**CSE5006: Multicore Architectures**

**Laboratory Record**

**Submitted by**

**To**

**School of Computer Science and Engineering**

**September2020**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week** | **Date** | **S.No** | **Topic** | **Page No.** |
| **1** | **29-Jul-20** |  | **Context Setting** |  |
|  |  |  |  |  |
| **2** | **01-Aug-20** |  | **Introduction to OpenMP** | **1** |
|  |  | 1 | How to create a Project using Visual Studio |  |
|  |  | 2 | Writing Sample OpenMp Program |  |
|  |  | 3 | Setting up properties |  |
|  |  | 4 | How to compile & Execute OpenMP program |  |
|  |  | 5 | OpenMP manual study |  |
|  |  | 6 | a/c creation on Intel |  |
|  |  |  |  |  |
| **3** | **05-Aug-20** |  | **Develop a program using following construct and describe scenario for the need of construct** | **6** |
|  |  | 1 | parallel Construct |  |
|  |  | 2 | Determine the Number of processors in a parallel Region |  |
|  |  | 3 | Find the thread ID of each processor |  |
|  |  |  |  |  |
| **4** | **12-Aug-20** |  | **Execution Time computation** | **8** |
|  |  | 1 | Execution Time using OpenMP clock |  |
|  |  | 2 | Execution Time using windows clock |  |
|  |  |  |  |  |
| **5** | **19-Aug-20** |  | **Develop a sample program using Execution Environment Routines and write interesting observations by comparing various routines** | **11** |
|  |  | 1 | theomp\_set\_num\_threads routine. |  |
|  |  | 2 | theomp\_get\_num\_threads routine. |  |
|  |  | 3 | theomp\_get\_max\_threads routine. |  |
|  |  | 4 | theomp\_get\_thread\_num routine. |  |
|  |  | 5 | theomp\_get\_num\_procs routine. |  |
|  |  | 6 | theomp\_in\_parallel routine. |  |
|  |  | 7 | theomp\_set\_dynamic routine. |  |
|  |  | 8 | theomp\_get\_dynamic routine. |  |
|  |  | 9 | the omp\_get\_cancellation routine |  |
|  |  | 10 | theomp\_set\_nested routine. |  |
|  |  | 11 | theomp\_get\_nested routine. |  |
|  |  | 12 | theomp\_set\_schedule routine. |  |
|  |  | 13 | theomp\_get\_schedule routine. |  |
|  |  | 14 | theomp\_get\_thread\_limit routine. |  |
|  |  | 15 | theomp\_set\_max\_active\_levels routine. |  |
|  |  | 16 | theomp\_get\_max\_active\_levels routine. |  |
|  |  | 17 | theomp\_get\_level routine. |  |
|  |  | 18 | theomp\_get\_ancestor\_thread\_num routine. |  |
|  |  | 19 | theomp\_get\_team\_size routine. |  |
|  |  | 20 | theomp\_get\_active\_level routine. |  |
|  |  | 21 | theomp\_in\_final routine. |  |
|  |  | 22 | theomp\_get\_proc\_bind routine. |  |
|  |  | 23 | theomp\_set\_default\_device routine. |  |
|  |  | 24 | theomp\_get\_default\_device routine. |  |
|  |  | 25 | theomp\_get\_num\_devices routine. |  |
|  |  | 26 | theomp\_get\_num\_teams routine. |  |
|  |  | 27 | theomp\_get\_team\_num routine. |  |
|  |  |  |  |  |
| **6** | **26-Aug-20** |  | **Develop a program using following construct and describe scenario for the need of construct** | **14** |
|  |  | 1 | Worksharing Constructs |  |
|  |  | 1.1 | loop construct |  |
|  |  | 1.2 | sections construct |  |
|  |  | 1.3 | single construct |  |
|  |  |  |  |  |
| **7** | **09-Sep-20** |  | **Develop a program using following construct and describe scenario for the need of construct** |  |
|  |  | 1 | schedule clause |  |
|  |  | 1.1 | Static |  |
|  |  | 1.2 | Dynamic |  |
|  |  | 1.3 | Guided |  |
|  |  |  |  |  |
| **8** | **16-Sep-20** |  | **Develop a program using following construct and describe scenario for the need of construct** |  |
|  |  | 1 | Data Environment Constructs |  |
|  |  | 1.1 | Shared Clause |  |
|  |  | 1.2 | Critical Construct |  |
|  |  | 1.3 | Reduction Clause |  |
|  |  |  |  |  |
| **9** | **19-Sep-20** |  | **Develop a program using following construct and describe scenario for the need of construct** |  |
|  |  | 1 | Master Construct |  |
|  |  | 2 | Nowait clause |  |
|  |  | 3 | Barrier Construct |  |
|  |  | 4 | Atomic Construct |  |
|  |  |  |  |  |
| **10-13** | **23-Sep-20**  **30-Sep-20**  **07-Oct-20**  **14-Oct-20** |  | **Develop parallel programs for given serial programs and profile the program using VtuneAnalyis tool** |  |
|  |  | 1 | Convolution |  |
|  |  | 2 | Matrix-Matrix multiplication |  |
|  |  | 3 | Matrix-Vector multiplication |  |
|  |  | 4 | Prefix Scans |  |
|  |  | 5 | Quicksort |  |
|  |  | 6 | Minimum Spanning Tree |  |
|  |  |  |  |  |
| **14-16** | **21-Oct-20**  **28-Oct-20**  **31-Oct-20** |  | **Sample CUDA Programs** |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Week:2  Date: 01/08/2020 | **Introduction to Open-MP and perform the following functions:**  **1. How to create a Project using Visual Studio**  **2. Writing Sample Open-MP program**  **3. Setting up properties**  **4. How to compile & execute Open-MP program** | | | | |
|  | | | | | |
| Objectives | To Demonstrate how to create project in Visual Studio,Writting Code,complie and Execute it. | | | | |
|  | | | | | |
| Requirements |  | | | | |
| Hardware: | S.No | Hardware | | Minimum Req. | Recommended |
| 1 | Number of Cores | | 2 | 4 |
| 2 | Clock | | 1.4 GHz | 2.2GHz |
| 3 | RAM | | 2GB | 4 GB |
| 4 | Hard Disk Space | | 20GB-50GM | 50GB |
| Software: | S.No | Software | | Version Details | Description |
| 1 | Visual studio | | 2019 | Development Environment for C,C++,C# etc. |
| 2 | OpenMP | | 4.0 | The parallel programming library to develop parallel applications |
|  |  | |  |  |
|  | | | | | |
| Program | #include<stdio.h>  #include<omp.h>  int main()  {  int i;  #pragma omp parallel for  for (i = 0;i <= 5;i++)  {  printf("Prachi %d Printed By Thread Num=%d \n", i, omp\_get\_thread\_num());  }  return 0;  } | | | | |
|  | | | | | |
| Description of OpenMP construct | S.No | OpenMP Construct | Category of Construct &  Description of construct with reference to the application/program | | |
| 1 | #pragma omp parallel | To parallelize structured block | | |
| 2 | omp\_get\_thread\_num() | This is used to give the thread number which is currently executing. | | |
|  | | | | | |
| Input: | NA | | | | |
| Output: |  | | | | |
|  | | | | | |
| Experimental Set-up Details |  | | | | |
|  | | | | | |
|  | | | | | |
| Observations | Steps to create and Execute the openmp program:  Step1: create a new project    Step2: Select Empty project.      Step3: Enter a project name.    Step4: After that, right click on source file and add new item.    Step5:Select C++ file.    Step6: then write the oepnmp program and set the properties.    Step7: code generation -> Runtime library -> multi-threaded.  Then in Language-> openmp->yes    Step 8: then after successful completion of writing code you have to compile the code by Ctrl+F7. And then run the code by using Ctrl+F5. | | | | |
|  | | | | | |
| References | <https://www.openmp.org/>  <https://docs.microsoft.com/en-us/cpp/parallel/openmp/openmp-in-visual-cpp> | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Week:3  Date: 05/08/2020 | **Develop a program using following construct and describe scenario for the need of the**  **construct:**  **5. Parallel construct**  **6. Determine the number of threads/processors in a parallel Region**  **7. Find the thread ID of each thread/processor** | | | | |
|  | | | | | |
| Objectives | 1. To demonstrate the utility of parallel construct 2. Determine the number of processors in a parallel region 3. To find thread ID of each processor | | | | |
|  | | | | | |
| Requirements |  | | | | |
| Hardware: | S.No | Hardware | | Minimum Req. | Recommended |
| 1 | Number of Cores | | 2 | 4 |
| 2 | Clock | | 1.4 GHz | 2.2GHz |
| 3 | RAM | | 2GB | 4 GB |
| 4 | Hard Disk Space | | 20GB-50GM | 50GB |
| Software: | S.No | Software | | Version Details | Description |
| 1 | Visual studio | | 2019 | Development Environment for C,C++,C# etc. |
| 2 | OpenMP | | 4.0 | The parallel programming library to develop parallel applications |
|  |  | |  |  |
|  | | | | | |
| Program | #include <stdio.h>  #include <conio.h>  #include <omp.h>  int main()  {  int in, i, threadNo;  #pragma omp parallel  in = omp\_get\_max\_threads();  printf("\nNo. of threads inside parallel region: %d", in);  #pragma omp parallel for  for (i = 0; i < 5; i++) {  printf("\nIteration %d printed by thread %d", i, omp\_get\_thread\_num());  }  } | | | | |
|  | | | | | |
| Description of OpenMP construct | S.No | OpenMP Construct | Category of Construct &  Description of construct with reference to the application/program | | |
| 1 | #pragma omp parallel | To parallelize structured block | | |
| 2 | #pragma omp for | To distribute work among threads in the for loop | | |
|  | | | | | |
| Input: | NA | | | | |
| Output: |  | | | | |
|  | | | | | |
| Experimental Set-up Details |  | | | | |
|  | | | | | |
|  | | | | | |
| Observations | Demonstrated function of parallel construct and retrieved number processors in parallel region and thread ID. | | | | |
|  | | | | | |
| References | https://www.openmp.org/ | | | | |
| p. NoWeek:3  Date: 30/07/2019 | of construct | | | | |
| Week:4  Date:  **12-08-2020** | **Execution Time Computation**  **1. Execution time using OpenMP clock**  **2. Execution time using Windows clock** | | | | |
|  | | | | | |
| Objectives | To calculate the execution time of program we need time measering Construct and can be done using  1)OpenMp Clock  2) Window Clock | | | | |
|  | | | | | |
| Requirements |  | | | | |
| Hardware: | S.No | Hardware | | Minimum Req. | Recommended |
| 1 | Number of Cores | | 2 | 4 |
| 2 | Clock | | 1.4 GHz | 2.2GHz |
| 3 | RAM | | 2GB | 4 GB |
| 4 | Hard Disk Space | | 20GB-50GM | 50GB |
| Software: | S.No | Software | | Version Details | Description |
| 1 | Visual studio | | 2019 | Development Environment for C,C++,C# etc. |
| 2 | OpenMP | | 4.0 | The parallel programming library to develop parallel applications |
|  |  | |  |  |
|  | | | | | |
| Program | #include <stdio.h>  #include <omp.h>  #include <time.h>  int main()  {  double omp\_start\_time\_0, omp\_start\_time\_1;  double windows\_start\_time = clock();  double omp\_start\_time = omp\_get\_wtime();  #pragma omp parallel  {  int thread\_id = omp\_get\_thread\_num();  if (thread\_id == 0)  omp\_start\_time\_0 = omp\_get\_wtime();  if (thread\_id == 1)  omp\_start\_time\_1 = omp\_get\_wtime();    }  double windows\_end\_time = clock();  double omp\_end\_time = omp\_get\_wtime();  printf("\n\nProgram Execution Time Using Windows Clock: %f seconds", (windows\_end\_time -  windows\_start\_time));  printf("\nProgram Execution Time Using OpenMP Clock: %f milliseconds", (omp\_end\_time -  omp\_start\_time));  printf("\nTime taken by Thread-0: %f", (omp\_end\_time - omp\_start\_time\_0));  printf("\nTime taken by Thread-1: %f", (omp\_end\_time - omp\_start\_time\_1));    return 0;  } | | | | |
|  | | | | | |
| Description of OpenMP construct | S.No | OpenMP Construct | Category of Construct &  Description of construct with reference to the application/program | | |
| 1 | #pragma omp parallel | To parallelize structured block | | |
| 2 | omp\_get\_thread\_num() | This is used to give the thread number which is currently executing. | | |
|  | 3 | omp\_get\_wtime() | It is used to find out the taken by each thread in this code. | | |
|  | | | | | |
| Input: | NA | | | | |
| Output: |  | | | | |
|  | | | | | |
| Experimental Set-up Details |  | | | | |
|  | | | | | |
|  | | | | | |
| Observations | Here We have written a code to demonstrate the how we can use the time measering construct of opennmp and  the window time .Here we have calculated the time taken by each thread to do some task and also wh have the  time taken by parallel region by both the ways 1) omp\_get\_time();  2 )clock() of window. | | | | |
|  | | | | | |
| References | <https://www.openmp.org/>  https://docs.microsoft.com/en-us/cpp/parallel/openmp/openmp-in-visual-cpp | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Week:5  Date:  **19-08-2020** | **Develop a sample program using Execution Environment Routines and write**  **interesting observations by comparing various routines** | | | | | |
|  | | | | | | |
| Objectives | To execute an OpenMP program demonstrating the results of various environment constructs. | | | | | |
|  | | | | | | |
| Requirements |  | | | | | |
| Hardware: | S.No | Hardware | | | Minimum Req. | Recommended |
| 1 | Number of Cores | | | 2 | 4 |
| 2 | Clock | | | 1.4 GHz | 2.2GHz |
| 3 | RAM | | | 2GB | 4 GB |
| 4 | Hard Disk Space | | | 20GB-50GM | 50GB |
| Software: | S.No | Software | | | Version Details | Description |
| 1 | Visual studio | | | 2019 | Development Environment for C,C++,C# etc. |
| 2 | OpenMP | | | 4.0 | The parallel programming library to develop parallel applications |
|  |  | | |  |  |
|  | | | | | | |
| Program | #include <stdio.h>  #include <omp.h>  int main()  {  int j;  int noOfThreads;  omp\_set\_num\_threads(2);  if (omp\_in\_parallel != 0)  printf("\noutside parallel region");  #pragma omp parallel  if (omp\_in\_parallel)  printf("\ninside parallel region");  noOfThreads = omp\_get\_num\_threads();  printf("\nNo of threads in parallel region are: %d", noOfThreads);  printf("\nMaximum number of threads in parallel region are: %d", omp\_get\_max\_threads());  printf("\nnumber of processors available in parallel region are: %d", omp\_get\_num\_procs());  #pragma omp parallel for  for (j = 0; j < 5; j++) {  printf("\niteration is : %d and hello by thread: %d", j, omp\_get\_thread\_num());  }  return 0;  } | | | | | |
|  | | | | | | |
| Description of OpenMP construct | S.No | OpenMP Construct | | Category of Construct &  Description of construct with reference to the application/program | | |
| 1 | omp\_set\_num\_threads | | Specifies the number of threads used by default in subsequent parallel sections | | |
| 2 | omp\_in\_parallel | | This function returns true if currently running in parallel, false otherwise | | |
|  | 3 | omp\_get\_num\_threads | | Returns the number of threads in the current team. In a sequential section of the program omp\_get\_num\_threads returns 1 | | |
|  | 4 | omp\_get\_max\_threads | | maximum number of threads that can be used to form a new team if a parallel region | | |
|  | 5 | omp\_get\_num\_procs | | The number of online processors on the machine | | |
|  | 6 | omp\_get\_thread\_num | | function returns the number of the currently executing thread within the team. The number returned will always be between 0 and NUM\_PARTHDS - 1 | | |
|  | | | | | | |  |
| Input: | NA | | | | | |
| Output: |  | | | | | |
|  | | | | | | |
| Experimental Set-up Details |  | | | | | |
|  | | | | | | |
|  | | | | | | |
| Observations | Here we have executed some Environment routine to get the idea about there functionlity and how they are used in the effectively way to parallelize the code | | | | | |
|  | | | | | | |
| References | https://www.openmp.org/ | | | | | |
| p. No.:1,2,3  Week:3  Date: 30/07/2019 | of construct | | | | | |
| Week:6  Date: 30/07/2019 | **An OpeMP programs for demonstrate of Worksharing Constructs**  **1.Loop Constructs**  **2.Section Constructs**  **3.Single Construct** | | | | | |
|  | | | | | | |
| Objectives | 1. To demonstrate how the loop is parallelized by dividing the iterations among the threads using Loop  Construct  2. To demonstrate how the omp sections directive distributes work among threads bound to a defined  parallel region.  3. The omp single directive identifies a section of code that must be run by a single available thread. | | | | | |
|  | | | | | | |
| Requirements |  | | | | | |
| Hardware: | S.No | Hardware | | | Minimum Req. | Recommended |
| 1 | Number of Cores | | | 2 | 4 |
| 2 | Clock | | | 1.4 GHz | 2.2GHz |
| 3 | RAM | | | 2GB | 4 GB |
| 4 | Hard Disk Space | | | 20GB-50GM | 50GB |
| Software: | S.No | Software | | | Version Details | Description |
| 1 | Visual studio | | | 2019 | Development Environment for C,C++,C# etc. |
| 2 | OpenMP | | | 4.0 | The parallel programming library to develop parallel applications |
|  |  | | |  |  |
|  | | | | | | |
| Program | //Program for Loop Construct  #include<stdio.h>  #include<omp.h>  void main()  {  int i;  #pragma omp parallel for  for (i = 0;i <8;i++)  {  printf("\nIteration %d printed by thread %d", i, omp\_get\_thread\_num());  }  }  // program for section construct  #include <stdio.h>  #include <omp.h>  int sum(int a, int b, int c) {  return (a + b + c);  }  float avg(int a, int b, int c) {  return ((a + b + c) / 3);  }  int max(int a, int b, int c) {  if (a >= b && a >= c) {  return a;  }  else if (b >= a && b >= c) {  return b;  }  else if (c >= a && c >= b) {  return c;  }  }  int min(int a, int b, int c)  {  if (a < b && a < c) {  return a;  }  else if (b < a && b < c) {  return b;  }  else {  return c;  }  }  int main()  {  int a, b, c;  printf("Enter 3 Numbers");  scanf\_s("%d%d%d", &a, &b, &c);  #pragma omp parallel  #pragma omp sections  {  #pragma omp section  {  printf("\nsum is : %d calculated by thread number: %d \n", sum(a, b, c), omp\_get\_thread\_num());  }  #pragma omp section  {  printf("\naverage is : %f calculated by thread number: %d \n", avg(a, b, c), omp\_get\_thread\_num());  }  # pragma omp section  {  printf("\nminimum is : %d calculated by thread number: %d \n", min(a, b, c), omp\_get\_thread\_num());  }  # pragma omp section  {  printf("\nmaximum is : %d calculated by thread number: %d \n", max(a, b, c), omp\_get\_thread\_num());  }  }  return 0;  }  //Program for Single Construct  #include <stdio.h>  #include <omp.h>  void testFunc() {  printf("\nfunction executed by multiple threads, thread no: %d", omp\_get\_thread\_num());  }  void singleTestFunc() {  printf("\n\nfunction executed by single thread, thread no: %d\n\n", omp\_get\_thread\_num());  }  int main()  {  #pragma omp parallel  testFunc();  #pragma omp single  {  singleTestFunc();  }  #pragma omp parallel  testFunc();  return 0;  } | | | | | |
|  | | | | | | |
| Description of OpenMP construct | S.No | OpenMP Construct | Category of Construct &  Description of construct with reference to the application/program | | | |
| 1) | #pragma omp parallel | The loop is parallelized by dividing the iterations among the threads using Loop | | | |
| 2) | #pragma omp sections | The omp sections directive distributes work among threads bound to a defined | | | |
|  | 3) | #pragma omp  single | The omp single directive identifies a section of code that must be run by a single  available thread here parallel loop is executing but “Sum” is shown in Single  Constructs | | | |
|  | | | | | | |
| Input: | NA | | | | | |
| Output: | Output of loop Construct:    Output of section construct:    Output of single construct: | | | | | |
|  | | | | | | |
| Experimental Set-up Details |  | | | | | |
|  | | | | | | |
|  | | | | | | |
| Observations | We have demonstrated ,  1) For loop construct by execting for loop and using the runtime routines os openmp for output veryfications. And  found that each core is printing each iteration for each time.  2) Single Construct executes corresponding block even if the code is executing in parallel regions.  3) Section construct by ruuing code for AVG and SUM fuctions in parallel along with we have taken the time  for execution of code in sec | | | | | |
|  | | | | | | |
| References | <https://www.openmp.org/>  https://docs.microsoft.com/en-us/cpp/parallel/openmp/openmp-in-visual-cpp | | | | | |