

Final Year B. Tech., Sem VI 2021-22

High Performance Computing Lab

Assignment submission

PRN No: 2019BTECS00070

Full name: Prathmesh Basaweshwar Killedar

Batch: B3

Assignment: 3

Q1. Analyse and implement a Parallel code for below program using OpenMP.

Ans:-

Code:-

```
// C Program to find the minimum scalar product of two vectors (dot product)
#include<bits/stdc++.h>
#include <omp.h>

using namespace std;

int sort(int arr[], int n)
{
    int i, j;
    #pragma omp parallel shared(arr) private(j)
    #pragma omp for schedule(dynamic)
    for (i = 0; i < n-1; i++)
        for (j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1])
            {
                int temp = arr[j];
```

```

        arr[j] = arr[j+1];
        arr[j+1] = temp;
    }
}

int sort_des(int arr[], int n)
{
    int i,j;
    #pragma omp parallel shared(arr) private(j)
    #pragma omp for schedule(dynamic)
    for (i = 0; i < n; ++i)
    {
        for (j = i + 1; j < n; ++j)
        {
            if (arr[i] < arr[j])
            {
                int a = arr[i];
                arr[i] = arr[j];
                arr[j] = a;
            }
        }
    }
}

```

```

int main()
{
    //fill the code;
    int i,tid,n,psum;
    int threads = 4;
    cout<<"Enter Size of Array: ";
    cin>>n;
    int arr1[n], arr2[n];
    cout<<"Enter Elements of First Array:\n";
    for(i = 0; i < n ; i++)
    {
        cin>>arr1[i];
    }
}

```

```

    cout<<"Enter Elements of Second Array:\n";
    for(i = 0; i < n ; i++)
    {
        cin>>arr2[i];
    }
    sort(arr1, n);
    for(auto it:arr1)cout<<it<<" ";
    cout<<endl;
    sort_des(arr2, n);
    for(auto it:arr2)cout<<it<<" ";
    cout<<endl;
    int sum = 0;
    #pragma omp parallel private(i,tid,psum) num_threads(threads)
    {
        psum=0;
        tid = omp_get_thread_num();
        #pragma omp for reduction(+:sum) schedule(static)
        for(int i=0; i<n; i++)
        {
            sum += arr1[i] * arr2[i];
            psum+=sum;
        }
        printf("Thread %d partial sum = %d\n",tid,psum);
    }
    cout<<"Sum: "<<sum<<endl;

    return 0;
}

```

Output:-

```
C:\Users\DELL\Desktop\sem7\hpc assignment\complete\Assignment3\openmpq1.exe
Enter Size of Array: 4
Enter Elements of First Array:
1 2 3 4
Enter Elements of Second Array:
5 6 7 8
1 2 3 4
8 7 6 5
Thread 3 partial sum = 20
Thread 0 partial sum = 8
Thread 1 partial sum = 14
Thread 2 partial sum = 18
Sum: 60

-----
Process exited after 25.51 seconds with return value 0
Press any key to continue . . .
```

Q2. Write OpenMP code for two 2D Matrix addition, vary the size of your matrices from 250, 500, 750, 1000, and 2000 and measure the runtime with one thread (Use functions in C to calculate the execution time or use GPROF)

- i. For each matrix size, change the number of threads from 2,4,8., and plot the speedup versus the number of threads.**
- ii. Explain whether or not the scaling behaviour is as expected**

Code:-

```
#include <bits/stdc++.h>
#include <omp.h>

using namespace std;

int main()
{
    int tid, nthreads , i, j;
    int n=100;
    while(1){
        if(n==500)
            break;
        else
            n+=100;
        nthreads=4;
        int a[n][n], b[n][n], c[n][n];

        int index = 0;

        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
```

```

        {
            a[i][j] = b[i][j] = (i+j);
        }
    }

    printf("Time Required to do Matrix Multiplication of size
%d\nUsing Threads: %d",n,nthreads);

    double time = omp_get_wtime();

    #pragma omp parallel shared(a, b, c, nthreads) private(tid, i,
j) num_threads(nthreads)
    {
        # pragma omp parallel for
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                c[i][j] = a[i][j] + b[i][j];
            }
        }
    }

    printf("\nDone In %f Seconds\n\n", omp_get_wtime() -
time);

    }
    return 0;
}

```

Output:-

```
C:\Users\DELL\Desktop\sem7\hpc assignment\assignment3\openmpq2.exe
Time Required to do Matrix Multiplication of size 200
Using Threads: 4
Done In 0.001000 Seconds

Time Required to do Matrix Multiplication of size 300
Using Threads: 4
Done In 0.001000 Seconds

Time Required to do Matrix Multiplication of size 400
Using Threads: 4
Done In 0.001000 Seconds

-----
Process exited after 0.9296 seconds with return value 3221225725
Press any key to continue . . .
```

```
C:\Users\DELL\Desktop\sem7\hpc assignment\assignment3\openmpq2.exe
Time Required to do Matrix Multiplication of size 200
Using Threads: 2
Done In 0.000000 Seconds

Time Required to do Matrix Multiplication of size 300
Using Threads: 2
Done In 0.000000 Seconds

Time Required to do Matrix Multiplication of size 400
Using Threads: 2
Done In 0.000000 Seconds

-----
Process exited after 0.5671 seconds with return value 3221225725
Press any key to continue . . .
```

Q3. For 1D Vector (size=200) and scalar addition, Write a OpenMP code with the following:

- i. Use STATIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup.
- ii. Use DYNAMIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup.
- iii. Demonstrate the use of nowait clause

Ans:

Use of Static Schedule

Code:

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

int main(){
    int n = 200, i, j=99;

    int arr1[n], answer[n];

    for(i = 0; i < n; i++){
        arr1[i] = rand()%100;
    }

    double time = omp_get_wtime();
    #pragma omp parallel for schedule(static,20) shared(arr1, answer,j)
    private(i)

    for(i = 0; i < n; i++)
    {
        answer[i] = arr1[i] + j;
    }
```



```

printf("\nDone In %f Seconds\n\n", omp_get_wtime() - time);

printf("\nArray 1: \n");
for(i = 0; i < n; i++){
    printf("\t %d", arr1[i]);
}

printf("\nAnswer: \n");
for(i = 0; i < n; i++){
    printf("\t %d", answer[i]);
}
return 0;
}

```

Output:

```

C:\Users\DELL\Desktop\sem7\hpc assignment\assignment3\openmpq31.exe
Done In 0.001000 Seconds

Array 1:
41      67      34      0      69      24      78      58      62      64      5      45      81      27
61      91      95      42      27      36      91      4      2      53      92      82      21      16      18
95      47      26      71      38      69      12      67      99      35      94      3      11      22      33
73      64      41      11      53      68      47      44      62      57      37      59      23      41      29
78      16      35      90      42      88      6      40      42      64      48      46      5      90      29
70      50      6      1      93      48      29      23      84      54      56      40      66      76      31
8      44      39      26      23      37      38      18      82      29      41      33      15      39      58
4      30      77      6      73      86      21      45      24      72      70      29      77      73      97
12      86      90      61      36      55      67      55      74      31      52      50      50      41      24
66      30      7      91      7      37      57      87      53      83      45      9      9      58      21
88      22      46      6      30      13      68      0      91      62      55      10      59      24      37
48      83      95      41      2      50      91      36      74      20      96      21      48      99      68
84      81      34      53      99      18      38      0      88      27      67      28      93      48      83
7      21      10      17      13      14

Answer:
140      166      133      99      168      123      177      157      161      163      104      144      180      126
160      190      194      141      126      135      190      103      101      152      191      181      120      115      117
194      146      125      170      137      168      111      166      198      134      193      102      110      121      132
172      163      140      110      152      167      146      143      161      156      136      158      122      140      128
177      115      134      189      141      187      105      139      141      163      147      145      104      189      128
169      149      105      100      192      147      128      122      183      153      155      139      165      175      130
107      143      138      125      122      136      137      117      181      128      140      132      114      138      157
103      129      176      105      172      185      120      144      123      171      169      128      176      172      196
111      185      189      160      135      154      166      154      173      130      151      149      149      140      123
165      129      106      190      106      136      156      186      152      182      144      108      108      157      120
187      121      145      105      129      112      167      99      190      161      154      109      158      123      136
147      182      194      140      101      149      190      135      173      119      195      120      147      198      167
183      180      133      152      198      117      137      99      187      126      166      127      192      147      182
106      120      109      116      112      113

-----
Process exited after 0.05321 seconds with return value 0
Press any key to continue . . .

```

Use of Dynamic Schedule

Code:

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

int main(){
    int n = 200, i ,j=99;

    int arr1[n], answer[n];

    for(i = 0; i < n; i++)
    {
        arr1[i] = rand()%100;
    }

    double time = omp_get_wtime();

    #pragma omp parallel for schedule(dynamic,20) shared(arr1,
answer,j) private(i)
    for(i = 0; i < n; i++)
    {
        answer[i] = arr1[i] + j;
    }

    printf("\nDone In %f Seconds\n\n", omp_get_wtime() - time);

    printf("\nArray 1: \n");
    for(i = 0; i < n; i++)
    {
        printf("\t %d", arr1[i]);
    }
```

```

        printf("\nAnswer: \n");
    for(i = 0; i < n; i++)
    {
        printf("\t %d", answer[i]);
    }
    return 0;
}

```

Output:

```

C:\Users\DELL\Desktop\sem7\hpc assignment\assignment3\openmpq32.exe
Done In 0.001000 Seconds

Array 1:
41      67      34      0      69      24      78      58      62      64      5      45      81      27
61      91      95      42      27      36      91      4      2      53      92      82      21      16      18
95      47      26      71      38      69      12      67      99      35      94      3      11      22      33
73      64      41      11      53      68      47      44      62      57      37      59      23      41      29
78      16      35      90      42      88      6      40      42      64      48      46      5      90      29
70      50      6      1      93      48      29      23      84      54      56      40      66      76      31
8       44      39      26      23      37      38      18      82      29      41      33      15      39      58
4       30      77      6      73      86      21      45      24      72      70      29      77      73      97
12      86      90      61      36      55      67      55      74      31      52      50      50      41      24
66      30      7      91      7      37      57      87      53      83      45      9      9      58      21
88      22      46      6      30      13      68      0      91      62      55      10      59      24      37
48      83      95      41      2      50      91      36      74      20      96      21      48      99      68
84      81      34      53      99      18      38      0      88      27      67      28      93      48      83
7       21      10      17      13      14

Answer:
140      166      133      99      168      123      177      157      161      163      104      144      180      126
160      190      194      141      126      135      190      103      101      152      191      181      120      115      117
194      146      125      170      137      168      111      166      198      134      193      102      110      121      132
172      163      140      110      152      167      146      143      161      156      136      158      122      140      128
177      115      134      189      141      187      105      139      141      163      147      145      104      189      128
169      149      105      100      192      147      128      122      183      153      155      139      165      175      130
107      143      138      125      122      136      137      117      181      128      140      132      114      138      157
103      129      176      105      172      185      120      144      123      171      169      128      176      172      196
111      185      189      160      135      154      166      154      173      130      151      149      149      140      123
165      129      106      190      106      136      156      186      152      182      144      108      108      157      120
187      121      145      105      129      112      167      99      190      161      154      109      158      123      136
147      182      194      140      101      149      190      135      173      119      195      120      147      198      167
183      180      133      152      198      117      137      99      187      126      166      127      192      147      182
106      120      109      116      112      113

-----
Process exited after 0.04895 seconds with return value 0
Press any key to continue . . .

```

Use of Nowait Clause

Code:

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

int main()
{
    int n = 200, i ,j=99;

    int arr1[n], answer[n];

    for(i = 0; i < n; i++)
    {
        arr1[i] = rand()%100;
    }

    double time = omp_get_wtime();

    #pragma omp parallel
    {
        #pragma omp for nowait
        for(i = 0; i < n; i++)
        {
            answer[i] = arr1[i] + j;
        }
    }

    printf("\nDone In %f Seconds\n\n", omp_get_wtime() - time);

    printf("\nArray 1: \n");
    for(i = 0; i < n; i++){
        printf("\t %d", arr1[i]);
    }

    printf("\nAnswer: \n");
```

```

    for(i = 0; i < n; i++){
        printf("\t %d", answer[i]);
    }
    return 0;
}

```

Output:

```

C:\Users\DELL\Desktop\sem7\hpc assignment\assignment3\openmp33.exe
Done In 0.001000 Seconds

Array 1:
41      67      34      0      69      24      78      58      62      64      5      45      81      27
61      91      95      42      27      36      91      4      2      53      92      82      21      16      18
95      47      26      71      38      69      12      67      99      35      94      3      11      22      33
73      64      41      11      53      68      47      44      62      57      37      59      23      41      29
78      16      35      90      42      88      6      40      42      64      48      46      5      90      29
70      50      6      1      93      48      29      23      84      54      56      40      66      76      31
8       44      39      26      23      37      38      18      82      29      41      33      15      39      58
4       30      77      6      73      86      21      45      24      72      70      29      77      73      97
12      86      90      61      36      55      67      55      74      31      52      50      50      41      24
66      30      7      91      7      37      57      87      53      83      45      9      9      58      21
88      22      46      6      30      13      68      0      91      62      55      10      59      24      37
48      83      95      41      2      50      91      36      74      20      96      21      48      99      68
84      81      34      53      99      18      38      0      88      27      67      28      93      48      83
7       21      10      17      13      14

Answer:
140     166     133     99      168     123     177     157     161     163     104     144     180     126
160     190     194     141     126     135     190     103     101     152     191     181     120     117
194     146     125     170     137     168     111     166     198     134     193     102     110     132
172     163     140     110     152     167     146     143     161     156     136     158     122     140     128
177     115     134     189     141     187     105     139     141     163     147     145     104     189     128
169     149     105     100     192     147     128     122     183     153     155     139     165     175     130
107     143     138     125     122     136     137     117     181     128     140     132     114     138     157
103     129     176     105     172     185     120     144     123     171     169     128     176     172     196
111     185     189     160     135     154     166     154     173     130     151     149     149     140     123
165     129     106     190     106     136     156     186     152     182     144     108     108     157     120
187     121     145     105     129     112     167     99      190     161     154     109     158     123     136
147     182     194     140     101     149     190     135     173     119     195     120     147     198     167
183     180     133     152     198     117     137     99      187     126     166     127     192     147     182
106     120     109     116     112     113

-----
Process exited after 0.05625 seconds with return value 0
Press any key to continue . . .

```