consisteix en realitzar un estudi de rendiment del solver implementat.

Per a la realització d'aquest estudi utilitzarem el paquet d'anàlisi de rendiment TAU (https://www.cs.uoregon.edu/research/tau/home.php). Aquest paquet es pot utilitzar a nivell d'usuari i es troba en la següent ubicació al clúster de la UOC

```
/export/apps/tau/
```

Us caldrà també afegir el següent a $PATH:

```
export PATH=$PATH:/export/apps/tau/x86_64/bin
```

El paquet de TAU porta tant les eines d'instrumentació i d'anàlisi com el visualitzador de traces jumpshot. Els passos bàsics per a la utilització de TAU i les eines relaciones es proporcionen a continuació:

- **Ús recomanant de TAU per a la PAC (amb instrumentació dinàmica)**

En el vostre script SGE us caldrà utilitzar el següent:

```
export TAU_TRACE=1
(activates tracing)

mpirun –np NUM_PROCS tau_exec ./YOUR_MPI_PROGRAM
(dynamic instrumentation – recommended)
```

- **Anàlisi amb pprof (podeu explorar pel vostre compte l'eina paraprof si voleu aprofundir)**

Només cal executar la comanda pprof en el directori on s'hagi executat el programa MPI i on s'hauran generat els fitxers "`profile.*.*.*`". Un exemple bàsic es mostra a continuació:

```
[ivan@eimtarqso p4]$ pprof
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        3,000        3,015           1          12    3015921 .TAU application
  0.3           10           10           1           0      10235 MPI_Init()
  0.1            3            3           1           0       3450 MPI_Finalize()
  0.0        0.863        0.863           2           0        432 MPI_Barrier()
  0.0        0.545        0.545           3           0        182 MPI_Send()
  0.0        0.018        0.018           3           0          6 MPI_Recv()
  0.0        0.001        0.001           1           0          1 MPI_Comm_rank()
  0.0        0.001        0.001           1           0          1 MPI_Comm_size()

NODE 1;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        3,000        3,015           1          12    3015923 .TAU application
  0.3           10           10           1           0      10190 MPI_Init()
  0.1            3            3           1           0       3470 MPI_Finalize()
```

Podeu trobar les opcions d'aquesta comanda a continuació:

```
usage: pprof [-c|-b|-m|-t|-e|-i|-v] [-r] [-s] [-n num] [-f filename] [-p] [-l] [-d] [node
numbers]
 -a : Show all location information available
 -c : Sort according to number of Calls
 -b : Sort according to number of suBroutines called by a function
 -m : Sort according to Milliseconds (exclusive time total)
 -t : Sort according to Total milliseconds (inclusive time total)  (default)
 -e : Sort according to Exclusive time per call (msec/call)
 -i : Sort according to Inclusive time per call (total msec/call)
 -v : Sort according to Standard Deviation (excl usec)
 -r : Reverse sorting order
 -s : print only Summary profile information
 -n <num> : print only first <num> number of functions
 -f filename : specify full path and Filename without node ids
 -p : suPpress conversion to hh:mm:ss:mmm format
 -l : List all functions and exit
 -d : Dump output format (for tau_reduce) [node numbers] : prints only info about all
contexts/threads of given node numbers
```

- **Visualització de traces amb jumpshot**

Per poder entendre millor els possibles problemes relacionats amb el pas de missatges en MPI s'acostuma a visualitzar les traces de l'execució dels programes i fer-hi un estudi amb eines que treballin amb aquestes traces. Cal dir que l'estudi de rendiment mitjançant aquestes tècniques és un art i es requereix d'experiència per a fer anàlisis en profunditat. En aquesta PAC es demana que observeu les característiques més bàsiques i us familiaritzeu una mica amb aquestes eines.

El paquet de TAU porta l'eina jumpshot la qual podeu utilitzar mitjançant els següents passos:

```
tau_treemerge.pl
tau2slog2 tau.trc tau.edf -o tau.slog2
jumpshot tau.slog2
```

Si feu servir jumpshot directament al clúster de la UOC el temps de resposta serà molt gran i és possible que sigui impracticable. Per això es recomana que us descarregueu l'eina en el vostre entorn local. L'eina es troba en el següent directori del paquet de TAU:

```
/export/apps/tau/x86_64/bin/tau2slog2
```

Hi podreu trobar el fitxer `jumpshot.jar`

```
/export/apps/tau/x86_64/lib/jumpshot.jar
```

Només cal que tingueu instal·lat l'entorn d'execució de Java en el vostre sistema (ja sigui Linux, mac o Windows) per fer servir l'eina.

Preguntes:

9. Feu un breu estudi del comportament de la vostra implementació del solver mitjançant les eines introduïdes. Podeu incloure resums estadístics, captures de pantalla i els comentaris que considereu apropiats.

7. Eines d'avaluació de rendiment (Extrae/Paraver)

Aquest apartat consisteix en realitzar un estudi de rendiment de algunes de les aplicacions dels NAS Parallel Benchmarks (NPB) - https://www.nas.nasa.gov/publications/npb.html. En concret es demana que estudieu els benchmarks EP, BT i CG (classe S). La motivació d'aquest exercici és entendre com fer un estudi de rendiment d'aplicacions no conegudes a priori i de les quals no es pugui disposar del codi font.

Els binaris es troben als següents directoris:

```
/export/apps/aca/benchmarks/NPB3.2/NPB3.2-MPI/bin/ep.S.16
/export/apps/aca/benchmarks/NPB3.2/NPB3.2-MPI/bin/bt.S.16
/export/apps/aca/benchmarks/NPB3.2/NPB3.2-MPI/bin/cg.S.16
```

Aquests són de la classe S que és la més petita dels NPB i estan pensats per executar-los amb 16 processos MPI.

En aquest cas farem servir l'eina Extrae (https://tools.bsc.es/extrae) desenvolupada pel Barcelona Supercomputing Center

Per fer servir extrae haureu de definir les següents variables i executar el següent:

```
export EXTRAE_HOME=/export/apps/extrae
cp /export/apps/extrae/share/example/MPI/extrae.xml YOUR_DIRECTORY
source /export/apps/extrae/etc/extrae.sh
export EXTRAE_CONFIG_FILE=YOUR_DIRECTORY/extrae.xml
export LD_PRELOAD=/export/apps/extrae/lib/libmpitrace.so
```

i fer les execucions com habitualment, per exemple:

```
mpirun -np 16 ./filename
```

Com a resultat de l'execució de l'aplicació instrumentada s'obté una traça que està composada de tres fitxers: `filename.prv`, `filename.row` i `filename.pcf`.

Per visualitzar i analitzar la traça obtinguda haureu d'utilitzar l'eina Paraver (https://tools.bsc.es/paraver), la qual està disponible per múltiples plataformes i haureu d'instal·lar en el vostre terminal.

Preguntes:

10. Feu un breu estudi comparatiu dels benchmarks NPB EP, BT i CG utilitzant Extrae/Paraver. Podeu incloure resums estadístics, captures de pantalla i els comentaris que considereu apropiats.

Criteris d'avaluació

Es valorarà el correcte ús del model de programació MPI. També es tindrà en compte la utilització de les eines d'avaluació de rendiment i el comentaris relacionats.

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: "Cognom1Cognom2PAC3.pdf" (o *.tar). Els cognoms s'escriuran sense accents. Per exemple, l'estudiant Marta Vallès i Marfany utilitzarà el nom de fitxer següent: `VallesMarfanyPAC3.pdf (o *.tar)`

Data de lliurament

Divendres 30 de Novembre de 2018.