Class: Final Year (Computer Science and Engineering)

Year: 2022-23 **Semester:** 1

Course: High Performance Computing Lab

Practical No. 3

Exam Seat No: 2019BTECS00064

Name – Kunal Santosh Kadam

Title of practical: Study and Implementation of schedule, nowait, reduction, ordered and collapse clauses

Problem Statement 1:

Analyse and implement a Parallel code for below program using openMP

Department of Computer Science and Engineering

```
arr[j+1] = temp;
                      }
}
int sort_des(int arr[], int n)
{
       int i,j;
       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 n;
       scanf("%d",&n);
       int arr1[n], arr2[n];
       int i;
       for(i = 0; i < n; i++)
              scanf("%d",&arr1[i]);
       for(i = 0; i < n; i++)
```

Department of Computer Science and Engineering

```
scanf("%d",&arr2[i]);
}
sort(arr1, n);
sort_des(arr2, n);
int sum = 0;
for(i = 0; i < n; i++)
{
    sum = sum + (arr1[i] * arr2[i]);
}
printf("%d",sum);
return 0;
}
```

Screenshot #:

Information #:

```
// 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])</pre>
```

Department of Computer Science and Engineering

```
{
                          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);
      sort_des(arr2, n);
      int sum = 0;
      #pragma omp parallel private(i,tid,psum) num_threads(threads)
      {
             psum=0;
             tid = omp_get_thread_num();
```

Problem Statement 2:

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 in 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.

Screenshot #:

```
Time Required to do Matrix Multiplication of size 200
Using Threads: 2
Done In 0.002000 Seconds

Time Required to do Matrix Multiplication of size 300
Using Threads: 2
Done In 0.001000 Seconds

Time Required to do Matrix Multiplication of size 300
Using Threads: 2
Done In 0.001000 Seconds

Time Required to do Matrix Multiplication of size 400
Using Threads: 2
Done In 0.005000 Seconds

Process exited after 6.986 seconds with return value 3221225725
Press any key to continue . . . _
```

7 | Page

```
Time Required to do Matrix Multiplication of size 200
Using Threads: 4
Done In 0.003000 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 300
Using Threads: 4
Done In 0.001000 Seconds

Time Required to do Matrix Multiplication of size 400
Using Threads: 4
Done In 0.004000 Seconds

Process exited after 6.82 seconds with return value 3221225725
Press any key to continue . . . _
```

Information #:

```
#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;
}
```

Problem Statement 3:

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

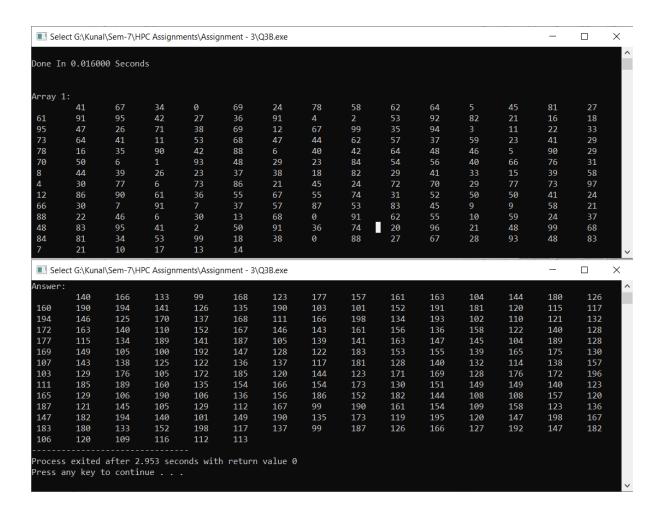
- i. Use STATIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup.
- ii. Use DYNAMIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup.
- iii. Demonstrate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

Screenshot #:

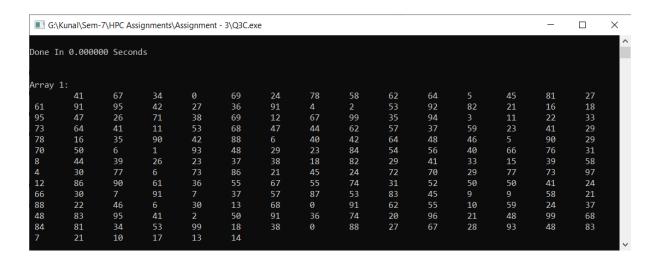
Use of Static Schedule

ne Ir	n 0.0150	00 Secon	ds												
ray :	1:														
	41	67	34	0	69	24	78	58	62	64	5	45	81	27	
1	91	95	42	27	36	91	4	2	53	92	82	21	16	18	
5	47	26	71	38	69	12	67	99	35	94	3	11	22	33	
3	64	41	11	53	68	47	44	62	57	37	59	23	41	29	
В	16	35	90	42	88	6	40	42	64	48	46	5	90	29	
9	50	6	1	93	48	29	23	84	54	56	40	66	76	31	
	44	39	26	23	37	38	18	82	29	41	33	15	39	58	
	30	77	6	73	86	21	45	24	72	70	29	77	73	97	
2	86	90	61	36	55	67	55	74	31	52	50	50	41	24	
	30	7 46	91	7	37	57	87	53	83	45	9	9	58	21	
5	22		6	30	13	68 91	0 36	91 74	62 20	55 96	10 21	59 48	24 99	37 68	
3	22		4.4					/4	20				99	80	
B B	83	95	41	2	50			00	27	67	20	0.7	40	0.7	
3	83 81 21	95 34 10	53 17	99 13	18 14	38	0	88	27	67	28	93	48	83	
3	83 81 21	95 34 10	53 17	99 13	18	38		88	27	67	28	93	48	83	2000
3 1 G:\k	83 81 21 Kunal\Sem-	95 34 10 -7\HPC Ass	53 17 ignments\A	99 13 Assignment	18 14 - 3\Q3A.ex	38 Ke	0						_		2000
3 3 1 G:\k	83 81 21 Kunal\Sem- : 140	95 34 10 7\HPC Ass	53 17 ignments\A 133	99 13 Assignment 99	18 14 - 3\Q3A.ex	38 (e 123	177	157	161	163	104	144	180	126	
3 3 1 G:\k	83 81 21 Kunal\Sem- : 140 190	95 34 10 7\HPC Ass 166 194	53 17 ignments\A 133 141	99 13 Assignment 99 126	18 14 - 3\Q3A.ex 168 135	38 ce 123 190	0 177 103	157 101	161 152	163 191	104 181	144 120	180 115	126 117	70
8 8 4 9 94	83 81 21 Kunal\Sem- : 140 190 146	95 34 10 7\HPC Ass 166 194 125	53 17 ignments\A 133 141 170	99 13 Assignment 99 126 137	18 14 - 3\Q3A.ex 168 135 168	38 ce 123 190 111	177 103 166	157 101 198	161 152 134	163 191 193	104 181 102	144 120 110	180 115 121	126 117 132	
G:\k	83 81 21 Kunal\Sem- : : 140 190 146 163	95 34 10 7\HPC Ass 166 194 125 140	53 17 ignments\A 133 141 170 110	99 13 Assignment 99 126 137 152	18 14 - 3\Q3A.ex 168 135 168 167	123 190 111 146	177 103 166 143	157 101 198 161	161 152 134 156	163 191 193 136	104 181 102 158	144 120 110 122	180 115 121 140	126 117 132 128	80
G:\k G:\k swer: 50 94 72	83 81 21 Kunal\Sem- : : 140 190 146 163 115	95 34 10 7\HPC Ass 166 194 125 140 134	53 17 ignments\A 133 141 170 110 189	99 13 Assignment 99 126 137 152 141	18 14 - 3\Q3A.ex 168 135 168 167 187	123 190 111 146 105	177 103 166 143 139	157 101 198 161 141	161 152 134 156 163	163 191 193 136 147	104 181 102 158 145	144 120 110 122 104	180 115 121 140 189	126 117 132 128 128	33
G:\k G:\k 50 94 72 77	83 81 21 Cunal\Sem- : 140 190 146 163 115 149	95 34 10 7\HPC Ass 166 194 125 140 134 105	53 17 ignments\A 133 141 170 110 189 100	99 13 Assignment 99 126 137 152 141 192	18 14 - 3\Q3A.ex 168 135 168 167 187 147	123 190 111 146 105 128	177 103 166 143 139 122	157 101 198 161 141 183	161 152 134 156 163 153	163 191 193 136 147 155	104 181 102 158 145 139	144 120 110 122 104 165	180 115 121 140 189 175	126 117 132 128 128 130	
G:\k G:\k 50 94 72 77 59	83 81 21 Kunal\Sem- : : 140 190 146 163 115	95 34 10 7\HPC Ass 166 194 125 140 134	53 17 ignments\A 133 141 170 110 189	99 13 Assignment 99 126 137 152 141	18 14 - 3\Q3A.ex 168 135 168 167 187 147 136	123 190 111 146 105	177 103 166 143 139	157 101 198 161 141	161 152 134 156 163 153 128	163 191 193 136 147	104 181 102 158 145	144 120 110 122 104	180 115 121 140 189 175 138	126 117 132 128 128 130 157	80
G:\k G:\k Swer: 60 94 72 77 69 87 83	83 81 21 Kunal\Sem- : : 140 190 146 163 115 149 143	95 34 10 7\HPC Ass 166 194 125 140 134 105 138	53 17 133 141 170 110 189 100 125	99 13 99 126 137 152 141 192 122	18 14 - 3\Q3A.ex 168 135 168 167 187 147	123 190 111 146 105 128 137	177 103 166 143 139 122 117	157 101 198 161 141 183 181	161 152 134 156 163 153	163 191 193 136 147 155 140	104 181 102 158 145 139 132	144 120 110 122 104 165 114	180 115 121 140 189 175	126 117 132 128 128 130	
G:\kwer 60 67 67 69 67	83 81 21 Xunal\Sem- : 140 190 146 163 115 149 143 129	95 34 10 -7\HPC Ass 166 194 125 140 134 105 138 176 189	53 17 17 133 141 170 110 189 100 125 105	99 13 99 126 137 152 141 192 122 172 135	18 14 - 3\Q3A.ex 168 135 168 167 187 147 136 185 154	123 190 111 146 105 128 137 120 166	177 103 166 143 139 122 117 144 154	157 101 198 161 141 183 181 123 173	161 152 134 156 163 153 128 171 130	163 191 193 136 147 155 140 169	104 181 102 158 145 139 132 128 149	144 120 110 122 104 165 114 176 149	180 115 121 140 189 175 138 172 140	126 117 132 128 128 130 157 196 123	
G:\\Mercolonian G:\Mercolonian G:\Merc	83 81 21 Kunal\Sem-: : 140 190 146 163 115 149 143 129 185	95 34 10 -7\HPC Ass 166 194 125 140 134 105 138 176	53 17 ignments\A 133 141 170 110 189 100 125 105 160	99 13 99 126 137 152 141 192 122 172	18 14 - 3\Q3A.ex 168 135 168 167 187 147 136 185	123 190 111 146 105 128 137 120	177 103 166 143 139 122 117 144	157 101 198 161 141 183 181 123	161 152 134 156 163 153 128 171	163 191 193 136 147 155 140 169	104 181 102 158 145 139 132 128	144 120 110 122 104 165 114 176	180 115 121 140 189 175 138 172	126 117 132 128 128 130 157	80
G:\ki	83 81 21 Kunal\Sem- : : 149 190 146 163 115 149 143 129 185 129	95 34 10 -7\HPC Ass 166 194 125 140 134 105 138 176 189 106	53 17 ignments\A 133 141 170 110 189 100 125 105 160 190	99 13 4ssignment 99 126 137 152 141 192 122 172 135 106	18 14 -3\Q3A.ex 168 135 168 167 187 147 136 185 154 136	123 190 111 146 105 128 137 120 166 156	177 103 166 143 139 122 117 144 154	157 101 198 161 141 183 181 123 173 152	161 152 134 156 163 153 128 171 130 182	163 191 193 136 147 155 140 169 151 144	104 181 102 158 145 139 132 128 149	144 120 110 122 104 165 114 176 149 108	180 115 121 140 189 175 138 172 140 157	126 117 132 128 128 130 157 196 123 120	
8 8 4	83 81 21 Sunal\Sem- : 140 190 146 163 115 149 143 129 185 129	95 34 10 -7\HPC Ass 166 194 125 140 134 105 138 176 189 106 145	53 17 ignments\A 133 141 170 110 189 100 125 105 160 190	99 13 99 126 137 152 141 192 122 172 135 106 129	18 14 - 3\Q3A.ex 168 135 168 167 187 147 136 185 154 136 112	123 190 111 146 105 128 137 120 166 156	177 103 166 143 139 122 117 144 154 186 99	157 101 198 161 141 183 181 123 173 152 190	161 152 134 156 163 153 128 171 130 182 161	163 191 193 136 147 155 140 169 151	104 181 102 158 145 139 132 128 149 108 109	144 120 110 122 104 165 114 176 149 108 158	180 115 121 140 189 175 138 172 140 157	126 117 132 128 128 130 157 196 123 120 136	

Use of Dynamic Schedule



Use of Nowait Clause



ıswer															
	140	166	133	99	168	123	177	157	161	163	104	144	180	126	
50	190	194	141	126	135	190	103	101	152	191	181	120	115	117	
94	146	125	170	137	168	111	166	198	134	193	102	110	121	132	
72	163	140	110	152	167	146	143	161	156	136	158	122	140	128	
77	115	134	189	141	187	105	139	141	163	147	145	104	189	128	
59	149	105	100	192	147	128	122	183	153	155	139	165	175	130	
3 7	143	138	125	122	136	137	117	181	128	140	132	114	138	157	
13	129	176	105	172	185	120	144	123	171	169	128	176	172	196	
11	185	189	160	135	154	166	154	173	130	151	149	149	140	123	
55	129	106	190	106	136	156	186	152	182	144	108	108	157	120	
37	121	145	105	129	112	167	99	190	161	154	109	158	123	136	
17	182	194	140	101	149	190	135	173	119	195	120	147	198	167	
33	180	133	152	198	117	137	99	187	126	166	127	192	147	182	
36	120	109	116	112	113										

Information #:

Use of Static Schedule

```
#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();</pre>
```

```
#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;
}
Use of Dynamic Schedule
#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];
```

Department of Computer Science and Engineering

```
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;
}
```

Use of Nowait Clause

```
#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;
}</pre>
```

Github Link:

https://github.com/Kunalkadam179/HPC-Assignment/tree/main/Assignment%20-%203