**CSE 5351**

**ASSIGNMENT 4**

**NITHYA BALASUNDAR**

**1001763661**

**SOLUTIONS:**

1. **DATA PARALLEL APPROACH** using DISTRIBUTED NON-SHARED MEMORY for SIEVE of Erastosthenes.

**RESULTS:**

Assume n=1000,then there will be 168 prime numbers .

|  |  |
| --- | --- |
| **Number of Processors** | **Time** |
| 1 | 0.000139 |
| 2 | 0.000259 |
| 4 | 0.000251 |
| 8 | 0.000331 |
| 16 | 0.0000478 |
| 32 | 0.0000734 |

**CURVE:**

The Number of Processors (n), values chosen are 1,2,4,8,16,32.

**PARALLELIZATION SCHEME :**

Using parallelization we can reduce the execution and increase the efficiency. In this program ,data decomposition and MPI B\_cast is used to perform parallelization. Each task represents an integer,they communicate and iterate until the loop. Reduction is performed to determine the new value of n and then we broadcast to all other tasks .

FOLLOWING FUNCTIONS ARE USED :

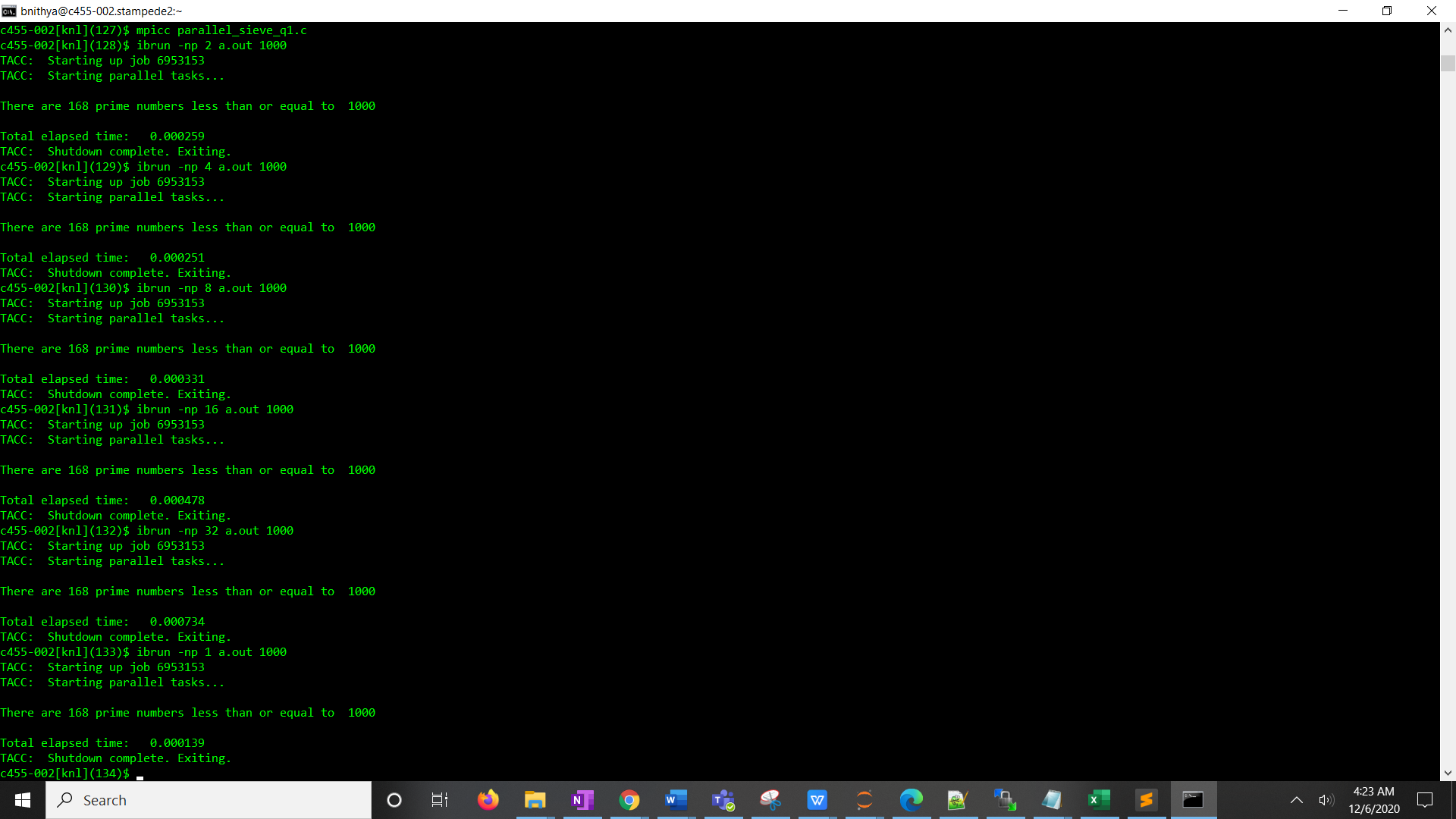
* BLOCK\_FIRST will store the first multiple and current prime number.
* BLOCK\_STEP for iterating odd nos.
* BLOCK\_LOW to store lowest index.
* BLOCK\_HIGH to store highest index.
* BLOCK\_SIZE identifies no of elements.
* BLOCK\_OWNER to identify the group.

The indices are divided from 0 to n - 1 into size nearly even contiguous groups.For example, if we had n = 14 elements and 4 groups, we might decompose the element indices into the groups {0, 1, 2}, {3, 4, 5, 6}, {7, 8, 9}, {10, 11,12, 13}.

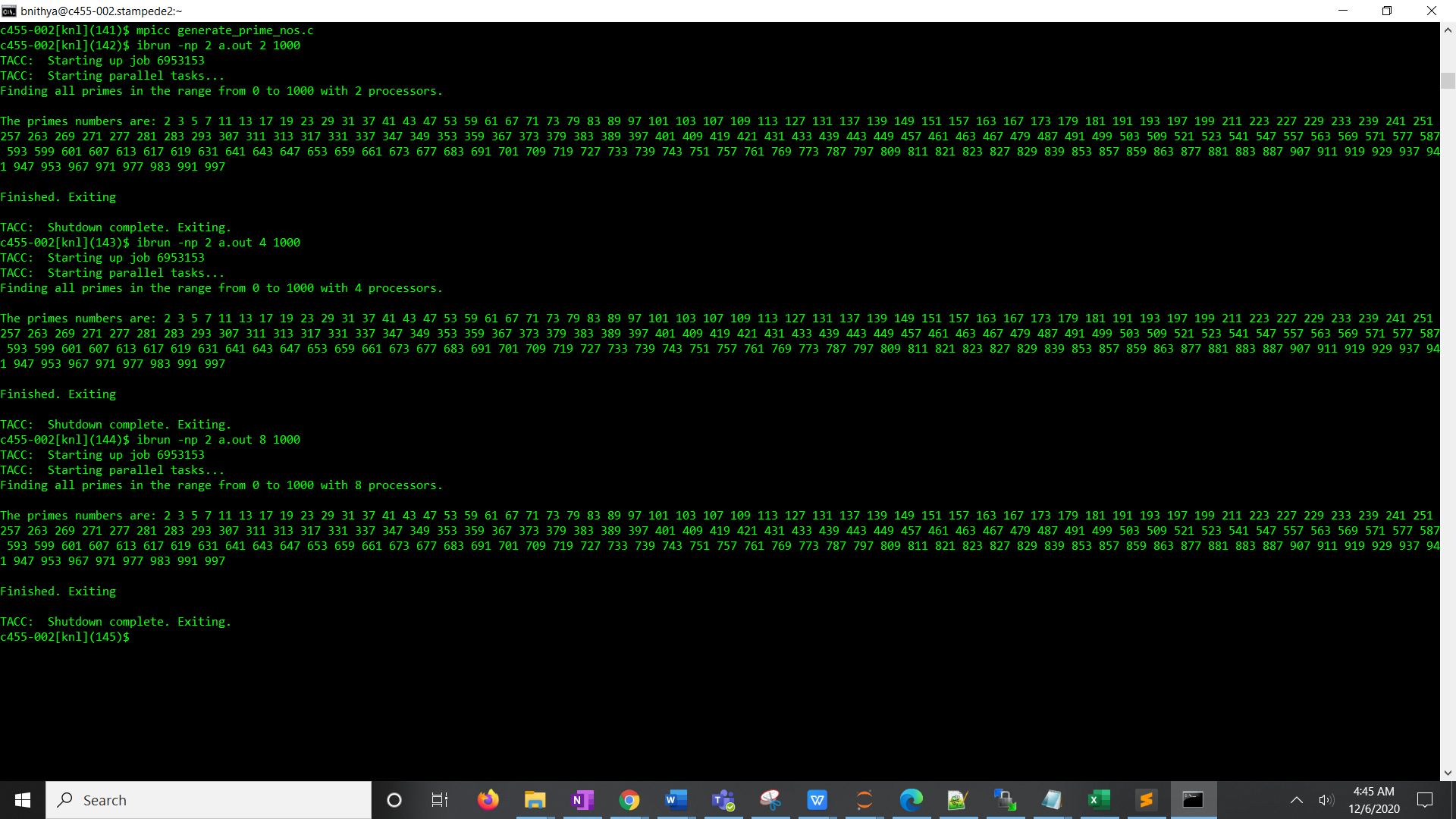
In this way parallelization scheme is optimized and implemented.

**OUTPUT:**

1.Calculating the time and number of prime numbers generated :



2.Generating all the prime numbers



2. **Parallel program using MPI to compute π using Simpson’s Rule**

**BLOCKING SEND :**

RESULTS:

The value of PI (from processor 0 only).For different values of n,assume intervals =20000

|  |  |
| --- | --- |
| N(no of processors) | Value of pi |
| 2 | 3.1416459863892134 |
| 4 | 3.1416620258145658 |
| 8 | 3.1416774334017399 |
| 32 | 3.1416498991832329 |
| 64 | 3.1416231736310665 |

The value of pi when we use 8 processors and 20000 intervals is **3.1416774334017399** using Simpsons method.

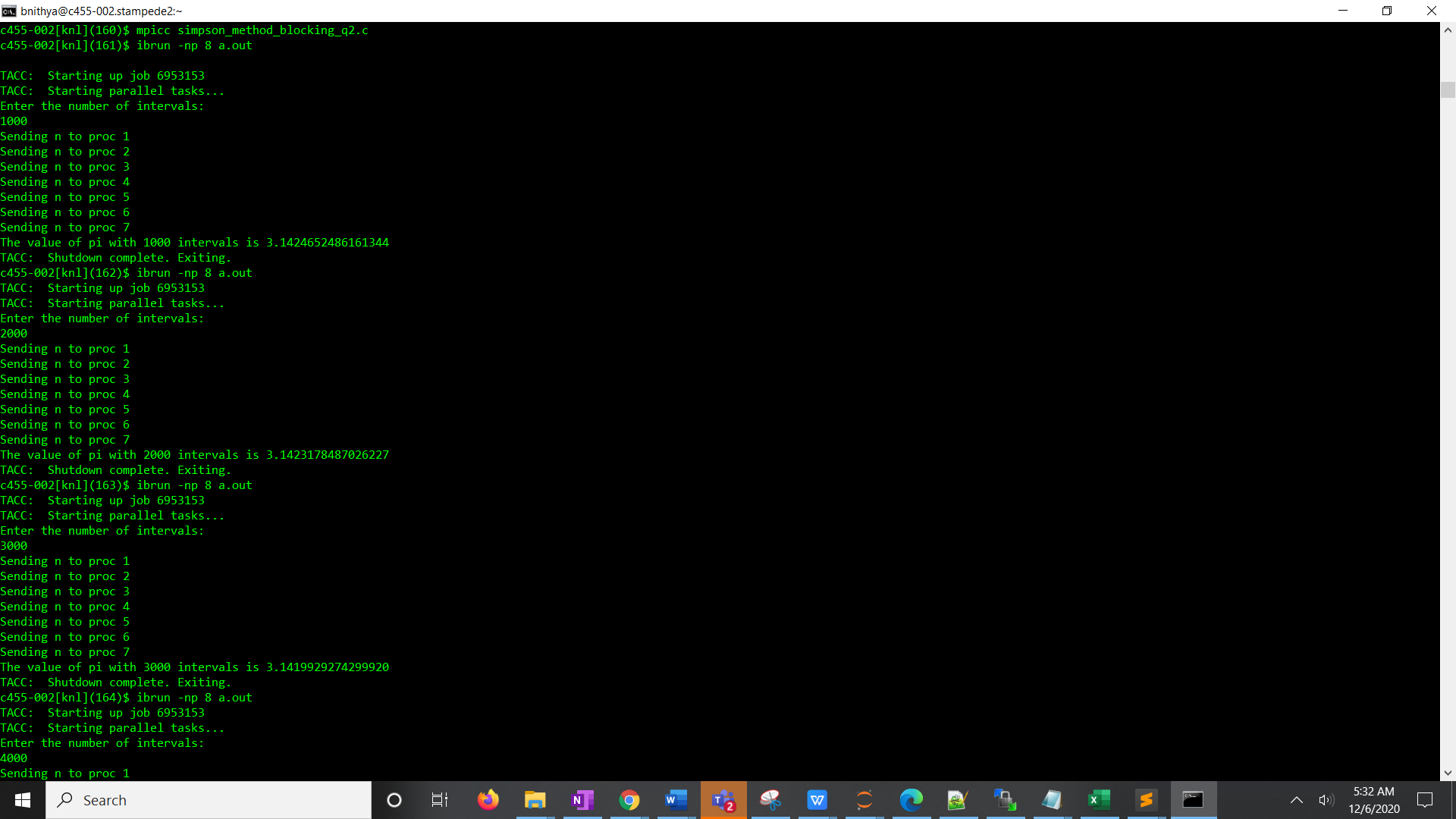
Graphically,we can represent this as,

Alternatively,Assume no of processors(n)=8,

|  |  |
| --- | --- |
| Number of Intervals | Value of PI |
| 1000 | 3.1424652486161344 |
| 2000 | 3.1423178487026227 |
| 3000 | 3.1419929274299920 |
| 4000 | 3.1418144289995862 |
| 5000 | 3.1418328189504119 |
| 6000 | 3.1416058752411207 |
| 7000 | 3.1417193828605110 |
| 8000 | 3.1417329297429966 |
| 9000 | 3.1417810526366692 |
| 10000 | 3.1417376951186804 |
| 20000 | 3.1416774334017399 |

OUTPUT:

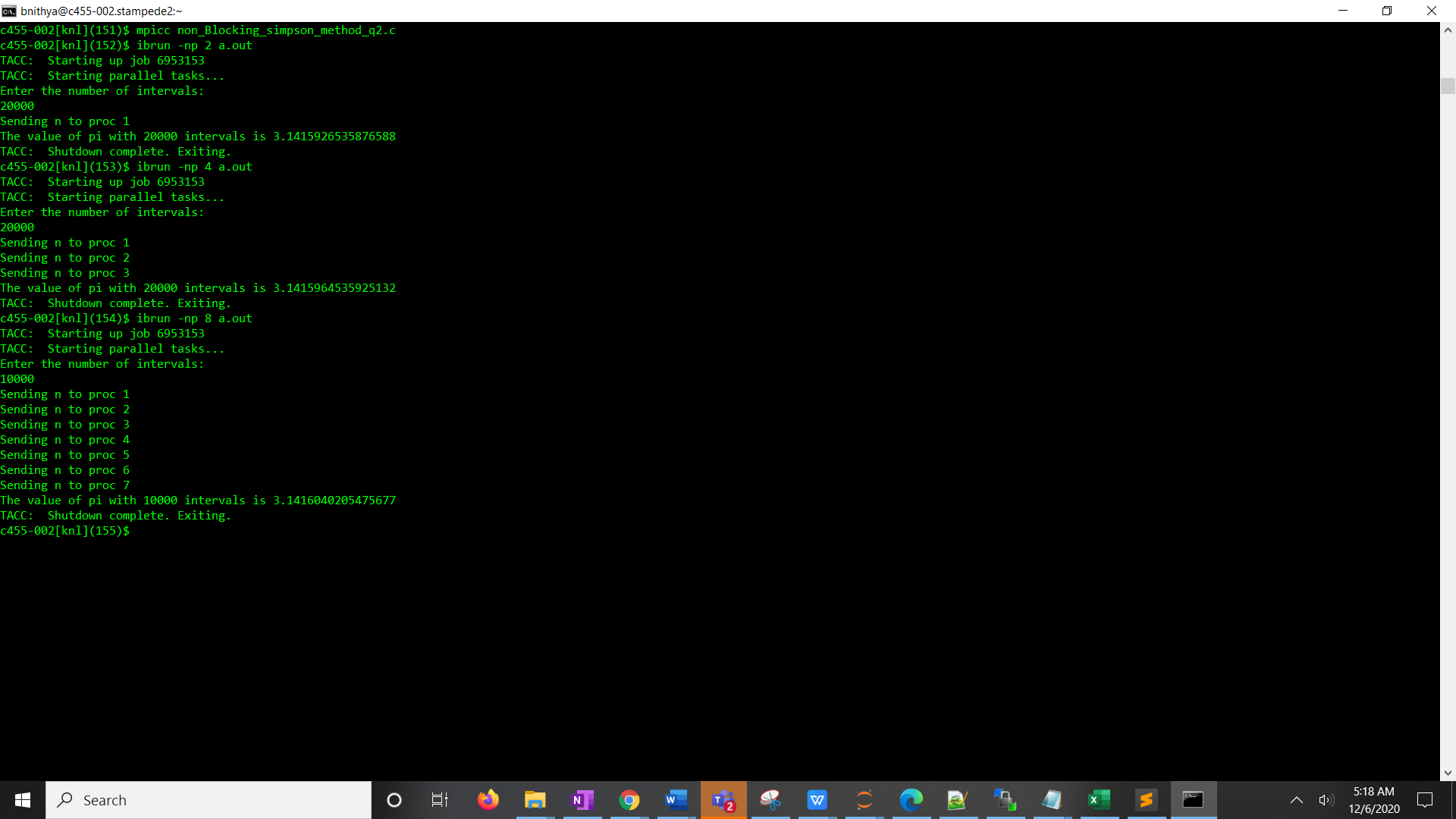
Assuming the no of processor as 8,we calculate for different results as shown below :



NON BLOCKING SEND:

OUTPUT:

Assuming the no of processor as 2,4,8 and number of interval as 20000,the output is as follows:



3.**MPI PROGRAM**

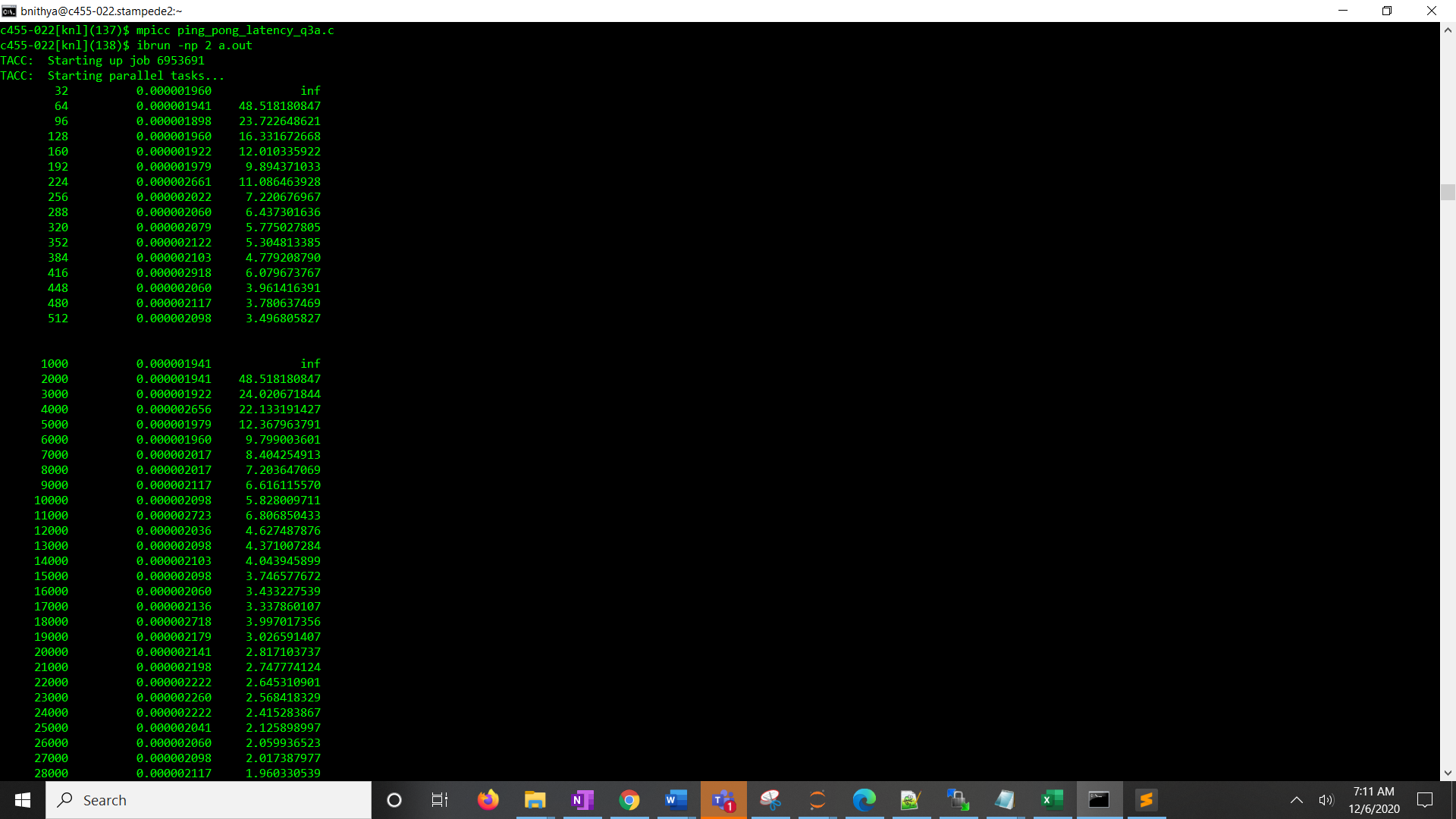
3a)PING PONG METHOD:

The MPI program for calculating the latency and communication time between two nodes using the ping pong algorithm shows the following results:

OUTPUT:

The 32byte and 1k byte increment is shown below:

* The first column represents the DATA SIZE.
* The second column shows the COMMUNICATION TIME.
* The third column shows the LATENCY.



RESULTS:

FOR 32 byte increment(0 to 512 bytes):

|  |  |  |
| --- | --- | --- |
| **Data Size(bytes)** | **Communication\_times(s)** | **Latency(s)** |
| 32 | 0.00000196 | inf |
| 64 | 0.000001898 | 47.44529724 |
| 96 | 0.000001941 | 24.25909042 |
| 128 | 0.00000196 | 16.33167267 |
| 160 | 0.000002699 | 16.86811447 |
| 192 | 0.000001917 | 9.58442688 |
| 224 | 0.00000196 | 8.165836334 |
| 256 | 0.000001984 | 7.084437779 |
| 288 | 0.00000206 | 6.437301636 |
| 320 | 0.000002079 | 5.775027805 |
| 352 | 0.00000268 | 6.699562073 |
| 384 | 0.000002079 | 4.725022749 |
| 416 | 0.000002136 | 4.450480143 |
| 448 | 0.000002041 | 3.92473661 |
| 480 | 0.000002017 | 3.601823534 |
| 512 | 0.000002141 | 3.568331401 |

**CURVE:**

FOR 1k increment(1k to 128k bytes),

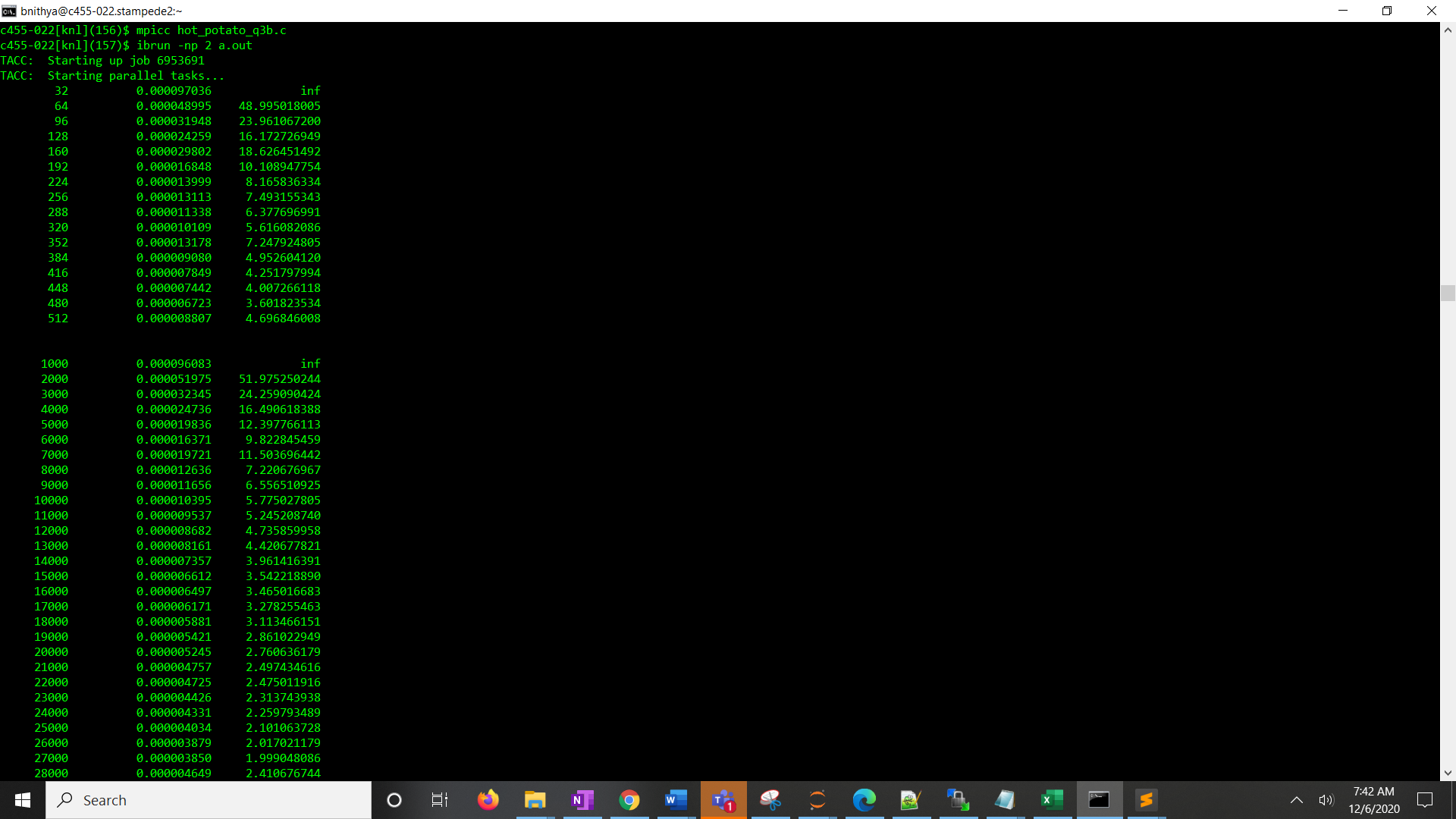
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Data Size** | **Communication\_time** | **Latency** |  | **Data Size** | **Communication\_time** | **Latency** |
| 1000 | 0.000002561 | inf |  | 41000 | 0.000002079 | 1.299381 |
| 2000 | 0.000001922 | 48.04134369 |  | 42000 | 0.00000206 | 1.256059 |
| 3000 | 0.000001941 | 24.25909042 |  | 43000 | 0.000002518 | 1.498631 |
| 4000 | 0.000001979 | 16.49061839 |  | 44000 | 0.000002141 | 1.244767 |
| 5000 | 0.000001941 | 12.12954521 |  | 45000 | 0.00000206 | 1.170418 |
| 6000 | 0.000001941 | 9.703636169 |  | 46000 | 0.000002079 | 1.155006 |
| 7000 | 0.000002017 | 8.404254913 |  | 47000 | 0.000002141 | 1.163586 |
| 8000 | 0.000002503 | 8.940696716 |  | 48000 | 0.000002079 | 1.105856 |
| 9000 | 0.000002098 | 6.556510925 |  | 49000 | 0.000002141 | 1.115104 |
| 10000 | 0.000002098 | 5.828009711 |  | 50000 | 0.000002604 | 1.328332 |
| 11000 | 0.000002079 | 5.197525024 |  | 51000 | 0.00000216 | 1.080036 |
| 12000 | 0.00000206 | 4.681673917 |  | 52000 | 0.000002198 | 1.077558 |
| 13000 | 0.000002022 | 4.212061564 |  | 53000 | 0.000002117 | 1.017864 |
| 14000 | 0.00000206 | 3.961416391 |  | 54000 | 0.000002179 | 1.027899 |
| 15000 | 0.000002561 | 4.572527749 |  | 55000 | 0.000002179 | 1.008864 |
| 16000 | 0.000002098 | 3.496805827 |  | 56000 | 0.000002861 | 1.300465 |
| 17000 | 0.00000206 | 3.218650818 |  | 57000 | 0.000002198 | 0.981348 |
| 18000 | 0.000002079 | 3.057367661 |  | 58000 | 0.000002403 | 1.054061 |
| 19000 | 0.000002098 | 2.914004856 |  | 59000 | 0.000002098 | 0.904346 |
| 20000 | 0.000002079 | 2.735539487 |  | 60000 | 0.000002179 | 0.923367 |
| 21000 | 0.000002079 | 2.598762512 |  | 61000 | 0.00000216 | 0.90003 |
| 22000 | 0.000002379 | 2.832639785 |  | 62000 | 0.000002141 | 0.877459 |
| 23000 | 0.000002098 | 2.384185791 |  | 63000 | 0.000002079 | 0.83831 |
| 24000 | 0.000002098 | 2.280525539 |  | 64000 | 0.000002122 | 0.842034 |
| 25000 | 0.000002036 | 2.120931943 |  | 65000 | 0.00000226 | 0.882894 |
| 26000 | 0.000002041 | 2.040863037 |  | 66000 | 0.000002179 | 0.838133 |
| 27000 | 0.000002084 | 2.003633059 |  | 67000 | 0.000002179 | 0.825434 |
| 28000 | 0.000002022 | 1.872027362 |  | 68000 | 0.00000216 | 0.805997 |
| 29000 | 0.000002818 | 2.516167504 |  | 69000 | 0.000002542 | 0.93439 |
| 30000 | 0.000002017 | 1.738811361 |  | 70000 | 0.000002117 | 0.767086 |
| 31000 | 0.000002098 | 1.748402913 |  | 71000 | 0.000002236 | 0.798702 |
| 32000 | 0.00000206 | 1.661239132 |  | 72000 | 0.000002184 | 0.768984 |
| 33000 | 0.000002103 | 1.642853022 |  | 73000 | 0.000002217 | 0.769893 |
| 34000 | 0.000002103 | 1.593069597 |  | 74000 | 0.000002179 | 0.746283 |
| 35000 | 0.000002079 | 1.528683831 |  | 75000 | 0.000002198 | 0.742642 |
| 36000 | 0.000002818 | 2.012934004 |  | 76000 | 0.000002642 | 0.880559 |
| 37000 | 0.000002079 | 1.443756951 |  | 77000 | 0.000002136 | 0.702707 |
| 38000 | 0.000002103 | 1.420845857 |  | 78000 | 0.000002198 | 0.713708 |
| 39000 | 0.000002041 | 1.342673051 |  | 79000 | 0.000002179 | 0.698444 |
| 40000 | 0.000002041 | 1.308245537 |  | 80000 | 0.000002203 | 0.697148 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Data Size** | **Communication\_time** | **Latency** |  | **Data Size** | **Communication\_time** | **Latency** |
| 81000 | 0.000002117 | 0.661611557 |  | 121000 | 0.000002437 | 0.507633 |
| 82000 | 0.000002198 | 0.678462747 |  | 122000 | 0.000002518 | 0.520186 |
| 83000 | 0.000002136 | 0.651289777 |  | 123000 | 0.00000236 | 0.483677 |
| 84000 | 0.000002179 | 0.656369221 |  | 124000 | 0.000002418 | 0.491375 |
| 85000 | 0.000002117 | 0.630106245 |  | 125000 | 0.000002723 | 0.54894 |
| 86000 | 0.000002141 | 0.629705541 |  | 126000 | 0.000002542 | 0.508308 |
| 87000 | 0.000002122 | 0.616838766 |  | 127000 | 0.000002518 | 0.499544 |
| 88000 | 0.00000216 | 0.620710439 |  | 128000 | 0.00000236 | 0.464635 |
| 89000 | 0.000002179 | 0.619075515 |  |  |  |  |
| 90000 | 0.000002141 | 0.601404169 |  |  |  |  |
| 91000 | 0.000002117 | 0.588099162 |  |  |  |  |
| 92000 | 0.000002804 | 0.770275409 |  |  |  |  |
| 93000 | 0.000002203 | 0.598637954 |  |  |  |  |
| 94000 | 0.000002184 | 0.587073706 |  |  |  |  |
| 95000 | 0.00000226 | 0.601119183 |  |  |  |  |
| 96000 | 0.000002623 | 0.690159045 |  |  |  |  |
| 97000 | 0.000002279 | 0.593562921 |  |  |  |  |
| 98000 | 0.000002279 | 0.587443716 |  |  |  |  |
| 99000 | 0.000005002 | 1.276025967 |  |  |  |  |
| 100000 | 0.00000236 | 0.596046448 |  |  |  |  |
| 101000 | 0.000002298 | 0.574588776 |  |  |  |  |
| 102000 | 0.000002837 | 0.702272547 |  |  |  |  |
| 103000 | 0.00000258 | 0.632276722 |  |  |  |  |
| 104000 | 0.000002337 | 0.567112154 |  |  |  |  |
| 105000 | 0.000002379 | 0.571975341 |  |  |  |  |
| 106000 | 0.000002437 | 0.580151876 |  |  |  |  |
| 107000 | 0.000003462 | 0.816471172 |  |  |  |  |
| 108000 | 0.000002556 | 0.597160553 |  |  |  |  |
| 109000 | 0.000002623 | 0.607084345 |  |  |  |  |
| 110000 | 0.000002441 | 0.559955562 |  |  |  |  |
| 111000 | 0.00000236 | 0.536441803 |  |  |  |  |
| 112000 | 0.000002341 | 0.527313164 |  |  |  |  |
| 113000 | 0.000003157 | 0.704612051 |  |  |  |  |
| 114000 | 0.000002661 | 0.588661801 |  |  |  |  |
| 115000 | 0.000002379 | 0.521802066 |  |  |  |  |
| 116000 | 0.000002341 | 0.508971836 |  |  |  |  |
| 117000 | 0.000002422 | 0.522054475 |  |  |  |  |
| 118000 | 0.000002441 | 0.521668002 |  |  |  |  |
| 119000 | 0.000002604 | 0.551595526 |  |  |  |  |
| 120000 | 0.00000246 | 0.516907508 |  |  |  |  |

**Curve:**

3b)**HOT POTATO ALGORITHM:**

**OUTPUT:**

The 32byte and 1k byte increment is shown below: 

* The first column represents the DATA SIZE.
* The second column shows the COMMUNICATION TIME.
* The third column shows the LATENCY.RESULTS:

FOR 32 byte increment(0 to 512 bytes):

|  |  |  |
| --- | --- | --- |
| **Data Size** | **Communication\_time** | **Latency** |
| 32 | 0.000095844 | inf |
| 64 | 0.000048041 | 48.04134369 |
| 96 | 0.000032028 | 24.02067184 |
| 128 | 0.000024021 | 16.01378123 |
| 160 | 0.000019789 | 12.36796379 |
| 192 | 0.000016332 | 9.799003601 |
| 224 | 0.000014441 | 8.424123128 |
| 256 | 0.000012249 | 6.999288286 |
| 288 | 0.000015444 | 8.687376976 |
| 320 | 0.000011086 | 6.159146627 |
| 352 | 0.000009732 | 5.352497101 |
| 384 | 0.000008841 | 4.822557623 |
| 416 | 0.000007996 | 4.331270854 |
| 448 | 0.000009928 | 5.346078139 |
| 480 | 0.000006676 | 3.576278687 |
| 512 | 0.000006258 | 3.337860107 |
|  |  |  |

**CURVE:**

FOR 1k increment(1k to 128k bytes):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Data Size** | **Communication\_time** | **Latency** |  | **Data Size** | **Communication\_time** | **Latency** |
| 1000 | 0.000097036 | inf |  | 41000 | 0.000002564 | 1.314282 |
| 2000 | 0.000047922 | 47.9221344 |  | 42000 | 0.00000252 | 1.290949 |
| 3000 | 0.000032345 | 24.25909042 |  | 43000 | 0.000002445 | 1.251698 |
| 4000 | 0.000034511 | 23.00739288 |  | 44000 | 0.00000239 | 1.222588 |
| 5000 | 0.000019026 | 11.89112663 |  | 45000 | 0.000002914 | 1.490116 |
| 6000 | 0.000016491 | 9.894371033 |  | 46000 | 0.000002286 | 1.168251 |
| 7000 | 0.000014305 | 8.344650269 |  | 47000 | 0.000003323 | 1.697437 |
| 8000 | 0.000012249 | 6.999288286 |  | 48000 | 0.000002335 | 1.192093 |
| 9000 | 0.000011656 | 6.556510925 |  | 49000 | 0.000002126 | 1.085301 |
| 10000 | 0.000010514 | 5.841255188 |  | 50000 | 0.00000206 | 1.050988 |
| 11000 | 0.000009255 | 5.090236664 |  | 51000 | 0.000002922 | 1.490116 |
| 12000 | 0.000008742 | 4.768371582 |  | 52000 | 0.000002017 | 1.028472 |
| 13000 | 0.000008161 | 4.420677821 |  | 53000 | 0.000001948 | 0.992647 |
| 14000 | 0.000007425 | 3.998096173 |  | 54000 | 0.00000196 | 0.998659 |
| 15000 | 0.000006994 | 3.746577672 |  | 55000 | 0.000001998 | 1.017694 |
| 16000 | 0.000006497 | 3.465016683 |  | 56000 | 0.00000189 | 0.962344 |
| 17000 | 0.000007826 | 4.157423973 |  | 57000 | 0.000002439 | 1.241054 |
| 18000 | 0.000005881 | 3.113466151 |  | 58000 | 0.000001883 | 0.957857 |
| 19000 | 0.000005521 | 2.914004856 |  | 59000 | 0.000001778 | 0.904346 |
| 20000 | 0.000005257 | 2.766910352 |  | 60000 | 0.000001848 | 0.939531 |
| 21000 | 0.000004859 | 2.551078796 |  | 61000 | 0.000001735 | 0.882149 |
| 22000 | 0.000004682 | 2.452305385 |  | 62000 | 0.000001727 | 0.877459 |
| 23000 | 0.00000452 | 2.362511375 |  | 63000 | 0.00000224 | 1.138256 |
| 24000 | 0.000005662 | 2.954317176 |  | 64000 | 0.000001688 | 0.857172 |
| 25000 | 0.000004044 | 2.106030782 |  | 65000 | 0.000001676 | 0.851229 |
| 26000 | 0.000003961 | 2.059936523 |  | 66000 | 0.000001618 | 0.821627 |
| 27000 | 0.000003771 | 1.957783332 |  | 67000 | 0.000001566 | 0.794729 |
| 28000 | 0.000003644 | 1.889687997 |  | 68000 | 0.000001718 | 0.871829 |
| 29000 | 0.000003519 | 1.82219914 |  | 69000 | 0.000001583 | 0.80291 |
| 30000 | 0.000003393 | 1.755254022 |  | 70000 | 0.000001856 | 0.941581 |
| 31000 | 0.000004292 | 2.217292786 |  | 71000 | 0.000001491 | 0.756127 |
| 32000 | 0.000003345 | 1.726611968 |  | 72000 | 0.000002209 | 1.119896 |
| 33000 | 0.000003179 | 1.639127731 |  | 73000 | 0.000001493 | 0.756648 |
| 34000 | 0.000003177 | 1.636418429 |  | 74000 | 0.00000146 | 0.739751 |
| 35000 | 0.000003031 | 1.560239231 |  | 75000 | 0.00000144 | 0.729754 |
| 36000 | 0.000002947 | 1.515660967 |  | 76000 | 0.000001735 | 0.87897 |
| 37000 | 0.000002835 | 1.457002428 |  | 77000 | 0.00000139 | 0.704276 |
| 38000 | 0.000003495 | 1.794583089 |  | 78000 | 0.000001397 | 0.707515 |
| 39000 | 0.00000261 | 1.339535964 |  | 79000 | 0.000001382 | 0.699972 |
| 40000 | 0.000002646 | 1.357151912 |  | 80000 | 0.000001386 | 0.701675 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Data Size** | **Communication\_time** | **Latency** |  | **Data Size** | **Communication\_time** | **Latency** |
| 81000 | 0.000001372 | 0.694394112 |  | 121000 | 0.000001025 | 0.516574 |
| 82000 | 0.000001332 | 0.674047588 |  | 122000 | 0.000000967 | 0.487674 |
| 83000 | 0.00000276 | 1.397074723 |  | 123000 | 0.000001285 | 0.647834 |
| 84000 | 0.000001595 | 0.807176153 |  | 124000 | 0.000001025 | 0.516574 |
| 85000 | 0.000001248 | 0.631525403 |  | 125000 | 0.000000967 | 0.487412 |
| 86000 | 0.000001278 | 0.646535088 |  | 126000 | 0.000000944 | 0.475883 |
| 87000 | 0.000001726 | 0.873277354 |  | 127000 | 0.000000946 | 0.476837 |
| 88000 | 0.000001704 | 0.861869461 |  | 128000 | 0.000000905 | 0.456187 |
| 89000 | 0.000001224 | 0.619075515 |  |  |  |  |
| 90000 | 0.000001176 | 0.594707018 |  |  |  |  |
| 91000 | 0.000001197 | 0.605318281 |  |  |  |  |
| 92000 | 0.000001153 | 0.582946526 |  |  |  |  |
| 93000 | 0.000001131 | 0.571427138 |  |  |  |  |
| 94000 | 0.000001649 | 0.833183207 |  |  |  |  |
| 95000 | 0.000001252 | 0.632823782 |  |  |  |  |
| 96000 | 0.000001167 | 0.589772275 |  |  |  |  |
| 97000 | 0.000001165 | 0.588595867 |  |  |  |  |
| 98000 | 0.000001163 | 0.587443716 |  |  |  |  |
| 99000 | 0.000001161 | 0.586315077 |  |  |  |  |
| 100000 | 0.000001421 | 0.717664006 |  |  |  |  |
| 101000 | 0.000001147 | 0.579357147 |  |  |  |  |
| 102000 | 0.000001129 | 0.570080068 |  |  |  |  |
| 103000 | 0.000001116 | 0.563322329 |  |  |  |  |
| 104000 | 0.000001116 | 0.563640039 |  |  |  |  |
| 105000 | 0.000001106 | 0.558220423 |  |  |  |  |
| 106000 | 0.000001152 | 0.581287202 |  |  |  |  |
| 107000 | 0.000001466 | 0.739997288 |  |  |  |  |
| 108000 | 0.000001066 | 0.538112961 |  |  |  |  |
| 109000 | 0.000001054 | 0.532026644 |  |  |  |  |
| 110000 | 0.000001235 | 0.623388028 |  |  |  |  |
| 111000 | 0.000001117 | 0.563534823 |  |  |  |  |
| 112000 | 0.000001054 | 0.531608994 |  |  |  |  |
| 113000 | 0.000001487 | 0.750379903 |  |  |  |  |
| 114000 | 0.000001115 | 0.562288065 |  |  |  |  |
| 115000 | 0.000001078 | 0.543761672 |  |  |  |  |
| 116000 | 0.000001163 | 0.586717025 |  |  |  |  |
| 117000 | 0.000001119 | 0.564188793 |  |  |  |  |
| 118000 | 0.000001441 | 0.726463448 |  |  |  |  |
| 119000 | 0.000001032 | 0.520277832 |  |  |  |  |
| 120000 | 0.000001057 | 0.532935647 |  |  |  |  |

**CURVE:**

4**.SHIFT PROCEDURE IMPLEMENTATION :**

The shift procedure id implemented using 3 functions :

* SHIFTARRAY
* RIGHT\_SHIFT
* LEFT\_SHIFT

Using the shift factor entered by the user,left or right shift is determined.

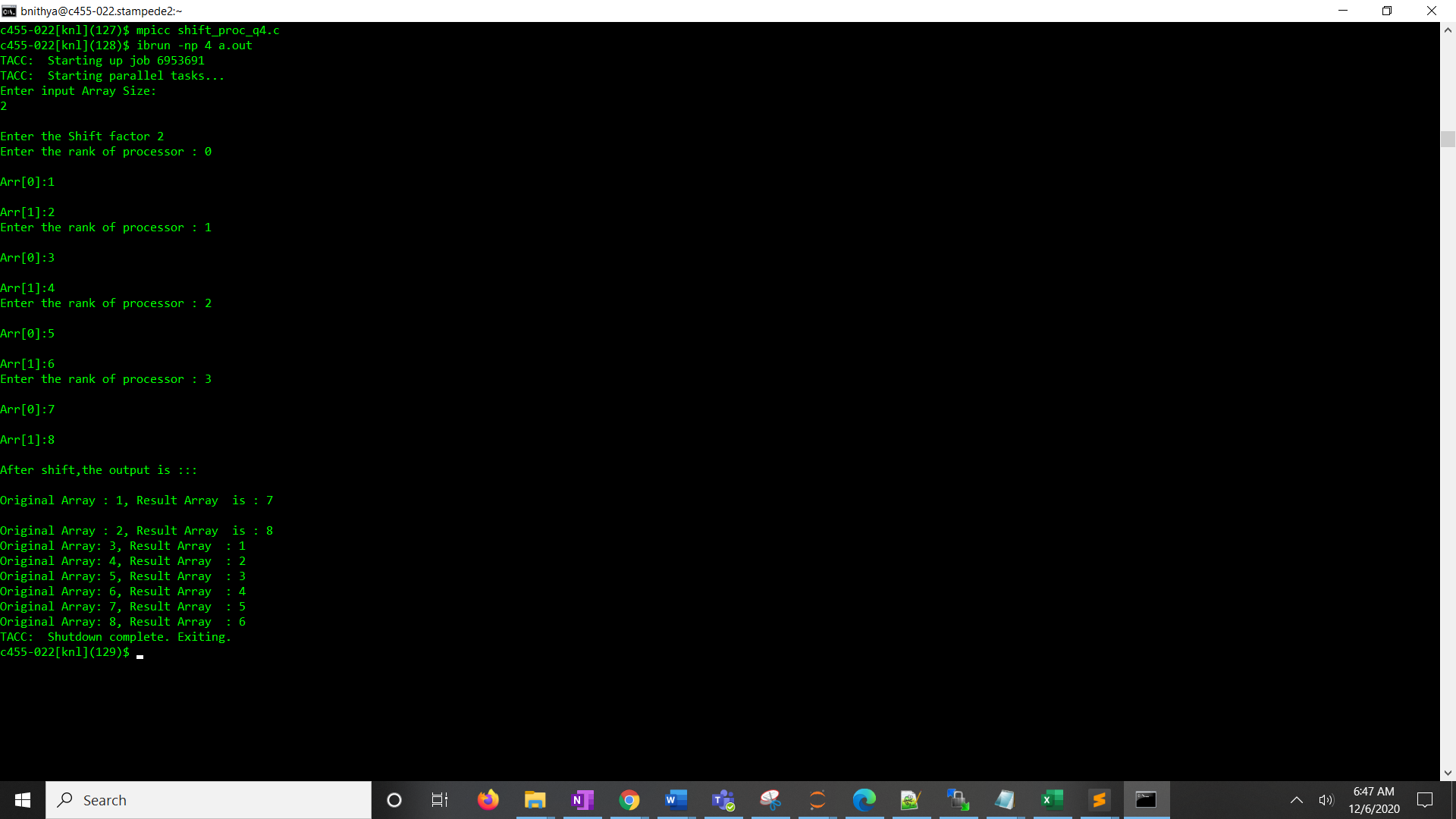
The output when shift factor is 2(positive) is shown below.

OUTPUT:

Assume ,

Size of the array is taken as 2.

No of Processors :4



The output of the array values before shift and after shift is shown in the screenshot .