COMPUTACIÓ D'ALTES PRESTACIONS – PAC4

Semestre Setembre 2017

- 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 objectiu aprendre tècniques d'avaluació de rendiment d'aplicacions d'altes prestacions. Per a la realització de la PAC es posaran en pràctica diversos 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 OpenMP/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

Els materials necessaris per a la realització de la pràctica estan disponibles en aquest enunciat, al clúster de la UOC i als links proporcionats en aquest enunciat. Addicionalment es proporcionarà una demostració online.

Enunciat de la pràctica

1. Avaluació de rendiment (OpenMP)

En aquesta PAC veurem com utilitzar eines bàsiques d'avaluació de rendiment per a aplicacions de memòria compartida com ara OpenMP.

Utilitzarem un exemple il·lustratiu basat en una multiplicació de matrius sense cap optimització. Trobareu amb aquest enunciat dos programes diferents que implementen la multiplicació de matrius (mm.c i mm2.c). La diferència entre aquests dos és que en la segona versió hem intercanviat els índexs dels bucles exterior i més interior com es mostra a continuació:

mm.c

```
(...)
for (i=0;i<SIZE;i++)
 for(j=0;j<SIZE;j++)
  for(k=0;k<SIZE;k++)
    mresult[i][j]=mresult[i][j] + matrixa[i][k]*matrixb[k][j];
(...)
```

mm2.c

```
(...)
for (k=0;k<SIZE;k++)
 for(j=0;j<SIZE;j++)
  for(i=0;i<SIZE;i++)
    mresult[i][j]=mresult[i][j] + matrixa[i][k]*matrixb[k][j];
(...)
```

Per compilar aquests exemples només us cal utilitzar

```
gcc mm.c –o mm
```

Amb el tamany de matriu per defecte (SIZE) igual a 1000 obtindreu un temps d'execució ràpid però que us permetrà observar diferències entre les dues versions de la multiplicació de matrius.

Utilitzant la comanda `time`, (per exemple, `time ./mm`) podreu observar una diferència en el temps d'execució d'un 15% aproximadament.

Nota:

Recordeu que s'espera que feu les vostres execucions mitjançant el sistema de cues, si no ho feu així pot passar que tingueu degradació de rendiment totalment aleatori degut a la compartició dels recursos (per exemple degut a contenció de memòria).

És evident que hi ha un impacte en el rendiment de l'execució de la multiplicació pel fet de modificar els índexs dels bucles i, per tant, de com estem accedint a les dades en memòria. Per tant, en aquest cap l'eina del sistema time no és suficient per entendre (o comprovar quantitativament) aquesta situació.

Nota:

No optimitzeu la compilació del codi proporcionat (NO utilitzeu opcions com ara –O3) ja que les optimitzacions automàtiques reduiran els temps d'execucions i faran modificacions típiques (fer alignment, loop unrolling, etc.) que no us permetran seguir aquesta part de la PAC. Recordeu que estem treballant un exemple il·lustratiu.

Per estudiar el comportament d'aquests programes i l'impacte en la utilització dels recursos utilitzarem PAPI (Performance Application Programming Interface). PAPI és una interfície molt útil que permet utilitzar comptador hardware que hi ha disponibles en la majoria de microprocessadors moderns.

Us proporcionem un parell d'exemples de com utilitzar PAPI però s'espera que feu la vostra pròpia cerca d'informació i exemples. Podeu començar a través del següent link online:

http://icl.cs.utk.edu/papi/

PAPI està disponible al clúster de la UOC en el següent directori: `/export/apps/papi/`

En l'exemple `flops.c` i `flops2.c` que es proporcionen amb l'enunciat d'aquesta PAC es mostra un exemple d'utilització de PAPI. Aquests exemples utilitzen una interfície de PAPI que proporciona els MFLOPS obtinguts en l'execució del programes. També us proporciona el temps d'execució de forma més acurada i el temps de processador.

Per tal de compilar-los podeu fer servir la següent comanda:

```
gcc -I/export/apps/papi/include/ flops.c /export/apps/papi/lib/libpapi.a -o flops
```

Veureu que observem una diferència significativa en el FLOPS obtinguts per a les dues versions de la multiplicació de matrius.

Preguntes

1.1. Per què penseu que és aquesta diferència en el temps d'execució i en els MFLOPs?

En la última versió dels exemples proporcionats (`counters.c` i `counters2.c`) s'utilitzen comptadors hardware de forma explícita. Per tal de compilar-los només heu de fer com anteriorment:

```
gcc -I/export/apps/papi/include/ counters.c /export/apps/papi/lib/libpapi.a -o counters
```

Veureu que la sortida de l'execució d'aquests us proporciona la quantitat total d'instruccions (PAPI_TOT_INS) i operacions en coma flotant (PAPI_FP_OPS).

Podreu observar que la sortida us indica que els dos exemples estan executant el mateix número d'operacions en coma flotant (tot i que podeu veure una certa variabilitat en el número total d'instruccions). Clarament s'han analitzar altres recursos relacionats amb l'accés a memòria per entendre millor l'impacte de l'intercanvi dels índexs dels bucles.

En aquesta PAC es demana que utilitzeu altres comptadors hardware per estudiar els exemples proporcionats. Podeu consultar el significat dels comptadors utilitzats en els exemples de la PAC i també el conjunt total de comptadors hardware disponibles als processadors del clúster de la UOC i els events disponibles per a consultar mitjançant la següent comanda:

```
/export/apps/papi/bin/papi_avail
```

Obtindreu el següent resultat (parcial - la sortida està tallada):

```
[@eimtarqso]$ /export/apps/papi/bin/papi_avail
Available PAPI preset and user defined events plus hardware information.
--------------------------------------------------------------------------------
PAPI Version             : 5.5.0.0
Vendor string and code   : GenuineIntel (1)
Model string and code    : Intel(R) Xeon(R) CPU           E5603  @ 1.60GHz (44)
CPU Revision             : 2.000000
CPUID Info               : Family: 6  Model: 44  Stepping: 2
CPU Max Megahertz        : 1600
CPU Min Megahertz        : 1200
Hdw Threads per core     : 1
Cores per Socket         : 4
Sockets                  : 1
NUMA Nodes               : 1
CPUs per Node            : 4
Total CPUs               : 4
Running in a VM          : no
Number Hardware Counters : 7
Max Multiplex Counters   : 32
--------------------------------------------------------------------------------
================================================================================
  PAPI Preset Events
================================================================================
    Name        Code    Avail Deriv Description (Note)
PAPI_L1_DCM  0x80000000  Yes   No   Level 1 data cache misses
PAPI_L1_ICM  0x80000001  Yes   No   Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002  Yes   Yes  Level 2 data cache misses
PAPI_L2_ICM  0x80000003  Yes   No   Level 2 instruction cache misses
PAPI_L3_DCM  0x80000004  No    No   Level 3 data cache misses
PAPI_L3_ICM  0x80000005  No    No   Level 3 instruction cache misses
PAPI_L1_TCM  0x80000006  Yes   Yes  Level 1 cache misses
PAPI_L2_TCM  0x80000007  Yes   No   Level 2 cache misses
PAPI_L3_TCM  0x80000008  Yes   No   Level 3 cache misses
PAPI_CA_SNP  0x80000009  No    No   Requests for a snoop
PAPI_CA_SHR  0x8000000a  No    No   Requests for exclusive access to shared cache line
PAPI_CA_CLN  0x8000000b  No    No   Requests for exclusive access to clean cache line
PAPI_CA_INV  0x8000000c  No    No   Requests for cache line invalidation
PAPI_CA_ITV  0x8000000d  No    No   Requests for cache line intervention
PAPI_L3_LDM  0x8000000e  Yes   No   Level 3 load misses
PAPI_L3_STM  0x8000000f  No    No   Level 3 store misses
(...)
--------------------------------------------------------------------------------
Of 108 possible events, 58 are available, of which 14 are derived.

avail.c                                  PASSED
```

Preguntes

1.2. Quins comptadors hardware faríeu servir per estudiar les diferències entre les dues implementacions? Per què?

1.3. Proporcioneu el codi on feu servir comptadors hardware per quantificar les diferències entre els dos exemples proporcionats. Proporcioneu els resultats obtinguts.

1.4. Proporcioneu versions paral·leles (OpenMP) i els resultats obtinguts utilitzant PAPI pels dos exemples proporcionats.

2. Eines d'avaluació de rendiment (TAU)

Aquest apartat consisteix en realitzar un estudi de rendiment d'una variació del solver que es va implementar a la PAC2 (utilitzeu els codis proporcionats de referència que tenen un màxim de 20 iteracions). En particular, es vol un anàlisi de les diferència entre la versió amb crides MPI síncrones (bloquejants) i no bloquejants.

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

2.1. Feu un breu estudi comparatiu del comportament de les implementacions del solver proporcionat en aquest apartat mitjançant les eines introduïdes. Podeu incloure resums estadístics, captures de pantalla i els comentaris que considereu apropiats.

3. 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

3.1. 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 l'ús les eines d'anàlisi de rendiment i la discussió comparativa de les diferents aplicacions treballades.

Format de lliurament

Es crearà un fitxer `tar` amb tots els fitxers font de la pràctica.

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: "Cognom1Cognom2PAC4.tar". Els cognoms s'escriuran sense accents. Per exemple, l'estudiant Marta Vallès i Marfany utilitzarà el nom de fitxer següent: `VallesMarfanyPAC4.tar`

Format de lliurament

Diumenge 31 de Desembre de 2017.