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 OpenMP i del seu entorn d'execució. També es treballarà el model de programació híbrid MPI+OpenMP. 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 OpenMP i 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. Els compiladors de C actuals incorporen les extensions per OpenMP per defecte.

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. OpenMP fundamentals

The first part of this assignment consists on understanding how to program and run simple OpenMP programs.

Our first program is a "hello world" type of program as shown below:

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
int nthreads, tid;

#pragma omp parallel private(nthreads, tid)
  {

  tid = omp_get_thread_num();
  printf("Hello World from thread = %d\n", tid);

  if (tid == 0)
    {
    nthreads = omp_get_num_threads();
    printf("Number of threads = %d\n", nthreads);
    }
  }
}
```

Note that this example follows the basic OpenMP syntax:

```
#include "omp.h" int main ()
{
     int var1, var2, var3;
    // Serial code
    ...
    // Beginning of parallel section.
    // Fork a team of threads. Specify variable scoping
    #pragma omp parallel private(var1, var2) shared(var3)
    {
       // Parallel section executed by all threads
       ...
       // All threads join master thread and disband
    }
    // Resume serial code ...
}
```

2. Running OpenMP

The code shown above can be compiled with the following option:

```
gcc -fopenmp hello_omp.c -o hello_omp
```

The obtained binary is expected to be run **using SGE**; however, the following example shows how to run an OpenMP program with different numbers of OpenMP threads.

```
[ivan@eimtarqso]$ gcc -fopenmp hello_omp.c -o hello_omp

[ivan@eimtarqso]$ ./hello_omp
Hello World from thread = 2
Hello World from thread = 1
Hello World from thread = 3
Hello World from thread = 0
Number of threads = 4
##(by default OpenMP uses: OpenMP threads = CPU cores)

[ivan@eimtarqso]$ export OMP_NUM_THREADS=3
##(we specify the number of threads to be used with the environment variable)

[ivan@eimtarqso]$ ./hello_omp
Hello World from thread = 1
Hello World from thread = 0
Number of threads = 3
Hello World from thread = 2

[ivan@eimtarqso]$ export OMP_NUM_THREADS=2

[ivan@eimtarqso]$ ./hello_omp
Hello World from thread = 1
Hello World from thread = 0
Number of threads = 2
```

Note that the number of OpenMP threads can be hardcoded in your programs but we target codes that use the OMP_NUM_THREADS variable to specify the number of OpenMP threads to be used.

The following scripts illustrate two possible ways to run an OpenMP program using SGE:

```
##(omp1.sge)

#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N omp1
#$ -o omp1.out.$JOB_ID
#$ -e omp1.out.$JOB_ID
#$ -pe openmp 3

export OMP_NUM_THREADS=$NSLOTS
./hello_omp
```

In the example shown above we use define the number of CPU cores to be allocated to the job via "**#$ -pe openmp 3**" and we use $NSLOTS (which takes the value provided in "-pe openmp") to specify the number of OpenMP threads to be used.

```
##(omp2.sge)

#!/bin/bash
#$ -cwd
#$ -S /bin/bash
#$ -N omp1
#$ -o omp1.out.$JOB_ID
#$ -e omp1.out.$JOB_ID
#$ -pe openmp 3

./hello_omp
```

This second example assumes that the OMP_NUM_THREADS variable is defined externally. The following command can be used to submit the job:

```
[ivan@eimtarqso]$ qsub -v OMP_NUM_THREADS='3' h.sge
```

Questions:

2.1 How many OpenMP threads would you use in the UOC cluster? Why? Explain the drawbacks of using other options.

3. Data Parallelism

The example shown below shows the use of "parallel" and "for" clauses in two different but equivalent ways.

```
#pragma omp parallel
{
     #pragma omp for
     {
          for(i=0; i<N; i++){
               c[i] = b[i]+a[i];
          }
     }
}


##(we will target this second way moving forward)

#pragma omp parallel for
{
     for(i=0; i<N; i++){
          c[i] = b[i]+a[i];
     }
}
```

The example shown above performs the addition of two vectors (or arrays).

Questions:

3.1 How would you implement a program to compute the sum of the elemnts of an array "a" (i.e., sum=a[0]+…+a[N-1])?

3.2 How would you implement the program requested in 3.1 using a reduction clause?

4. Task (or functional) Parallelism

In this assignment you can also use task parallelism, if needed (it is not mandatory though). OpenMP supports tasks, which, as opposed to data parallelism where the same operation is applied to all data elements, allows different sections of the code to execute different operations on different data elements. An OpenMP task has a code to execute, a data environment (shared, private, reduction), and an assigned thread that executes the code and uses the data.

The following code illustrates the use of OpenMP sections (see more details in the materials provided at UOC campus).

```
#pragma omp parallel shared(n,a,b,c,d) private(i)
{
     #pragma omp sections nowait
          #pragma omp section
               for (i=0; i<n; i++)
                    d[i] = 1.0/c[i];
          #pragma omp section
               for (i=0; i<n-1; i++)
                    b[i] = (a[i] + a[i+1])/2;
     } /*-- End of sections --*/
} /*-- End of parallel region
```

Note that this code will execute in parallel two different loops (sequentially) over different arrays using an OpenMP thread for each of them.

5. Scheduling Policies and Performance Evaluation

This section of the assignment requests the parallelization of the program provided with the assignment ("mm_omp.c") using OpenMP.

Once the application is parallelized you are requested to provide a performance evaluation taking into account the following parameters:

- Matrix size (use at least 4 sizes)
- Number of threads (1..4, i.e., the number of physical cores in one node)
- OpenMP scheduling policies (i.e., static, dynamic, guided)

Specifically, it is requested to provide a study of scalability with respect to core count, matrix size and OpenMP scheduling policy. Please provide:

- The serial execution time (i.e., the execution time of the original program) for each matrix size used (i.e., 4 sizes)
- Run time for each parallel execution (i.e., 4 thread count x 4 matrix sizes x 3 scheduling policies)
- Speedup

Note: Please use the methodology that was discussed in the first assignment of the course (i.e., statistical studies with the plots associated to them)

Questions:

5.1 Provide you parallel version of your code. Please describe your main implementation decisions.

5.2 Provide the performance evaluation described above. Elaborate the results obtained.

6. MPI+OpenMP

This last section of the assignment targets the hybrid MPI+OpenMP programming model. As opposed to regular OpenMP, MPI+OpenMP allows you to take advantage of distributed memory at the same time that simplifies the programmability of data parallelism using shared memory (as seen in section 3).

It is provided a program ("barr_sync.c") that allows you to generate load imbalance across MPI processes using variables statically defined in the program (SIZE, ITERS, IMBALANCE, IMBALANCE_BASE). You can use these variables or customize your program to use input arguments from the command line as the previous case.

The goal of this assignment is three-fold: parallelizing the program using OpenMP together with MPI, studying the effect of load imbalance, and understanding a real-word use case of MPI asynchronous operations.

Questions:

6.1 Analyze the program and provide an MPI+OpenMP version of the same.

Note: this exercise is not trivial so it will be described further during our next teleconference call.

6.2 Provide a performance evaluation (e.g., based on execution time) with respect to load imbalance. You can modify the code to obtain load imbalance but if you so consider using MPI_Wtime.

Criteris d'avaluació

Es valorarà el correcte ús del model de programació OpenMP I MPI. També es tindrà en compte la correcta programació, estructura i funcionament dels programes, l'avaluació de rendiment 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: `VallesMarfanyPAC3.pdf (o *.tar)`

El format zip també és acceptat.

Si us plau NO utilitzem el format rar i assegureu-vos que els fitxer *.tar o *.zip es realitzen a partir d'un directori amb els continguts de la vostra entrega de la PAC.

Data de lliurament

Dissabte 5 de Desembre de 2015.