**Homework 3**

**Parallel Distributed Num Algorithms**

**Name: Saurabh Kumar**

**UIN: 926009924**

# Parallelization with OpenMP.

Ans:.

## 1.1 Parallelized functions Code :

**Ans:** The modified code is submitted on ecampus

## 1.2 SpeedUp/Efficiency Calculations:

**Ans:**

**For Function LU:**

|  |  |  |  |
| --- | --- | --- | --- |
| **threads** | **Execution time (secs)** | **Speedup** | **Efficiency** |
| 1 | 778 secs | 1 | 1 |
| 2 | 378 secs | 2.1 | 1.05 |
| 4 | 202 secs | 3.9 | .98 |
| 10 | 176 secs | 4.4 | 0.44 |
| 20 | 104 secs | 7.5 | 0.37 |

**For Function solve\_L:**

|  |  |  |  |
| --- | --- | --- | --- |
| **p** | **Execution time (secs)** | **Speedup** | **Efficiency** |
| 1 | .68 | 1 | 1 |
| 2 | .37 | 1.8 | .9 |
| 4 | .32 | 2.1 | .53 |
| 10 | .98 | .69 | 0.06 |
| 20 | 1.1 | .62 | 0.03 |

**For Function solve\_U:**

|  |  |  |  |
| --- | --- | --- | --- |
| **p** | **Execution time (secs)** | **Speedup** | **Efficiency** |
| 1 | .67 | 1 | 1 |
| 2 | .38 | 1.8 | .9 |
| 4 | .29 | 2.3 | .58 |
| 10 | .87 | .77 | 0.07 |
| 20 | 1.02 | .66 | 0.03 |

**For Function matvec:**

Could not parallelize this function. Tried out some combinations but it kept inducing error in my results.

**For Function saxpy:**

|  |  |  |  |
| --- | --- | --- | --- |
| **p** | **Execution time (secs)** | **Speedup** | **Efficiency** |
| 1 | 1.9073e-05 | 1 | 1 |
| 2 | 2.0981e-05 | .91 | .46 |
| 4 | 1.8120e-05 | 1.1 | .26 |
| 10 | 5.0712e-04 | .04 | 0.004 |
| 20 | 2.8791e-03 | .006 | .0003 |

**For Function norm:**

|  |  |  |  |
| --- | --- | --- | --- |
| **p** | **Execution time (secs)** | **Speedup** | **Efficiency** |
| 1 | 7.2002e-05 | 1 | 1 |
| 2 | 3.4094e-05 | 2.1 | 1.05 |
| 4 | 4.7922e-05 | 1.5 | .38 |
| 10 | 6.3400e-03 | .011 | 0.0011 |
| 20 | 1.4591e-04 | .5 | 0.025 |

1. Strategy for exploiting shared caches

**Ans 2.1:**

In order to exploit shared caches, I have tried to use the openmp pragma directive proc\_bind. It is an openMp clause to control the thread binding for a team of threads in a parallel region. I have used the close affinity policy in proc\_bind to take advantage of the shared cache machine architecture.

**Ans 2.2 The comparison of run time, speed and efficiency with / without my strategy is as follows:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **threads** | **Execution time without (secs)** | **Speedup without** | **Efficiency without** | **Execution time with (secs)** | **Speedup with** | **Efficiency with** |
| 1 | 778 secs | 1 | 1 | 760 secs | 1 | 1 |
| 2 | 378 secs | 2.1 | 1.05 | 350 secs | 2.2 | 1.1 |
| 4 | 202 secs | 3.9 | .98 | 250 secs | 3 | .75 |
| 10 | 176 secs | 4.4 | 0.44 | 300 secs | 2.5 | .25 |
| 20 | 104 secs | 7.5 | 0.37 | 270 secs | 2.8 | .14 |

**Steps to Run:**

I have uploaded a shell script called run.sh. Please run this script using the following command after logging into ada.

This script will compile the source code and do a batch submission on ada.

**Command to run:**

* ./run.sh