High Performance Computer
Rovira e Virgili-UOC

Student: Felix

1. **How many OpenMP threads you would use in the UOC cluster (hint: use lscpu command)? Why? Elaborate an inconvenient of using other configurations.**
   a. Socket = 1, core per socket = 4, threads per core = 1 this mean a single node could run = 1*4*1 = 4 threads Therefore I will use 4 threads.
   b. For Instance, having **less** thread than CPU's will not fully utilize the resources, and having **more** thread than CPU's could cause some sort of thread fighting for CPU resources basically we are serializing and not Parallelizing in an ideal system we will have one thread for core but only if each individual thread need 100% of the CPU.

2. **How you would implement a program that does the sum of the different elements stored in a vector "a" (sum=a[0]+…+a[N-1]) using the reduction clouse?**

***Attached sum of element in a vector**

```
// Felix F Feliu Question 2
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int main(){
       // variable declaration
       int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
       int sum, i;
       //initialize sum
       sum = 0;
       //add vector elements using a
       //parallel block implementing reduction clause
#pragma omp parallel for reduction(+:sum) //reduction clause

       for (i = 0; i <= 9; i++)    {
              sum +=a[i];   }
       //print sum value
       printf("Sum of vector a is %d.\n", sum);
       return 0;}
```

3. **Provide the parallel version of mm.c and mm2.c codes using the different scheduling policies. Please describe your main implementation decisions.**

*See attached file mmparallel.c and mm2parallel.c

Main implementation decisions strategy

- No to make big change only enough to see the implementation working
- Find the best area to improve the parallelization. I found one area:
  1. Parallelize the for loop in order to use Scheduling
- Run the programs for different scheduling. I started in the next order static first, dynamic second, and guided third
- Increase the chunksize for each schedule, I used 1,2,3,4 and take the time for each one
- Make tables and plot the results

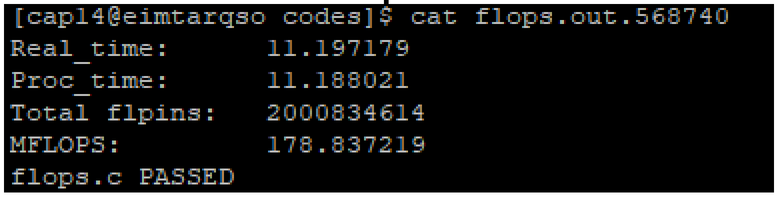
4. Provide the performance evaluation described above. Elaborate the results obtained.
***Excel file Time analysis attached for this answer**

Observations:

- mm2 is more time consuming than mm
- It is noticed a time increase of 16% in static, 29 % in dynamic and 24 % in guided when is compared mm to mm2.
- For sequential we can see a great improvement from mm sequential to parallel but not too much in mm2 to parallel basically we do not have any improvement at all.
- In average guided work faster for mm and static work faster for mm2 when the codes are parallelized from sequential.
- guided has more time consuming special for greater matrix size
- Static scheduling with better performance than dynamic and guided been dynamic behind then.

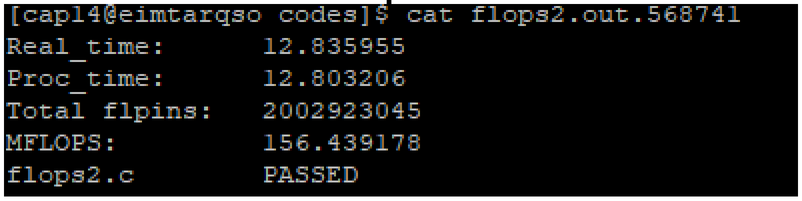
5. **Why do you think it is this difference in execution time and MFLOPs?**

Flops.c



```
[capl4@eimtarqso codes]$ cat flops.out.568740
Real_time:       11.197179
Proc_time:       11.188021
Total flpins:    2000834614
MFLOPS:          178.837219
flops.c PASSED
```

Flops2.c



```
[capl4@eimtarqso codes]$ cat flops2.out.568741
Real_time:       12.835955
Proc_time:       12.803206
Total flpins:    2002923045
MFLOPS:          156.439178
flops2.c         PASSED
```

- If we look to the **for loop** the outer and inner indexes have been exchanged the way in what the indexes have been changed from flops.c to flops2.c result in more instruction per second achieved, therefore we have increase in time due to the increase in instruction
- Interesting to observe the decrease in mflops in flops2 code indication that the programs do not use the same amount of floating point flops2 is using less floating points calculations.

6. **What hardware counters would you use to study the differences between the**

two implementations? Why?
PAPI_TOT_INS: Instructions completed,
PAPI_TOT_IIS: Instructions issued,
PAPI_LD_INS: load instruction
To see the difference between instructions issued and completed and how many are load instructions

Floating Point Operation

PAPI_FP_OPS : Floating point operations
PAPI_FP_INS: Floating point instructions
PAPI_SR_INS: store instruction
To see difference between floating point instruction and executed and how many are stored

7. **Provide the code where you use hardware counters to quantify the differences between the two examples provided. Provide the results obtained.**
***See attached file counters.c and counters2.c**

Sequential Version

***I was not able to run all at once after many attempts, I did it two by two**

counters		counters2	
PAPI_TOT_INS	**33073987856**	PAPI_TOT_INS	**33073992706**
PAPI_TOT_IIS	**33155717391**	PAPI_TOT_IIS	**33494928971**
PAPI_LD_INS	**13026014876**	PAPI_LD_INS	**13026020968**
PAPI_FP_OPS	**2003106424**	PAPI_FP_OPS	**2008824355**
PAPI_FP_INS	**2002259493**	PAPI_FP_INS	**2008184833**
PAPI_SR_INS	**2015013741**	PAPI_SR_INS	**2015019839**

8.**Provide parallel versions (OpenMP) and the results obtained using PAPI for the two examples provided.**
***See attached file countersparallel.c   counter2parallel.c**

Parallel Version

***I was not able to run all at once after many attempts, I did it two by two**

countersparallel		counters2parallel	
PAPI_TOT_INS	**33073987523**	PAPI_TOT_INS	**33073989045**
PAPI_TOT_IIS	**33150820026**	PAPI_TOT_IIS	**33375667064**
PAPI_LD_INS	**13026015358**	PAPI_LD_INS	**13026020451**
PAPI_FP_OPS	**2003041735**	PAPI_FP_OPS	**2005664366**
PAPI_FP_INS	**2002198709**	PAPI_FP_INS	**2005485182**
PAPI_SR_INS	**2015014226**	PAPI_SR_INS	**2015019321**

We can see a small improvement for which we can gain some time, there are some movements going down if we see to instructions completed, instruction issued, floating point operations, floating points instructions all they moved down consequently we are gaining overall a small time reduction .

9. Provide a parallel implementation of p2.c using OpenMP.

***Attached p2parallel.c**