# Assignment 3

Different parallel implementations of adding numbers

Three different parallel implementations are tried and are compared in the following report.

## System Settings

All tests were done on **Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz** processor. This computer is **dual core** where each core has **2 threads**. Also during each experiment it was insured that **no other applications** were running in the background, so that we don't have any biased readings.

## Experiments

### Implementation 1: Naive implementation to add n numbers.

1. $P_o$ sends n/p items to each processor.
2. Each PE sends n/p items.
3. Each PE sends a partial sum to $P_o$.
4. $P_o$ adds partial sum.

### Implementation 2: Recursive partitioning.

### Implementation 3: MPI API MPI_Reduce to compute the sum.

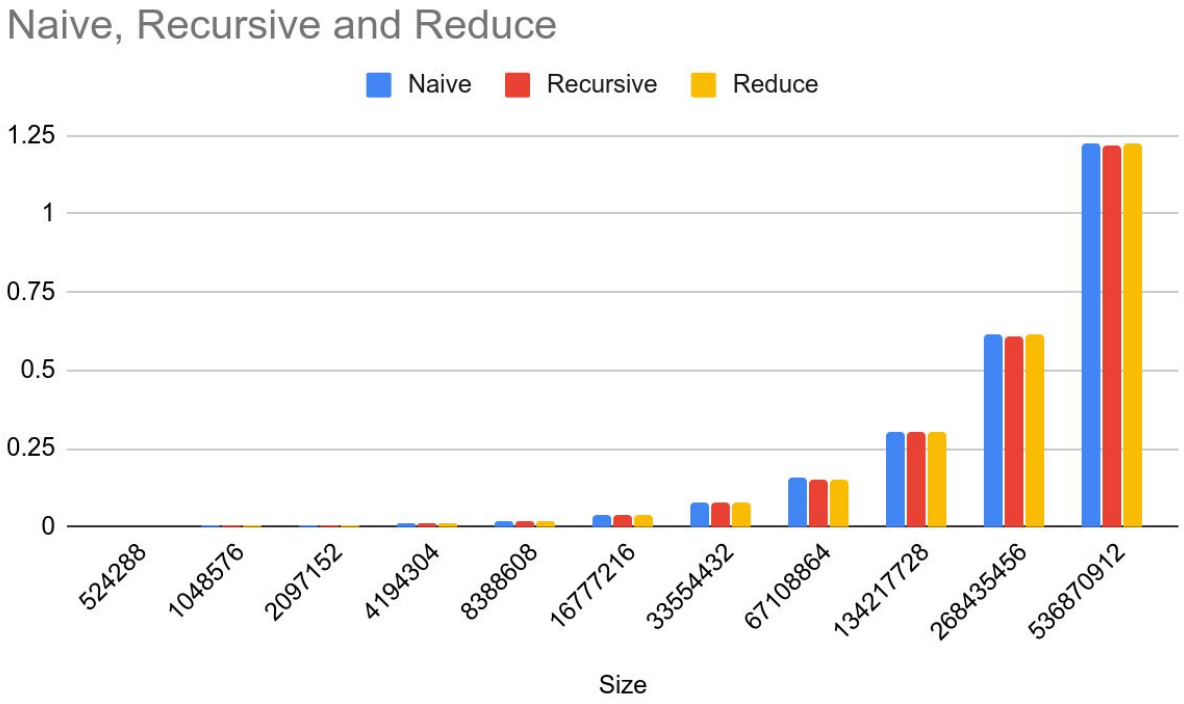
## Results

We will be varying input size along with the number of threads to compare. Input size varies from $\mathbf{2^{17}}$ to $\mathbf{2^{29}}$. And the number of threads will be varied as 2, 4, 8 and 16.

### N vs Time(in sec) using 2 Threads

| Size | Naive | Recursive | Reduce |
|---|---|---|---|
| 131072 | 0.000442 | 0.000582 | 0.000394 |
| 262144 | 0.000592 | 0.000799 | 0.000888 |
| 524288 | 0.001189 | 0.00172 | 0.001581 |

| 1048576 | 0.002407 | 0.002371 | 0.0024 |
|---|---|---|---|
| 2097152 | 0.004801 | 0.004765 | 0.004809 |
| 4194304 | 0.009596 | 0.009533 | 0.009625 |
| 8388608 | 0.01914 | 0.018964 | 0.019168 |
| 16777216 | 0.038331 | 0.03791 | 0.038284 |
| 33554432 | 0.076211 | 0.076376 | 0.076587 |
| 67108864 | 0.15474 | 0.151399 | 0.153904 |
| 134217728 | 0.304419 | 0.304786 | 0.306059 |
| 268435456 | 0.613292 | 0.608338 | 0.614904 |
| 536870912 | 1.225774 | 1.217968 | 1.224321 |

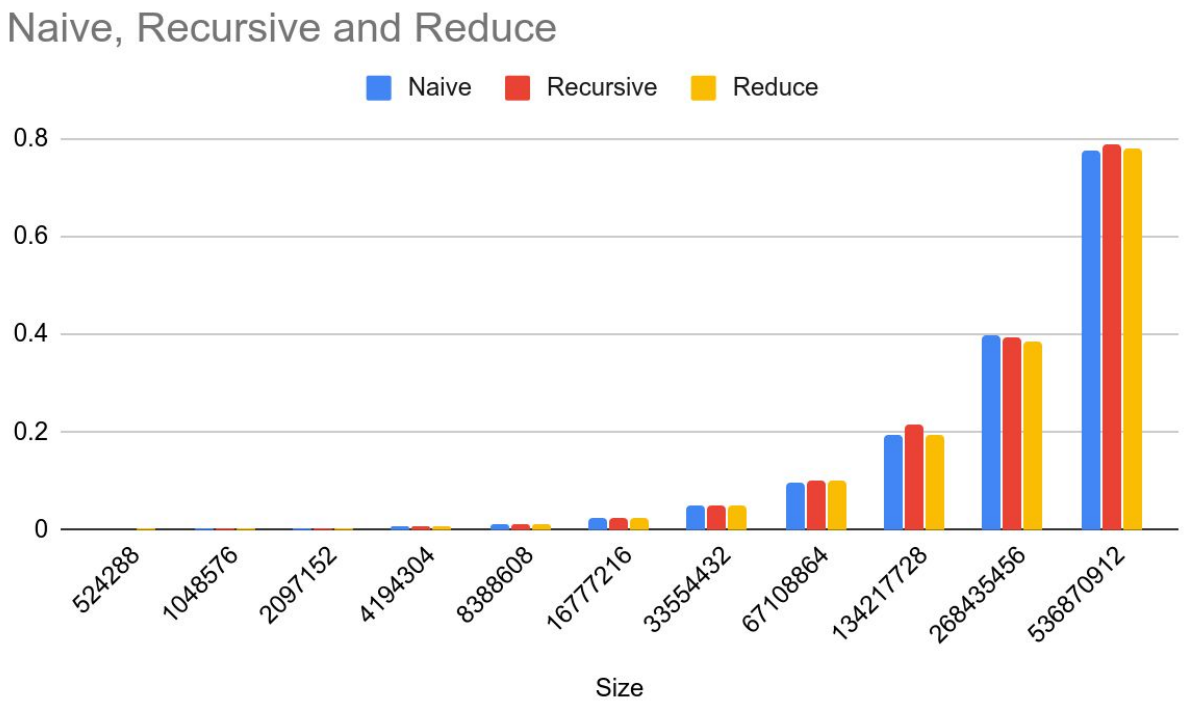



**Observations:**

We can observe that Naive implementation takes a little more time than the recursive and MPI_Reduce implementations which is as expected. But there is not much time difference between recursive and MPI_Reduce implementations.

## N vs Time(in sec) using 4 Threads

| Size | Naive | Recursive | Reduce |
|---|---|---|---|
| 131072 | 0.000218 | 0.000196 | 0.000246 |
| 262144 | 0.000382 | 0.000391 | 0.000381 |
| 524288 | 0.000747 | 0.000776 | 0.001436 |
| 1048576 | 0.001617 | 0.001625 | 0.001675 |
| 2097152 | 0.00318 | 0.003156 | 0.003078 |
| 4194304 | 0.006158 | 0.00621 | 0.006136 |
| 8388608 | 0.013901 | 0.012255 | 0.012296 |
| 16777216 | 0.024783 | 0.024572 | 0.024267 |
| 33554432 | 0.048763 | 0.051782 | 0.048477 |
| 67108864 | 0.097523 | 0.098864 | 0.102416 |
| 134217728 | 0.194858 | 0.214971 | 0.194483 |
| 268435456 | 0.399367 | 0.392984 | 0.386497 |
| 536870912 | 0.776501 | 0.786699 | 0.778035 |



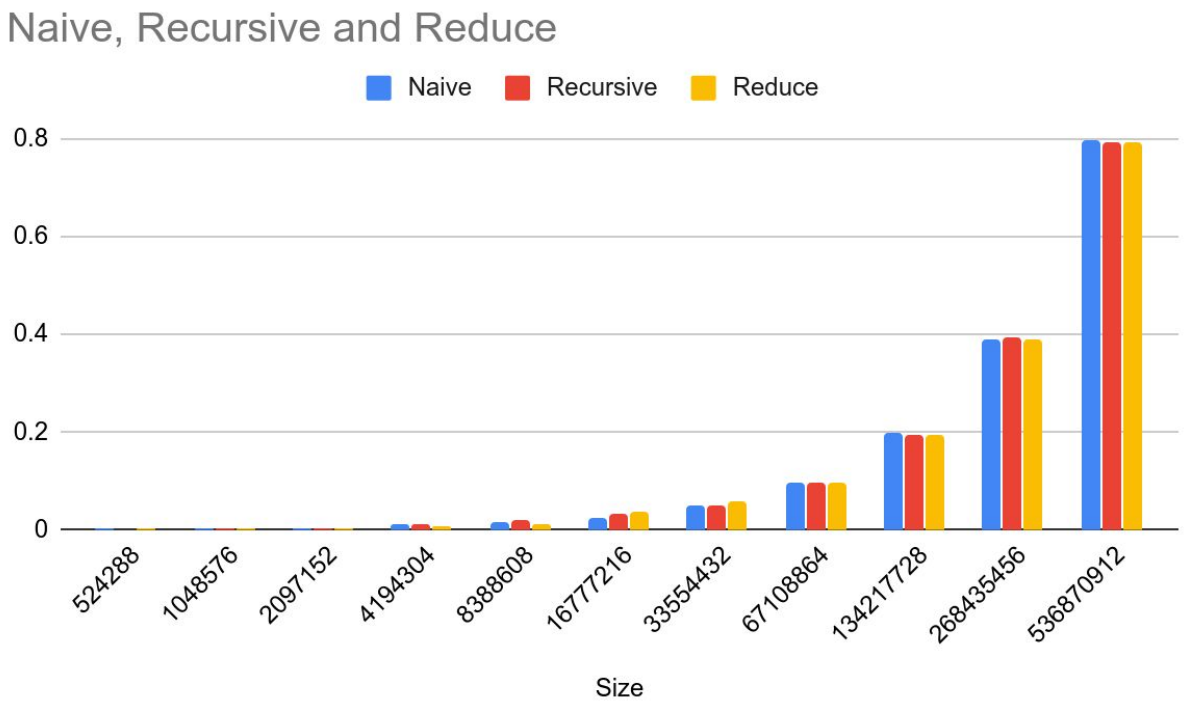

**Observations:**

No generic trend can be observed from the above graph. There are some n where we can see that recursive implementation takes a bit more time than other two implementations. Similarly there is no generic trend of time between naive and MPI_Reduce implementations.

## N vs Time(in sec) using 8 Threads

| Size | Naive | Recursive | Reduce |
|---|---|---|---|
| 131072 | 0.000102 | 0.00137 | 0.000394 |
| 262144 | 0.000914 | 0.000623 | 0.00068 |
| 524288 | 0.001251 | 0.001122 | 0.001327 |
| 1048576 | 0.00341 | 0.001684 | 0.002486 |
| 2097152 | 0.001646 | 0.002811 | 0.003469 |
| 4194304 | 0.009702 | 0.01351 | 0.009189 |
| 8388608 | 0.016235 | 0.019547 | 0.01283 |
| 16777216 | 0.025881 | 0.031611 | 0.036256 |

| | | | |
|---|---|---|---|
| 33554432 | 0.048998 | 0.048376 | 0.059087 |
| 67108864 | 0.096878 | 0.097278 | 0.096392 |
| 134217728 | 0.198106 | 0.195205 | 0.195036 |
| 268435456 | 0.388593 | 0.394827 | 0.389813 |
| 536870912 | 0.79536 | 0.793676 | 0.793187 |



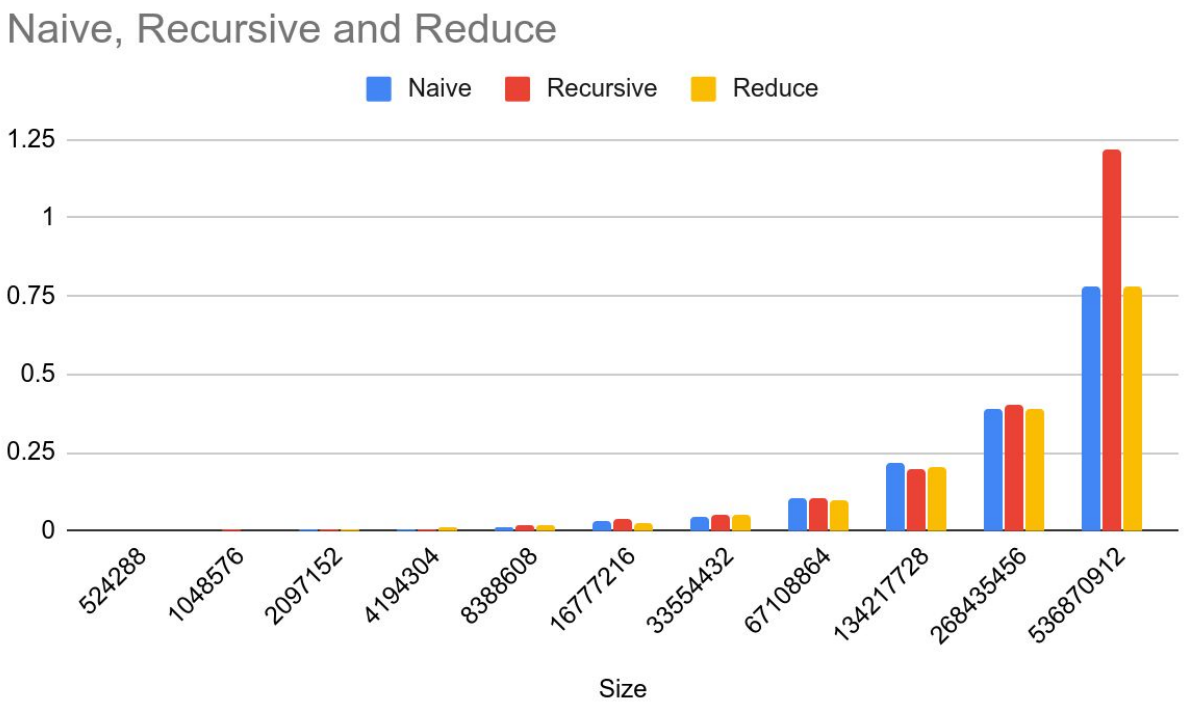

**Observations:**

We can clearly observe that Naive algorithm takes more time than the rest of the algorithms in most instances. And recursive and MPI_Reduce implementations take almost the same time but in many cases recursive implementation takes a bit more time.

## N vs Time(in sec) using 16 threads

| Size | Naive | Recursive | Reduce |
|---|---|---|---|
| 131072 | 0.000102 | 0.001001 | 0.000566 |
| 262144 | 0.000389 | 0.001769 | 0.001078 |

| | | | |
|---|---|---|---|
| 524288 | 0.000959 | 0.001442 | 0.000919 |
| 1048576 | 0.001598 | 0.003022 | 0.001563 |
| 2097152 | 0.002663 | 0.004238 | 0.002913 |
| 4194304 | 0.008427 | 0.006836 | 0.009124 |
| 8388608 | 0.013519 | 0.016652 | 0.018534 |
| 16777216 | 0.031474 | 0.037366 | 0.0281 |
| 33554432 | 0.048212 | 0.048622 | 0.048877 |
| 67108864 | 0.102587 | 0.104547 | 0.101035 |
| 134217728 | 0.21513 | 0.195225 | 0.203215 |
| 268435456 | 0.390403 | 0.404633 | 0.392726 |
| 536870912 | 0.779611 | 1.221056 | 0.779478 |

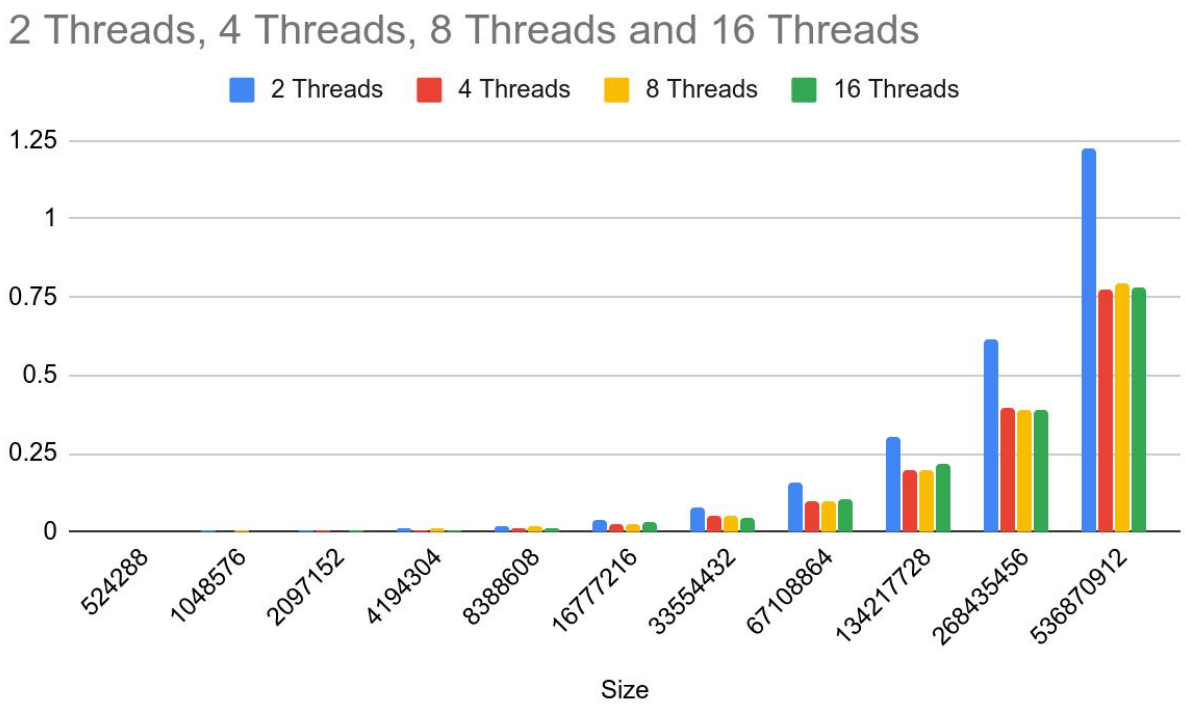



**Observations:**

Here recursive implementation takes more time than Naive and MPI_Reduce implementations which is not as expected. And in most cases, MPI_Reduce implementation takes less time than Naive implementations which is as expected.

## N vs No. of threads for Naive implementation

| Size | 2 Threads | 4 Threads | 8 Threads | 16 Threads |
|---|---|---|---|---|
| 131072 | 0.000442 | 0.000218 | 0.000102 | 0.000102 |
| 262144 | 0.000592 | 0.000382 | 0.000914 | 0.000389 |
| 524288 | 0.001189 | 0.000747 | 0.001251 | 0.000959 |
| 1048576 | 0.002407 | 0.001617 | 0.00341 | 0.001598 |
| 2097152 | 0.004801 | 0.00318 | 0.001646 | 0.002663 |
| 4194304 | 0.009596 | 0.006158 | 0.009702 | 0.008427 |
| 8388608 | 0.01914 | 0.013901 | 0.016235 | 0.013519 |
| 16777216 | 0.038331 | 0.024783 | 0.025881 | 0.031474 |
| 33554432 | 0.076211 | 0.048763 | 0.048998 | 0.048212 |
| 67108864 | 0.15474 | 0.097523 | 0.096878 | 0.102587 |
| 134217728 | 0.304419 | 0.194858 | 0.198106 | 0.21513 |
| 268435456 | 0.613292 | 0.399367 | 0.388593 | 0.390403 |
| 536870912 | 1.225774 | 0.776501 | 0.79536 | 0.779611 |




**Observations:**

The general trend is that time should decrease on increasing the number of processors which happens, but then time requires remains almost stagnant. This maybe because my processors support maximum 4 threads in parallel. Now, if we increase the number of threads beyond that then thread context switching time also comes into play and may increase the total time by a small amount. We have observed that after 4 threads, time hasn't decreased drastically which happened when we increased threads from 2 to 4.