CO3265 Parallel Programming

Semester-6, 2024

Laboratory session

OpenMP

Instructions:

- You are required to do each step as instructed bellow.
- You are required to write a report and submit within a week from your practical session. In your report, each problem should be addressed.
- You are advised to note any outputs and take trace files with you when you leave the laboratory to recollect the activities later and to prepare the report.
- Marks (100%): from the report and lab evaluation (if any).
- Time: 3 hours.

---

1. Copy the following code and compile.
   a. What is done in the below code? Explain,
   b. Write down the time consumed by the process.

```
#include <omp.h>
#include <stdio.h>
#define ARR_SIZE 1000000000
static int a[ARR_SIZE];
int main(int *argc, char *argv[])
{
        int i, sum = 0;

        double t1, t2;

        /*Initialize the array*/
        for(i =0; i<ARR_SIZE; i++){
                a[i] = 1;
        }
        t1 = omp_get_wtime();

        //sum up the array

                        for(i=0;i<ARR_SIZE;i++)
                                sum+=a[i];

                t2 = omp_get_wtime();

        printf("Time taken : %g for sum : %d \n", t2-t1, sum);
        return 0;
}
```

2. Modify the code as below and recompile it. Execute the code now and see the time taken for the process. Compare the new time with the previous time. If the programme consumes much time than the previous code, explain the reason for it.

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

#define ARR_SIZE 1000000000
static int a[ARR_SIZE];

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

        int i, sum = 0;
        int tid, numt = 0;
        double t1, t2;

        /*Initialize the array*/
        for(i =0; i<ARR_SIZE; i++)
                a[i] = 1;



        t1 = omp_get_wtime();

        //sum up the array
        #pragma omp parallel default(shared) private( i, tid )
        {
                tid = omp_get_thread_num();
                numt = omp_get_num_threads();


                for(i=0;i<ARR_SIZE;i++)
                        sum+=a[i];


        }
        t2 = omp_get_wtime();
        printf("Number of threads : %d\n", numt);
        printf("Time taken : %g for sum : %d \n", t2-t1, sum);
        return 0;
}
```

3. Modify the code to manually distribute the workload to different threads. Attach your code.
4. Modify the code to distribute the workload into multiple threads by using OpenMP libraries. Attach your codes.