CS3523:OS-2
Programming assignment-2:Sudoku Validator

CS21BTECH11055- ABHINAY SADINENI

February 3, 2023

1 Code design

To validate sudoku we need to checks:

1. all rows
2. all cloumns
3. all subgrids of size $\sqrt{N} \times \sqrt{N}$ where N is size of sudoku

it takes almost same time to validate whether it is a row,cloumn or subgrid.it is best to distrbute them among all threads to balance the load.

2 Implemenation

1. distribution of load same for both pthreads and openmp. evenly divided among all threads

```
for (int i = 1; i <=K; i++)
    {
        params*p=malloc(sizeof(params));
       if(i<=(3*N)%K){
            p->start=start;
            p->size=(3*N)/K+1;
            start=start+p->size;
       }
       else
       {
            p->start=start;
            p->size=(3*N)/K;
            start=start+p->size;
       }
        pthread_create(&tid[i-1],NULL,thread_work,(void*)p);
    }
```

2. threads work to validate its given load and store the result of each validation in the valid array

row1	row2	.....	col1	col2	.....	subgrid1	subgrid2	.....

↑
index i inside thread work, N rows,N columns,N subgrids

```
void  thread_work(params* p){

    for (int i = p->start; i < p->start+p->size; i++)
    {
        if(i<N){
            valid[i]=row_check(i);
        }
        else if(N<=i && i<2*N)
        {
              valid[i]=col_check(i-N);
        }
        else
        {
            valid[i]=grid_check(i-2*N);
        }

    }
}
```

3. function to validate a row

```
int row_check(int j){
    bool*check=calloc(N,sizeof(bool));

    for (int i = 0; i < N; i++)
    {
        int temp=mat[j][i]-1;
      if(check[temp]==false){
        check[temp]=true;
      }
      else
      {
        free(check);
        return 0;
      }
     }
     free(check);
    return 1;


}
```

4. function to validate a column

```
int col_check(int j){
    bool*check=calloc(N,sizeof(bool));
    for (int i = 0; i < N; i++)
    {
       int temp=mat[i][j]-1;
       if(check[temp]==false){
        check[temp]=true;
      }
      else
      {
        free(check);
        return 0;
      }
     }
     free(check);
    return 1;
}
```

5. function to validate a subgrid

```
int grid_check(int j){
    int sp=sqrt((double)N);
    bool*check=calloc(N,sizeof(bool));
     for (int i = (j%sp)*sp; i <(j%sp)*sp+sp; i++)
     {
        for (int k = (j/sp)*sp; k <(j/sp)*sp+sp; k++)
        {
           int temp=mat[i][k]-1;
           if(check[temp]==false){
        check[temp]=true;
      }
      else
      {
        free(check);
        return 0;
      }

        }

     }
     free(check);
    return 1;
}
```

In all the three functions we check whether all 1 to N numbers are present or not if there is a repetition of number then function immediately returns false.

6. This below code block searches all the valid array for false and declares the whether the sudoku is valid or not

```
int check=1;
    for (int i = 0; i < 3*N; i++)
    {
       if (valid[i]==0)
       {
         check=0;
         break;
       }
    }
```

The openmp code is done in the same way except for some small changes like replacing pthread_ create and pthread_join with omp library syntax

3 Performance Analysis

3.1 Graph: Time Taken v/s N

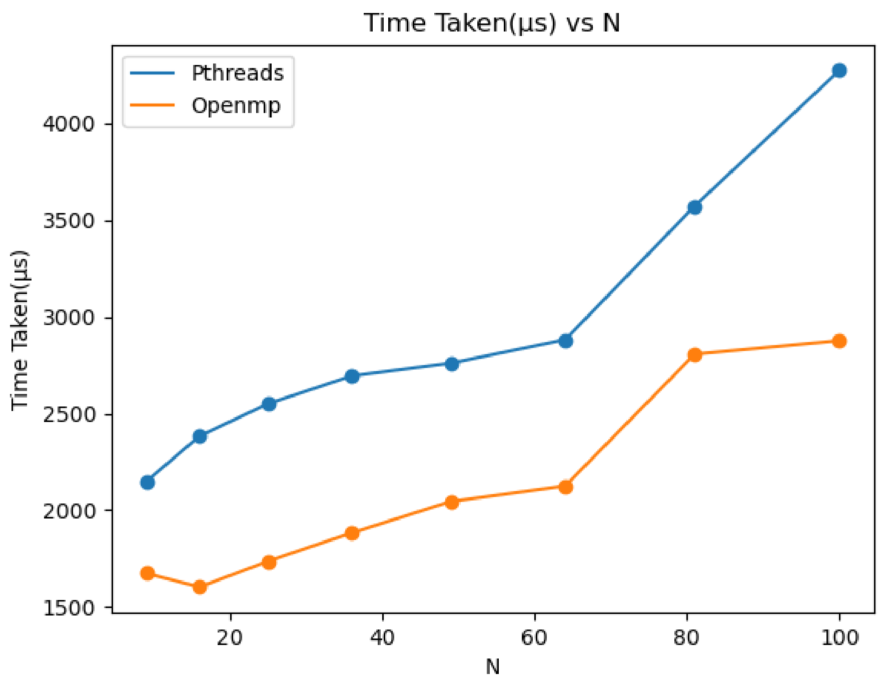

Figure 1: T v/s N(constant K=16)

Observations:

1. On observing carefully their is regular trend for both pthreads and openmp as the size of the sudoku the number of validations per each thread increases so therefore time taken also increases

2. There is slight defect of trend in openmp i.e. slight decrease at N=4 this maybe because of some CPU scheduling at the time of excecution.

3. Openmp's performance better than pthreads this is because openmp optimizes sequentail code blocks

3.2 Graph: Time Taken v/s No of threads

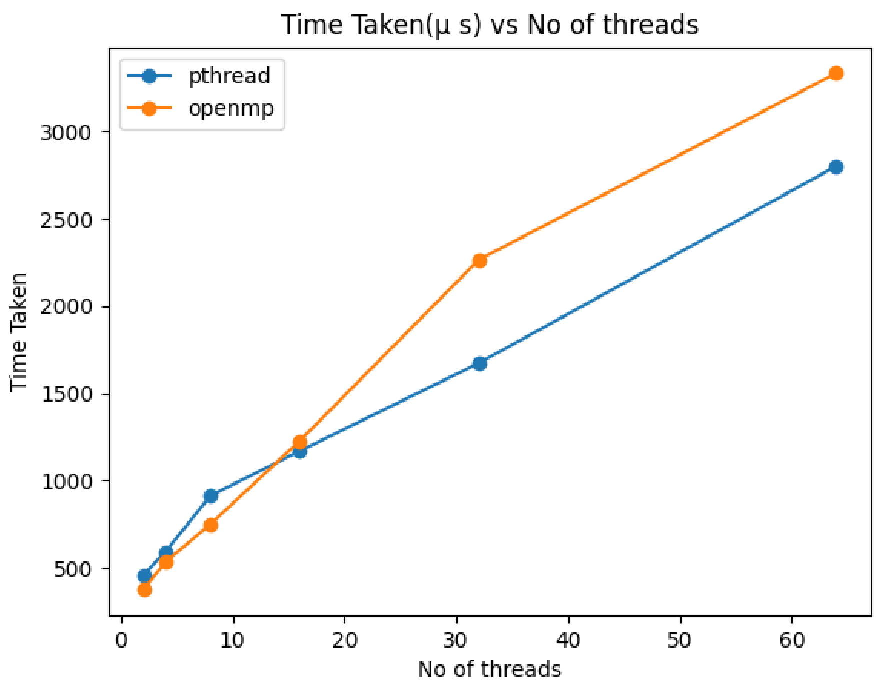

Figure 2: T v/s K (N=25 constant)

Observations:

1. We expect the time taken to decrease
2. But here the value of N is just 25 ,in which case the overhead of creation of threads exceeds the speed up by threads.
3. Usually threads speedup is significent when there is large amount of load.
4. for checking just 625 cells in a sudoku threads are not needed.
5. This given input works faster with normal iteration method.
6. CPU(intel i5-1035G1) has only 4 cores thats why at N=16 the openmp plot crosses the pthread plot.

3.3 Conculsion:

1. Openmp implementation works better than that of pthreads
2. But we cannot conclude with that one fact, since the omp threads also depend on number of cores avialable and architecture of the chip more than pthreads as seen in second graph