CSCE 435 Group project – Team 24

Group members

1. Caleb Bynum
2. Josef Munduchirakal
3. Aazmir Lakhani
4. Tyler Roosth

Project topic

Implementing parallel sorting algorithms (mergeSort, sampleSort, enumerationSort, oddEvenTranspositionSort) using MPI and CUDA, and subsequently analyzing and comparing the respective performances.

Brief Project Description

We have implemented the following algorithms in different paradigms:

1. MergeSort
    - MPI
    - CUDA
2. SampleSort
    - MPI
    - CUDA
3. EnumerationSort
    - MPI
    - CUDA
4. OddEvenTranspositionSort
    - MPI
    - CUDA

Pseudocode

OddEven Sort in MPI:

Variables in main:
- p : number of processes
- global_n : number of elements in global array
- local_n : number of elements in processes' local array
- local_A : each processes' local array

Main Procedure:
1. Perform MPI initialization functions
2. Allocate local_A
3. Populate local_A
4. Call Sort on each process
5. MPI Finalize

Interpreting the sorted array:
The global array can be formed by concatenating each local array in ascending order by process rank. In MPI, this can be done with MPI_Gather()

Variables in Sort:
- my_rank : the current processes' rank
- comm : MPI_COMM_WORLD
- phase : counter to track current phase, ranges from 0 to p
- temp_B : temporary local storage for MPI_Sendrecv
- temp_C : temporary local storage for Merge_high or Merge_low
- local_n : Same as in the Main Procedure
- local_A : Same as in the Main Procedure
- p : Same as in the Main Procedue
- even_partner : the process that will be communicated with during the even phase
- odd_partner : the process that will be communicated with during the odd phase

Sort Procedure:
1. Allocate local storage temp_B
2. Allocate local storage temp_C
3. Determine even_phase_partner
4. Determine odd_phase_partner
5. Call quickSort on local_A
6. For each phase:
7. Call Odd-Even-Iterate
8. Deallocate temp_B
9. Deallocate temp_C

Variables in Odd-Even-Iterate:

my_rank : the current processes' rank
comm : MPI_COMM_WORLD
status : MPI_status
phase : counter to track current phase, ranges from 0 to p
temp_B : temporary local storage for MPI_Sendrecv
temp_C : temporary local storage for Merge_high or Merge_low
local_n : Same as in the Main Procedure
local_A : Same as in the Main Procedure
p : Same as in the Main Procedure
even_partner : the process that will be communicated with during the even phase
odd_partner : the process that will be communicated with during the odd phase

Odd-Even-Iterate Procedure:

1. if phase is even:
2. Call MPI_Sendrecv to exchange values with even_partner
3. if my_rank is odd:
4. Call Merge_high
5. else:
6. Call Merge_low
7. else:
8. Call MPI_Sendrecv to exchange values with odd_partner
9. if my_rank is odd:
10. call Merge_low
11. else:
12. call Merge_high

Merge_high Variables:

local_A : local array for a process
temp_B : contains array sent from partner process
temp_C : to be filled with the larger half of values from local_A and temp_B
A_ptr : iterator for local_A
B_ptr : iterator for temp_B
C_ptr : iterator for temp_C

Merge_high Procedure:

1. Initialize A_ptr, B_ptr, C_ptr to point at the end of their respective arrays
2. While C_ptr >= 0:
3. if local_A[A_ptr] >= temp_B[B_ptr]:
4. temp_C[C_ptr] = local_A[A_ptr]
5. decrement C_ptr, A_ptr
6. else:
7. temp_C[C_ptr] = temp_B[B_ptr]
8. decrement C_ptr, B_ptr

Merge_low Variables:
local_A : local array for a process
temp_B : contains array sent from partner process
temp_C : to be filled with the larger half of values from local_A and temp_B
A_ptr : iterator for local_A
B_ptr : iterator for temp_B
C_ptr : iterator for temp_C

Merge_low Procedure:
1. Initialize A_ptr, B_ptr, C_ptr to 0
2. While C_ptr < local_n:
3. if local_A[A_ptr] <= temp_B[B_ptr]:
4. temp_C[C_ptr] = local_A[A_ptr]
5. increment C_ptr, A_ptr
6. else:
7. temp_C[C_ptr] = temp_B[B_ptr]
8. increment C_ptr, B_ptr

OddEven Sort in CUDA

Variables:
n : number of elements to sorted
t : number of threads
A : array to be sorted

Idea: avoid coordination between threads by launching kernels for each phase (odd or even) and subsequently synchronizing CUDA threads after each phase.

Main Procedure:
1. Copy A to GPU memory
2. for each phase in {0, 1, ..., n-1}:
3. if phase is even:
4. launch EvenPhase kernel
5. else:
6. launch OddPhase kernel
7. synchronize device
8. Copy A to device memory

EvenPhase Kernel:
1. Id = threadId + blockDim * blockId
2. index1 = 2 * Id
3. index2 = index1 + 1
4. compare and swap array elements at index1, index2

OddPhase Kernel:

1. Id = threadId + blockDim * blockId
2. index1 = 2 * Id + 1
3. index2 = index1 + 1
4. if thread is not the last thread
5. compare and swap array elements at index1, index2

EnumerationSort MPI

MPI Implementation Pseudocode

Define variables needed:

- Numprocs - Number of processes
- Myid - Process ID
- reg_x, reg_y, reg_z - Register variables for data exchange
- compare - Counter for comparisons
- z_count - Count of elements in reg_z
- N - Size of data
- T - Number of threads
- a, b, c, d - Data arrays
- size - Size of data arrays

Main Procedure:

1. User input
2. Allocate the necessary memory for each array being used
3. Generate random numbers to save to a file
4. If myid is 0
   - Distribute the numbers among the processors
   - Send numbers to reg_x
   - Send reg_y to next process
5. For all other processes:
   - Receive reg_x from process 0
   - Iterate through each processor
6. Swap reg_z values:
   - Set values
   - Broadcast the values using MPI
   - Send and receive reg_x and reg_z
7. Print the result for process 0
8. If a process is not 0:
   - Send and receive reg_z and z_count
9. Print time

EnumerationSort CUDA

CUDA Implementation Pseudocode

Define variables needed

- N - Array size
- B - Blocks
- T - Number of threads per block
- a, b, c, d - Arrays for data
- dev_a, dev_b, dev_c - Device memory for data
- size - Size of data arrays
- start, stop - CUDA events for timing

Main Procedure:

1.Import libraries
2.Initialize CUDA
3.User Input
4.Use user input to initialize CUDA grid and block dimensions
5.Allocate the necessary memory for each array being used
6.Generate numbers and populate array
7. Record start time
8. Use a CUDA kernel to parallel sort
9. Record end time
10. Perform sequential sorting to compare GPU results
11. Print the time for GPU and CPU execution.
12. Compare GPU and CPU

MergeSort MPI:

MPI Implementation Pseudocode

```
Declare a variable to track if the sorting check passed.

Define the CUDA kernel 'gpu_merge_sort' to merge subarrays:
    Determine the thread's portion of the array to work on.
    Merge the two sorted subarrays into one sorted array.

Define the 'merge_sort' function to sort the array using GPU:
    Allocate memory on the GPU.
    Copy the array from the host to the GPU.
    Perform merge sort in a loop, doubling the width each time:
        Launch the 'gpu_merge_sort' kernel with appropriate blocks and threads.
        Swap the pointers to work on the latest sorted data.
    Copy the sorted array back to the host.
    Free the GPU memory.

Define the 'main' function:
    Initialize Caliper for profiling.
```

Check for the correct command line argument for THREADS_PER_BLOCK.
Initialize an array with random integers.
Print the array before sorting.
Sort the array using 'merge_sort'.
Print the sorted array.
Check if the array is sorted and print the result.
Clean up and finalize Caliper.
Return from the program.

MergeSort CUDA:

CUDA Implementation Pseudocode
Declare a variable to track if the sorting check passed.

Define the CUDA kernel 'gpu_merge_sort' to merge subarrays:
Determine the thread's portion of the array to work on.
Merge the two sorted subarrays into one sorted array.

Define the 'merge_sort' function to sort the array using GPU:
Allocate memory on the GPU.
Copy the array from the host to the GPU.
Perform merge sort in a loop, doubling the width each time:
Launch the 'gpu_merge_sort' kernel with appropriate blocks and threads.
Swap the pointers to work on the latest sorted data.
Copy the sorted array back to the host.
Free the GPU memory.

Define the 'main' function:
Initialize Caliper for profiling.
Check for correct command line argument for THREADS_PER_BLOCK.
Initialize an array with random integers.
Print the array before sorting.
Sort the array using 'merge_sort'.
Print the sorted array.
Check if the array is sorted and print the result.
Clean up and finalize Caliper.
Return from the program.

Sample Sort MPI:

Variables in main:

Pivots: Array of random double to be sorted. Bucket_Num: Array to track the number of values that are being sent to each processor Displacements: Storage for displacement data for scatter and gather functionality D_list: final doubles list to store the sorted value

Main Function Logic:

1. MPI_Init, Size, and Rank to set up MPI parallel functionality
2. Create all the random values in the main thread only
3. Set the displacements and sort each value into their respective buckets in the main thread only
4. Scatter, Sort, and Gather each individual bucket in their own thread

Compare Function Logic:

This is a customer comparator function that is designed to help the custom_qsort function sort through doubles.

1. Cast and dereference the two input values
2. Subtract the two values
3. If the first is greater than the second return 1
4. If the second is greater than the first return -1
5. If both values are equal return 0

Custom Quicksort Function Logic:

This function simply is a wrapper for the stl qsort function. The only modification to the stl function is the use of the custom comparator mentioned above in the compare function logic.

Find Bucket Logic:

In order to complete the sample sort operation it is important to sort values into buckets for each processor.

Loop through every processor

Check if the current input value is lower than the upper limit of each processor
Once a processor is found return the index number of the processor

Sample Sort Cuda:

Main Function Logic:

1. Create vector of random floats
2. Call bucket sort function
3. Print out the sorted vector

Bucket Sort Logic:

1. Create arrays for data and bucket value counters
2. Call the bucket_sort_kernel function and synchronize
3. Use thrust sort to sort each individual bucket
4. Free all the previously created data structures for data and bucket value counters

Bucket Sort Kernel Logic:

1. Check if the current value being passed to the function is less than the max bucket values
2. Modify the bucket that we add the value to... If the current bucket is within the bound of the number of buckets we can add it if not we must go back a bucket

3. Add the value to the appropriate bucket

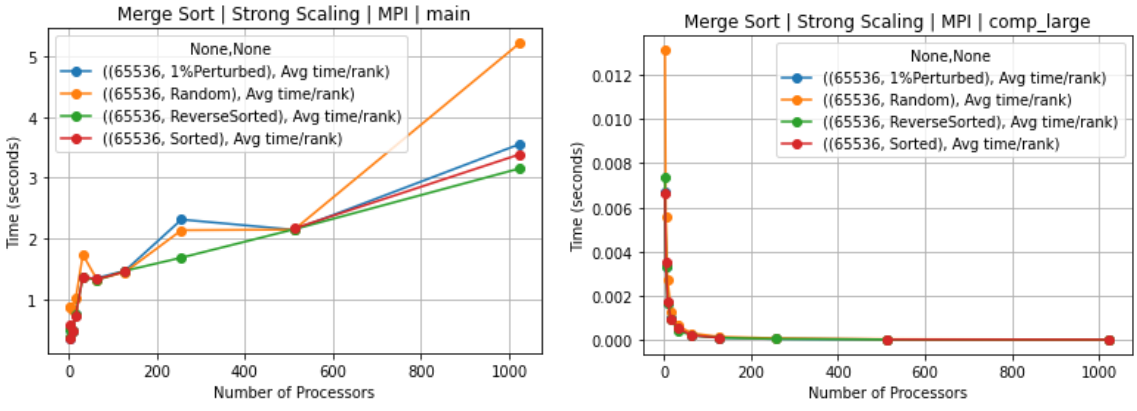
Evaluation Plan

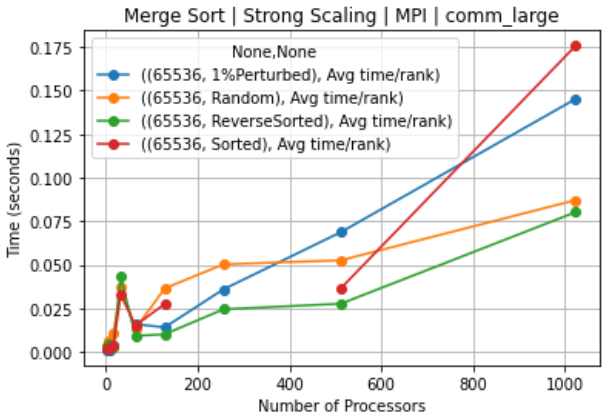
– For each algorithm, we plan to measure the runtime of various components with our programs with respect to input size.
– In testing, we plan to vary the array input sizes with the following counts {2^16, 2^20, 2^24}.
– Input properties of float arrays to be tested: { randomized, reversed, sorted }.
– We plan to evaluate how our algorithms behave with respect to Strong Scaling (same problem size {2^24}, increase number of processors/nodes)
– MPI Strong Scaling: increase number of cores {2, 4, 8, 16, 32, 64}
– CUDA Strong Scaling: number of threads per block {64, 128, 512, 1024}
– Weak scaling (increase problem size, increase number of processors)
– MPI Weak Scaling:
– Increase number of cores {2, 4, 8, 16, 32, 64}
– Increase input array size {2^4, 2^8, 2^12, 2^16, 2^20, 2^24}
– CUDA Weak Scaling:
– Increase number of threads per block {64, 128, 512, 1024}
– Increase input array size {2^12, 2^16, 2^20, 2^24}

Performance Evaluation

Merge Sort MPI:

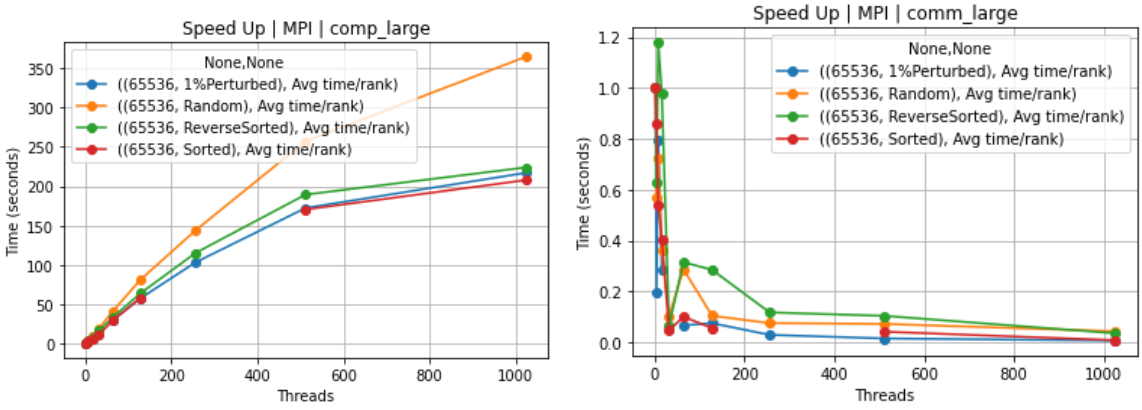
Strong Scaling 65536

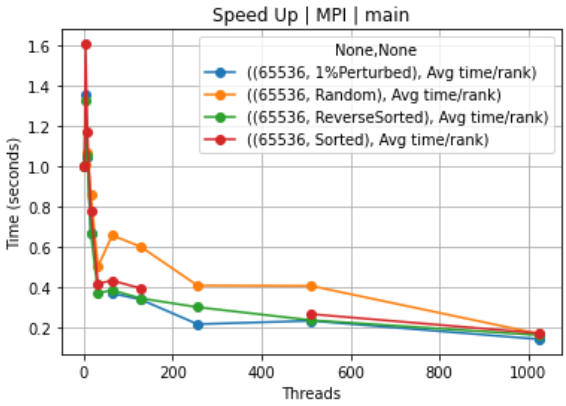




You can see at 65536 comp_large has quite a big drop off at the beginning and starts to straighten out at a higher number of processes. But while with comm_large, the time it takes seems to only increase over time most likely making it the overhead for main.

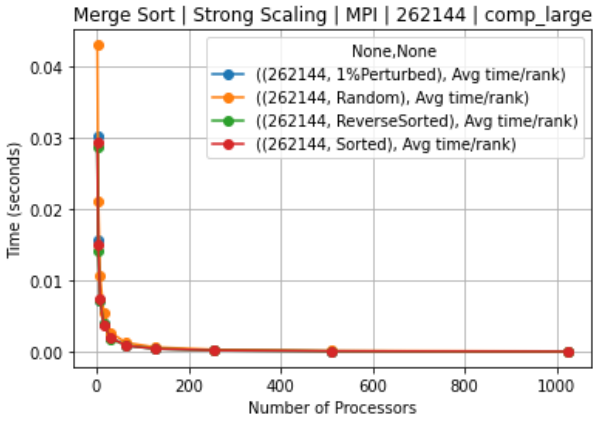
Strong Scaling 65536 Speed Up

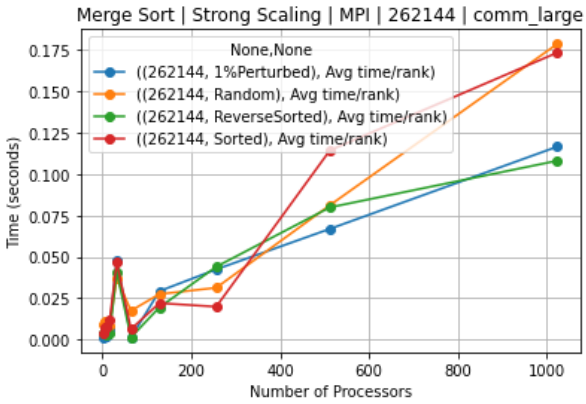


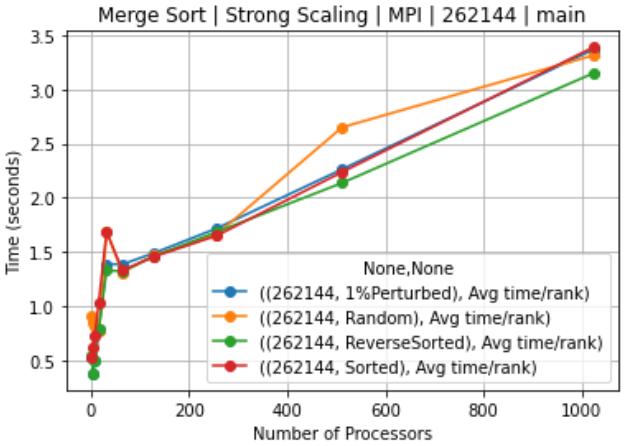


The speed up here shows that the comp_large experiences quite a bit of speed up while the comm_large does not which is fairly straight foward.

Strong Scaling 262144

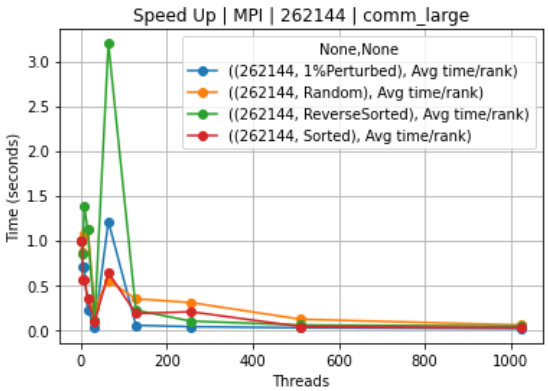


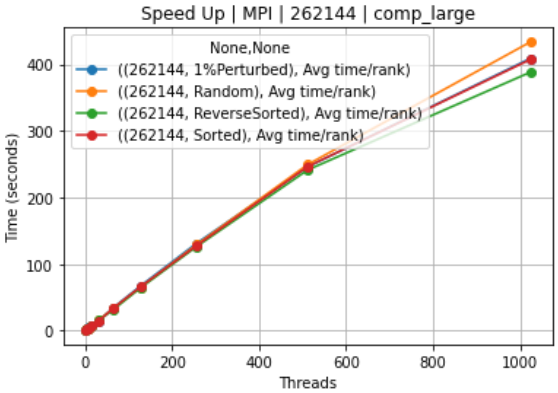


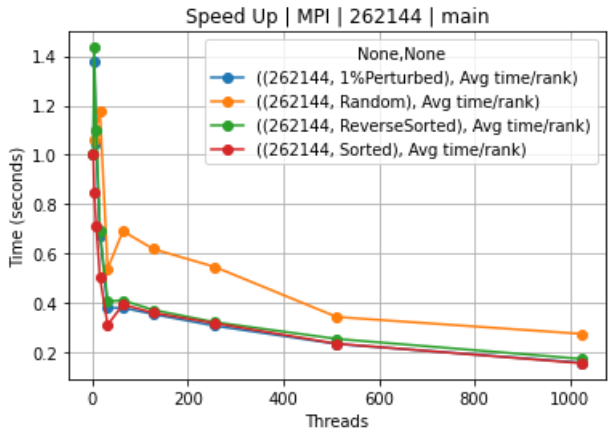


This continues to follow the trend from 65536 in that comp_large has a drop off over time, and comm_large continues to take a majority of the time.

Strong Scaling 262144 Speed Up

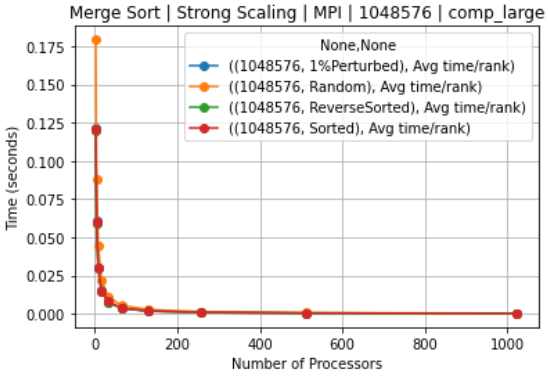


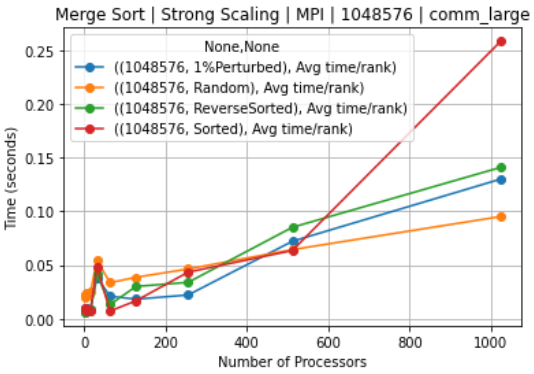


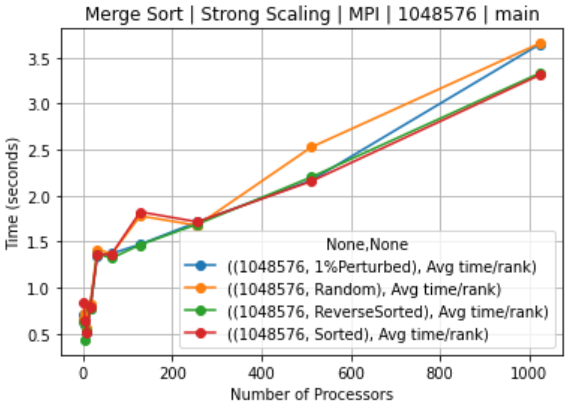


The speed up here is also similar to 65536 so there isn't much to say which may be a trend with this MergeSort.

Strong Scaling 1048576

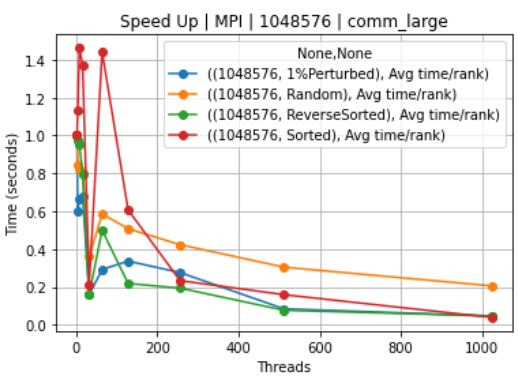


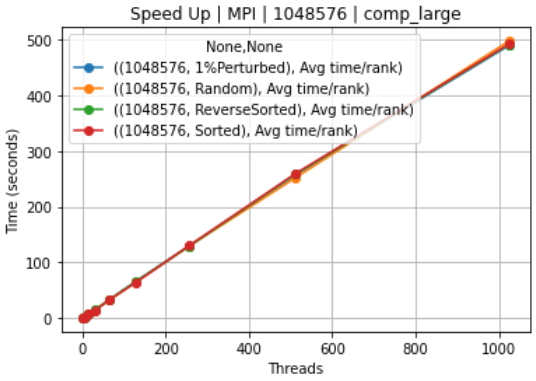


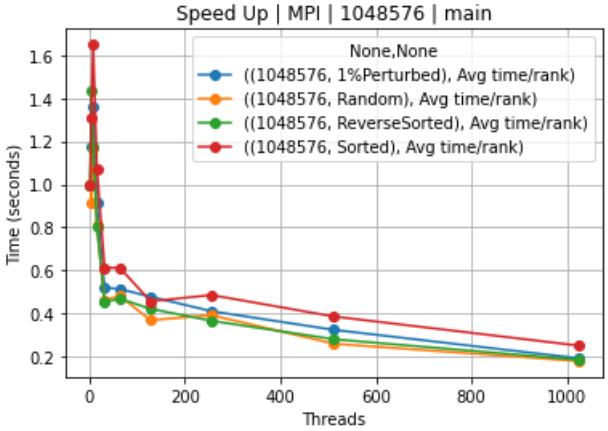


Once again following the same trends as before.

Strong Scaling 1048576 Speed Up

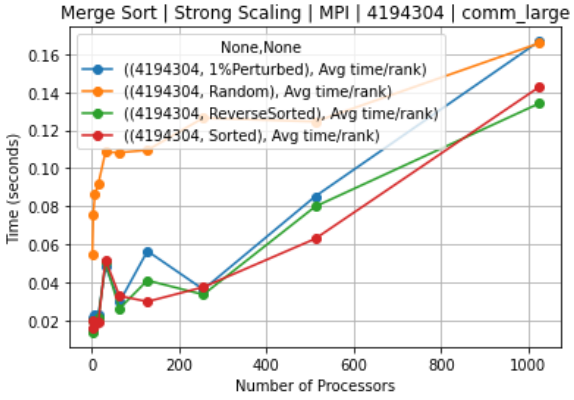


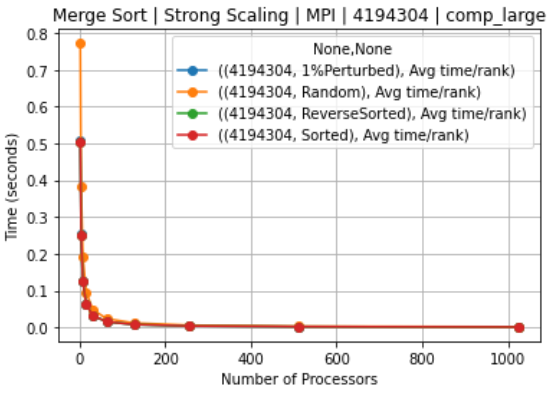


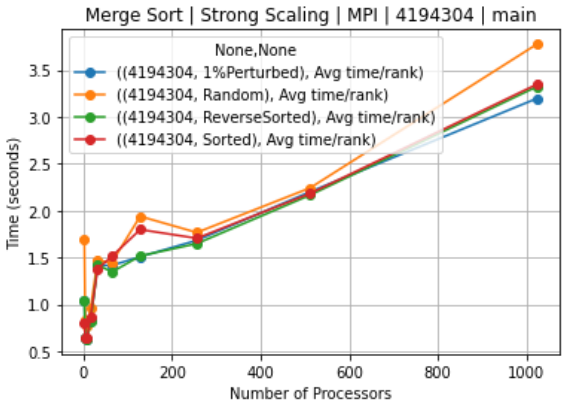


The speed up also seems very similar to the previous input sizes with more jumps at the beginning in the comm_large.

Strong Scaling 4194304

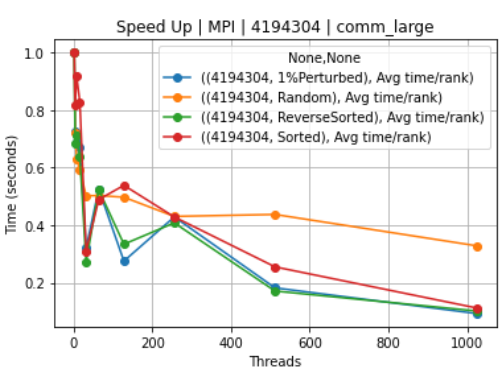


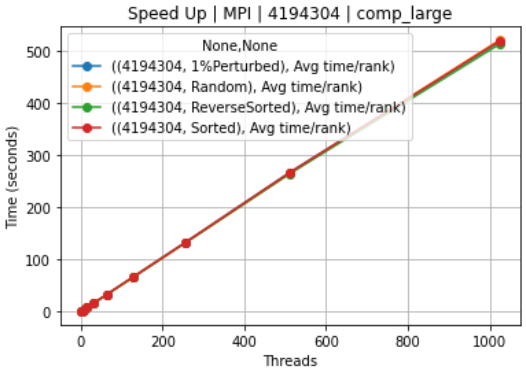


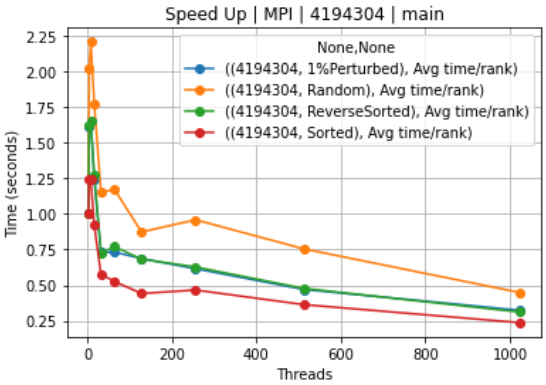


Still very similar to the previous input sizes which is very interesting since it seems to continue following this trend.

Strong Scaling 4194304 Speed Up

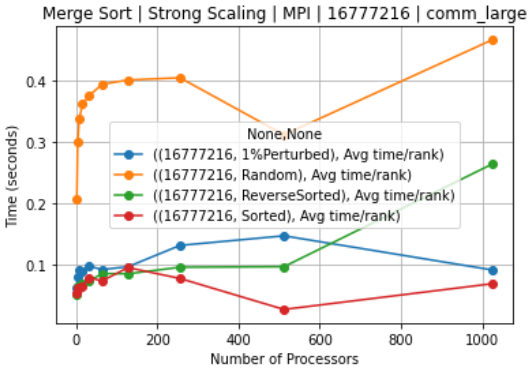


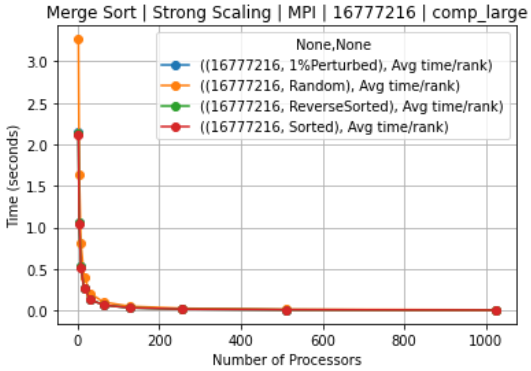


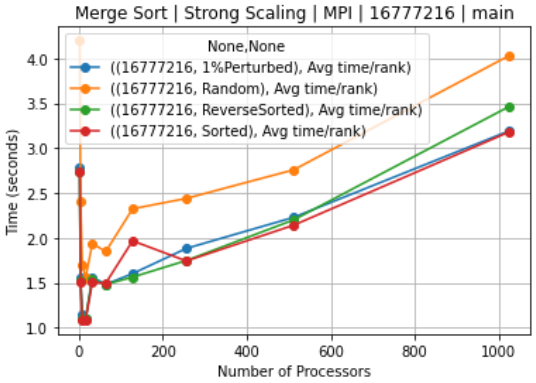


Again, very similar to the other speed ups with comp_large, speeding up substantially, but comm_large decreasing as the processors increase.

Strong Scaling 16777216

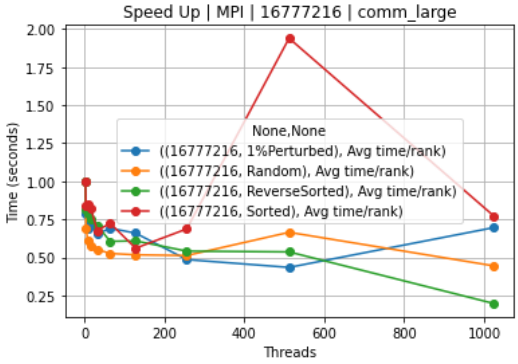


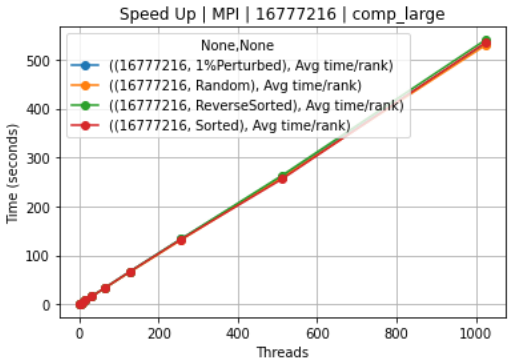


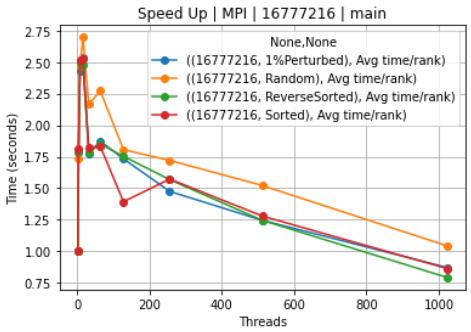


Here while it looks similar, the comm_large, specifically the Random input type, has an interesting curve to it that isn't seen previously.

Strong Scaling 16777216 Speed Up

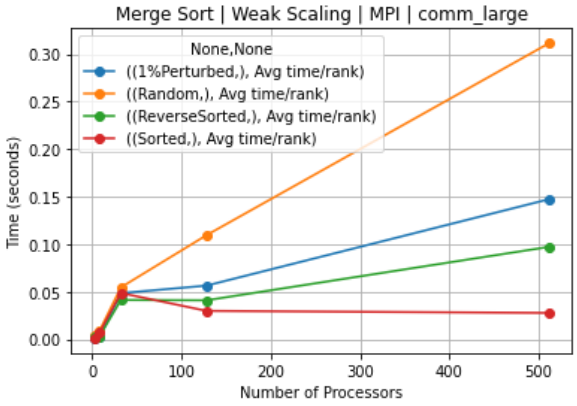


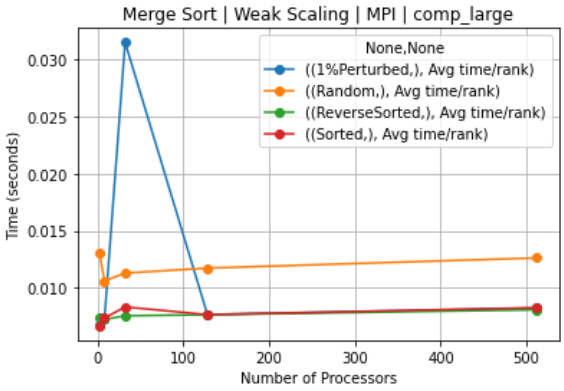


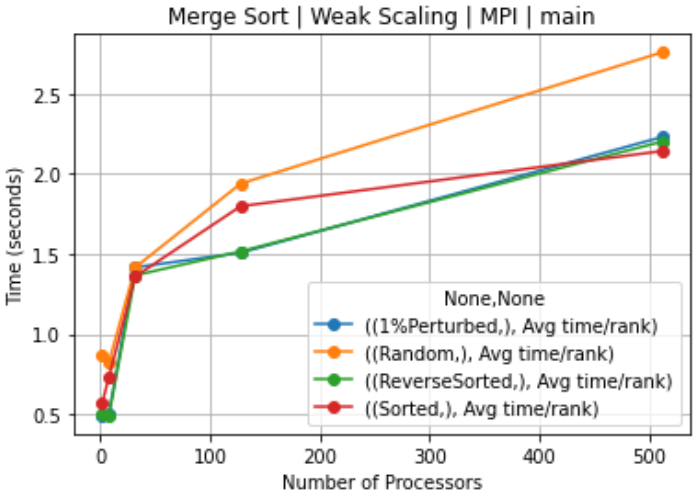


And here it is a little different as well in that there is a moment of speed up for the sorted input type in the comm_large that hasn't been seen before.

Weak Scaling



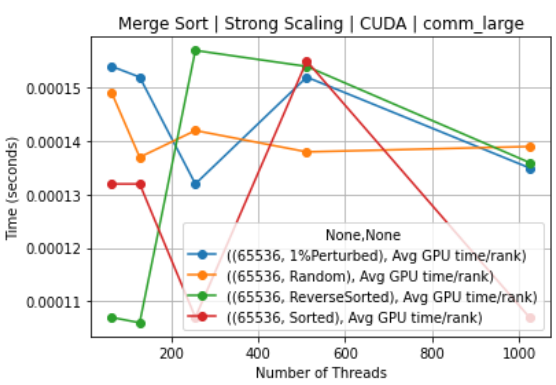


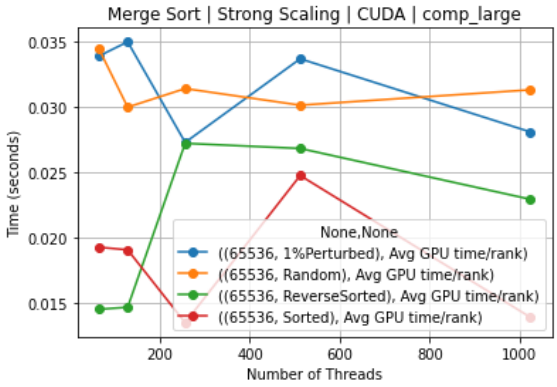


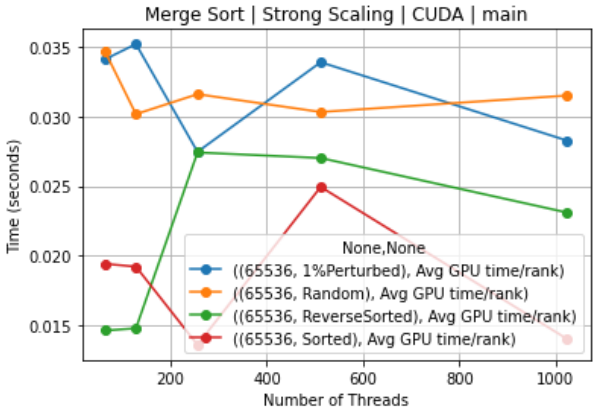
In the Weak Scaling it can be seen that the comp_large graph tapers off into a somewhat straight line as the processors increase but the comm_large continues to take more and more time especially for the Random input type.

Merge Sort CUDA:

Strong Scaling 65536

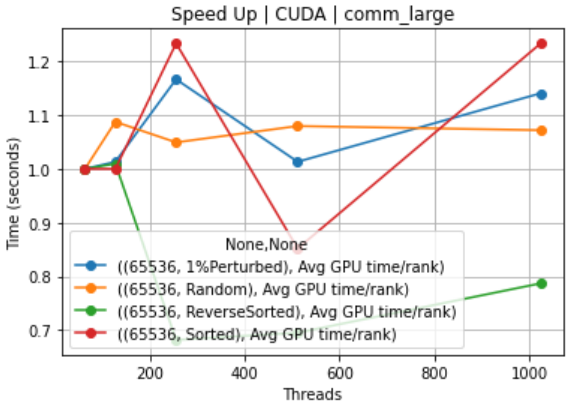


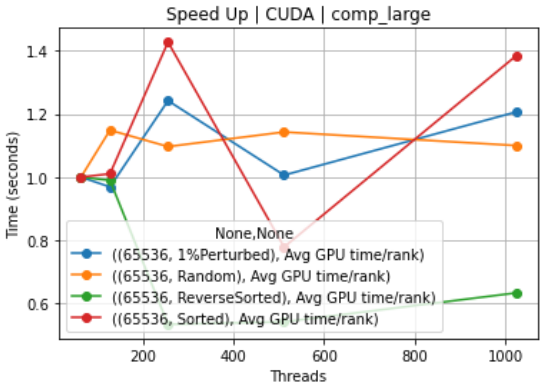


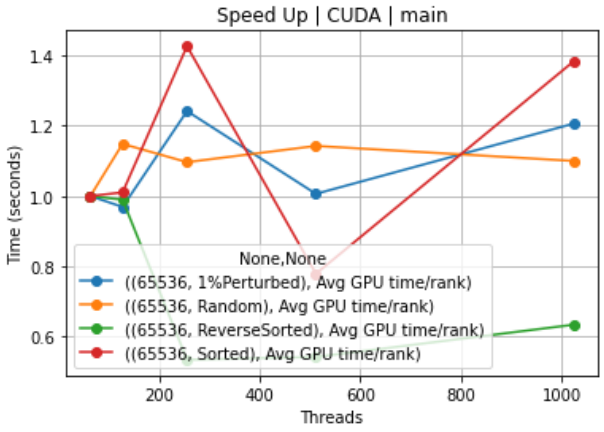


Here, the lines are all over the place with ups and downs in both comp_large and comm_large with the Random and 1%Perturbed input types seemingly taking the most time which makes sense.

Strong Scaling 65536 Speed Up

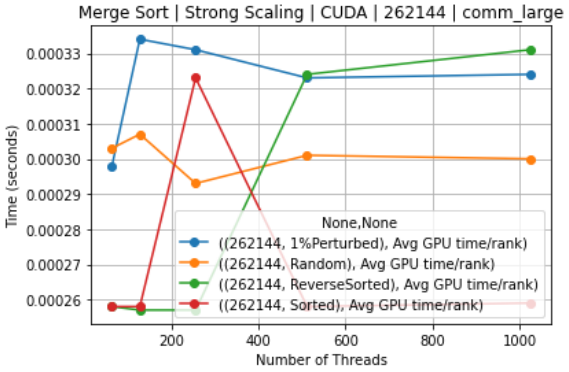


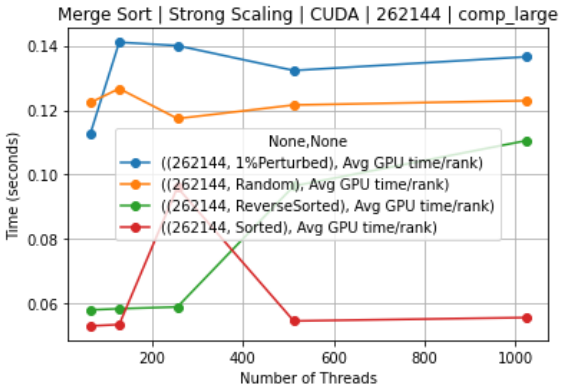


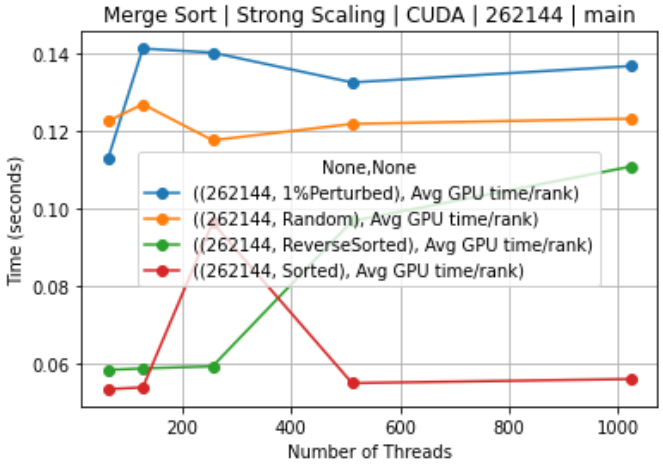


The speed up is also similar in that it goes up and down quite a bit with speeds up from one processor to the next while also speeding down a bit as well.

Strong Scaling 262144

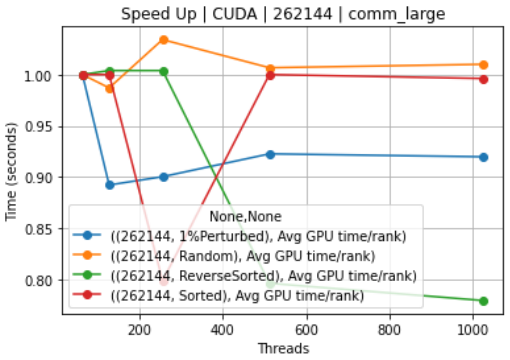


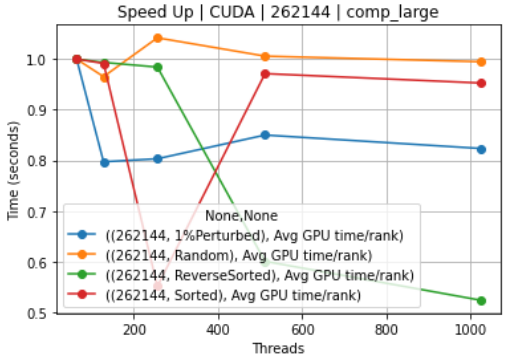


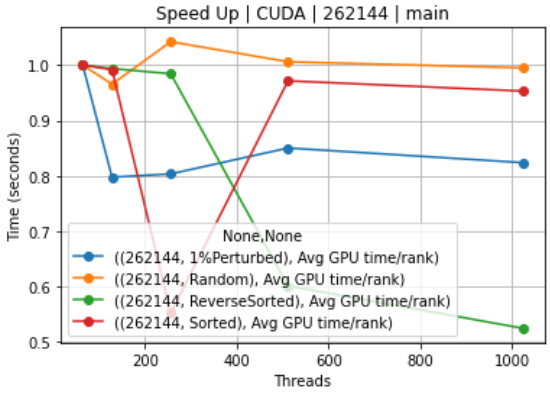


This looks more calm compared to 65536 in that the lines have less chaotic trends but overall seem to straighten out in the end.

Strong Scaling 262144 Speed Up

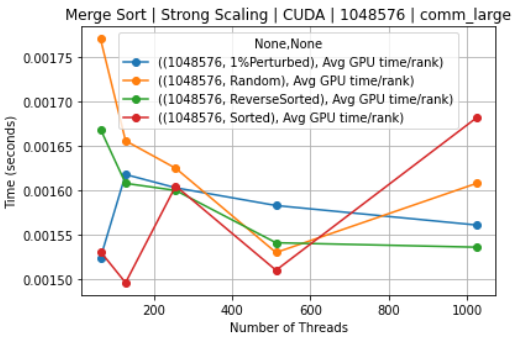


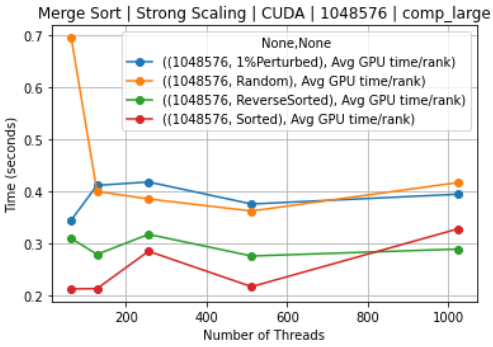


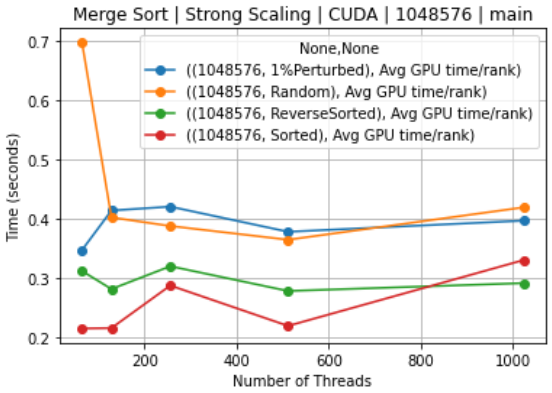


The speed up doesn't seem to show anything substantial with little to no speed up at all.

Strong Scaling 1048576

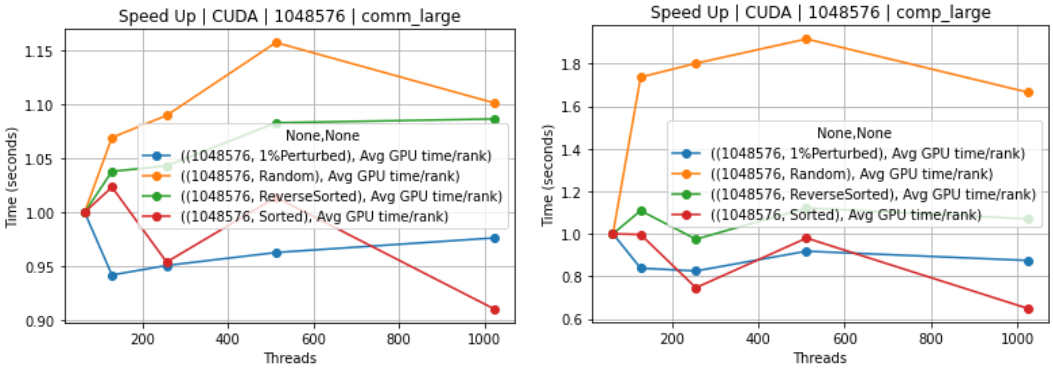


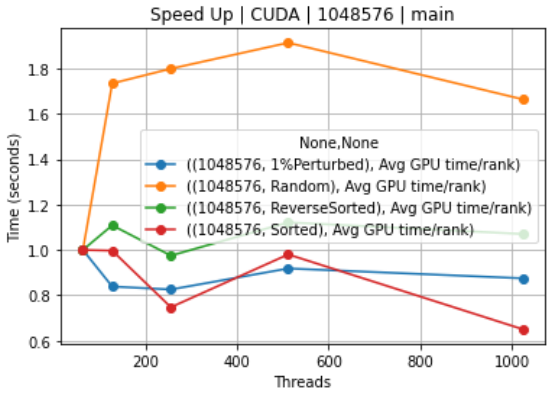




Here we actually start to see the graphs move down a bit more than before and seem to have a pattern decreasing for the most part.

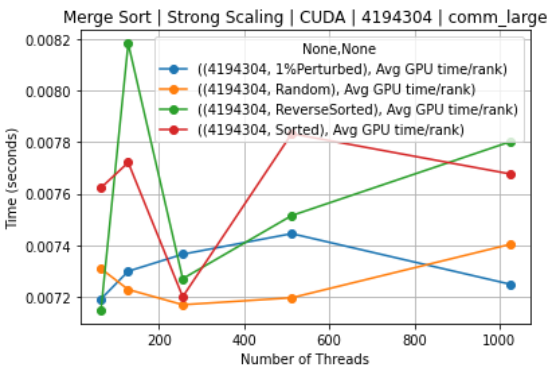
Strong Scaling 1048576 Speed Up

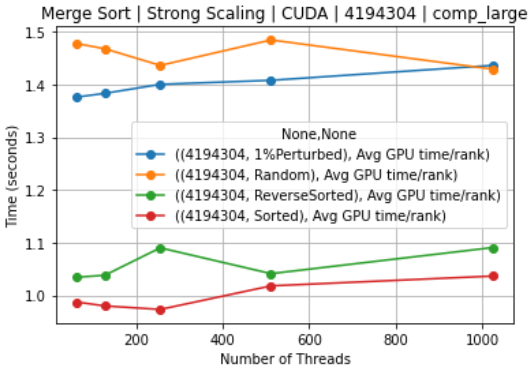


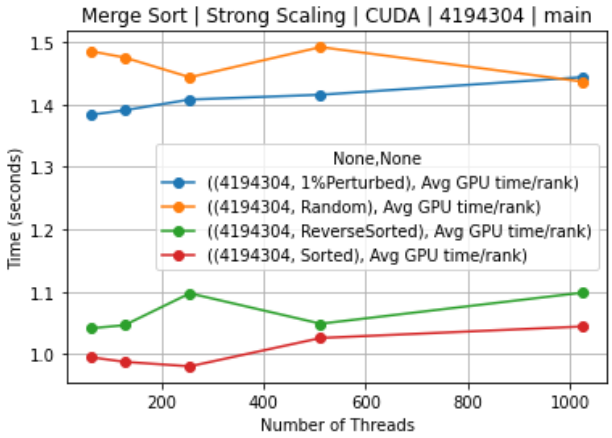


The speed up also reflects that in the comp_large almost hitting 2x speed up for the Random input type for comp_large, and a very little speed up for Random in comm_large.

Strong Scaling 4194304

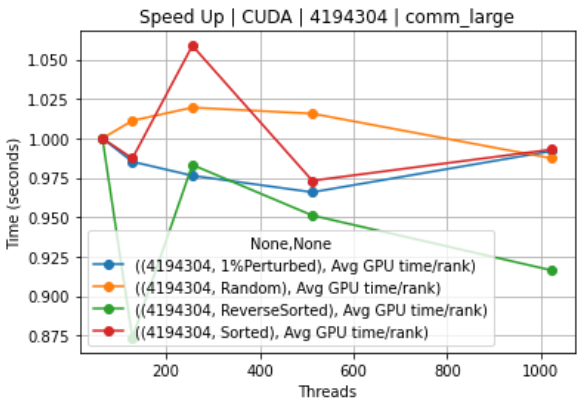


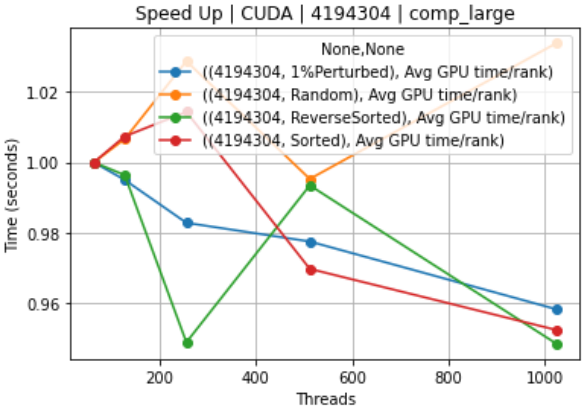


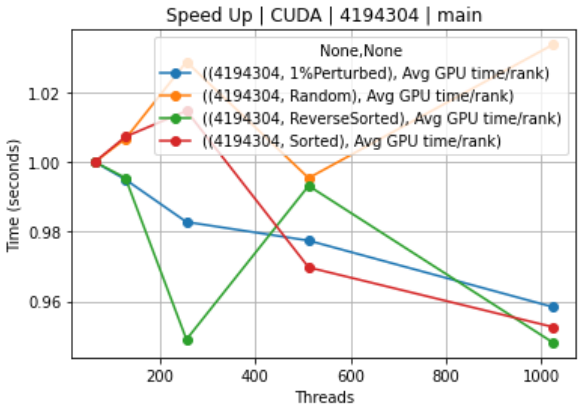


Here we seem to go back to the previous input sizes with some interesting lines in comm_large and fairly linear lines in comp_large.

Strong Scaling 4194304 Speed Up

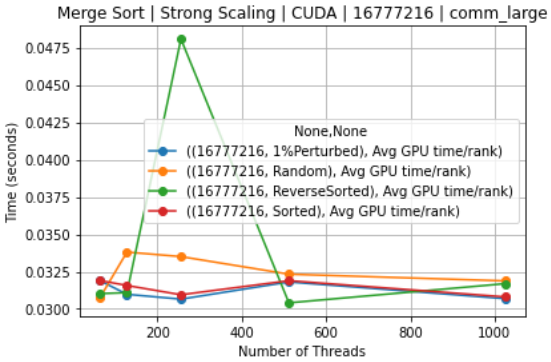


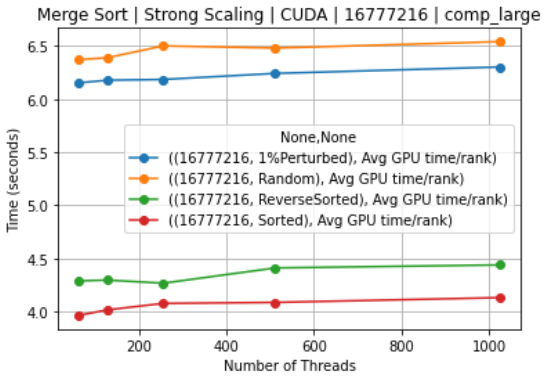


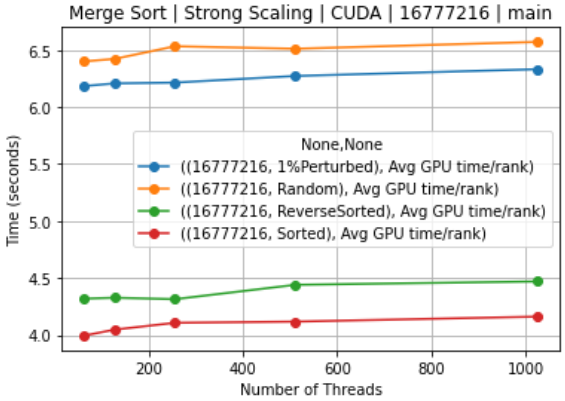


In the speed up, there seems to be little to none for either the comm_large and comp_large.

Strong Scaling 16777216

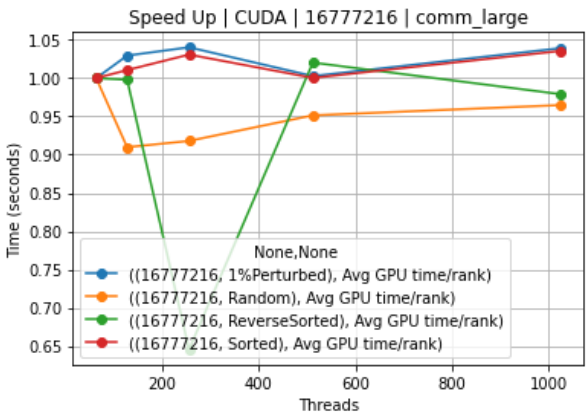


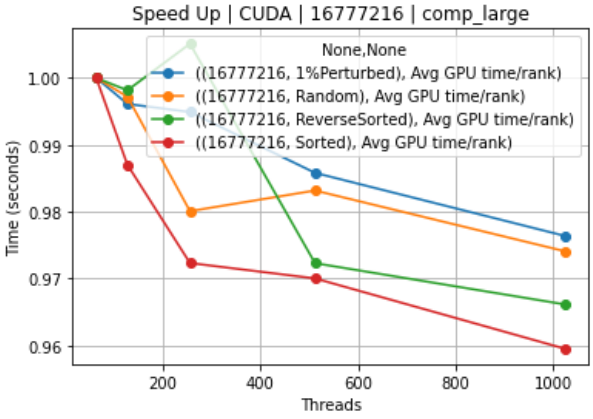


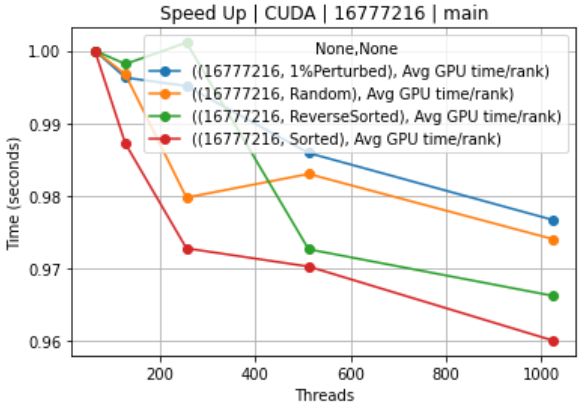


This follows the previous input size, but is quite linear for both comm_large and comp_large with a increase and decrease in comm_large.

Strong Scaling 16777216 Speed Up

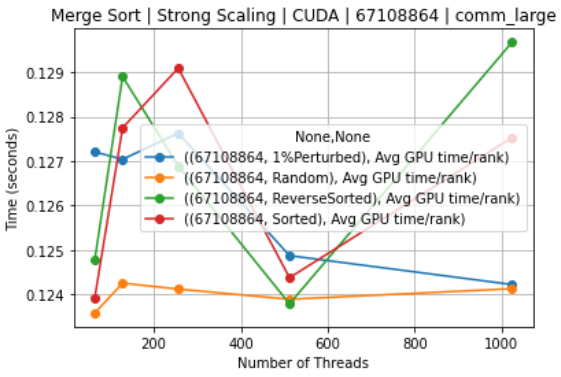


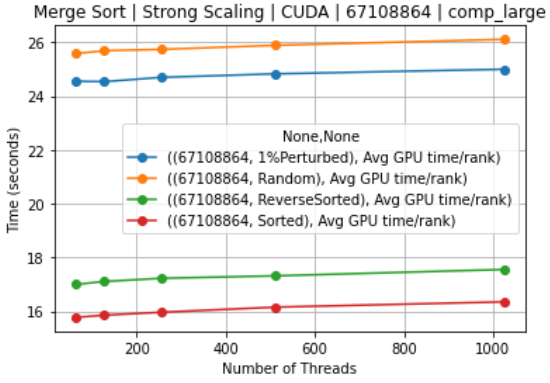


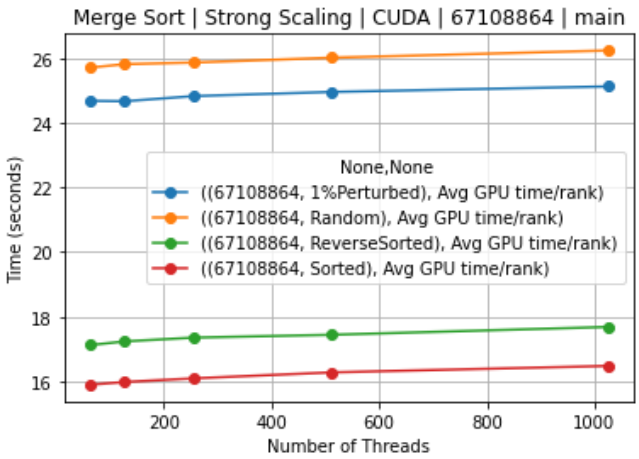


The speed up, does not show much results either, decreasing in performance as the processors increase.

Strong Scaling 67108864

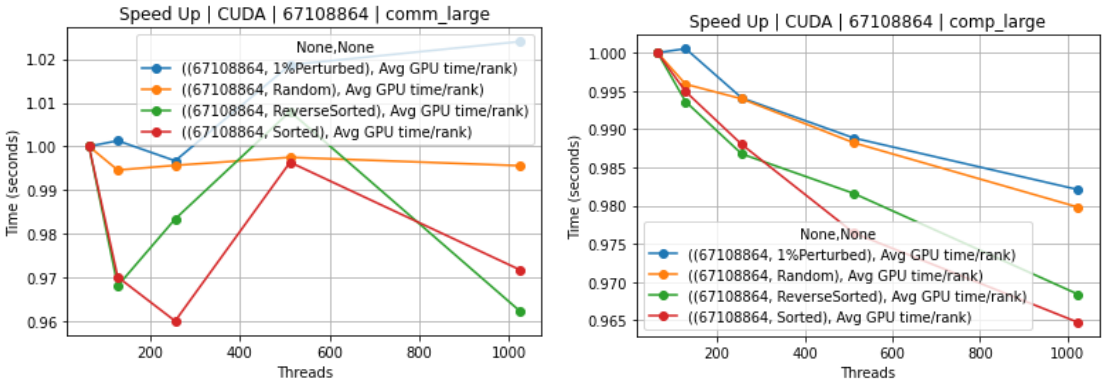


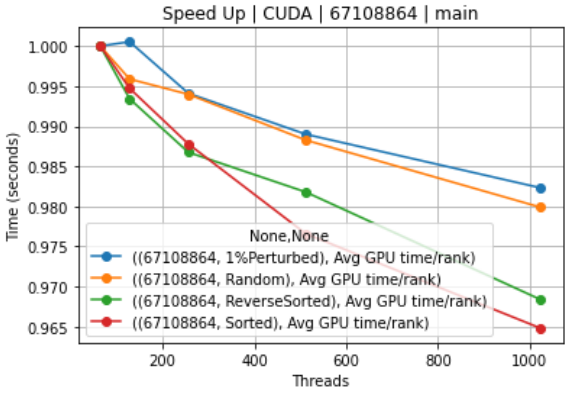




Finally this is slightly different, while the comp_large is still quite linear, the comm_large is a less stagnant but also the input types seem to follow a pattern aside from Random.

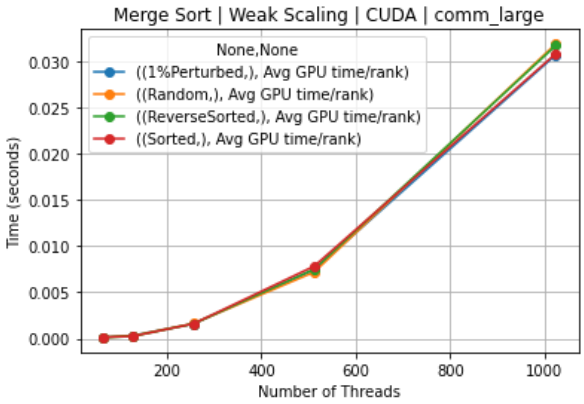
Strong Scaling 67108864 Speed Up

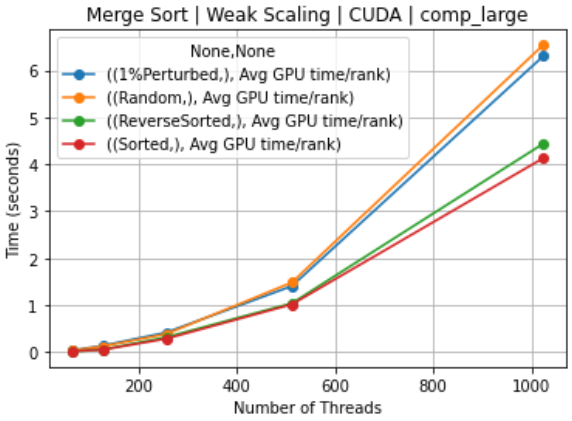


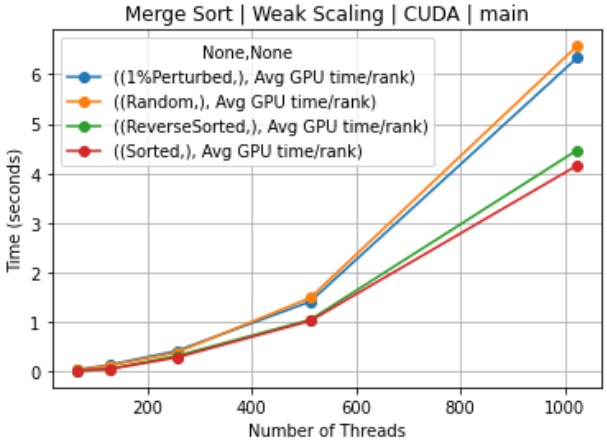


But similar to the other input sizes, the performance does seem to decrease as the processors increase.

Weak Scaling



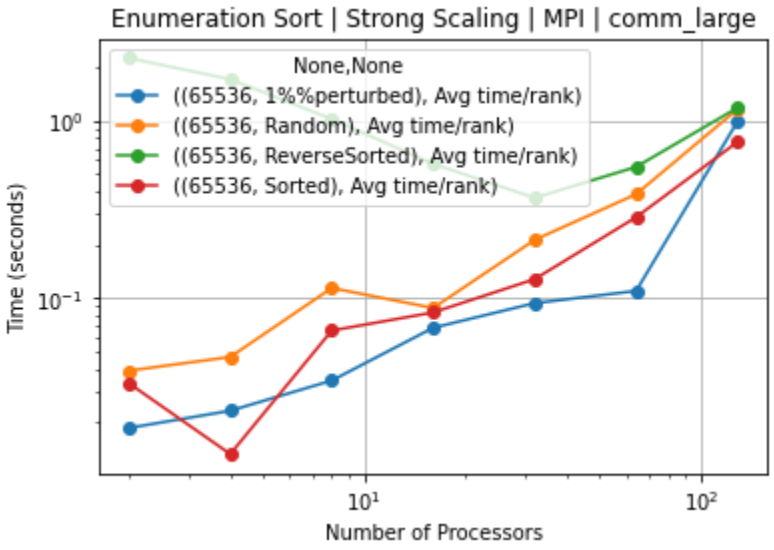


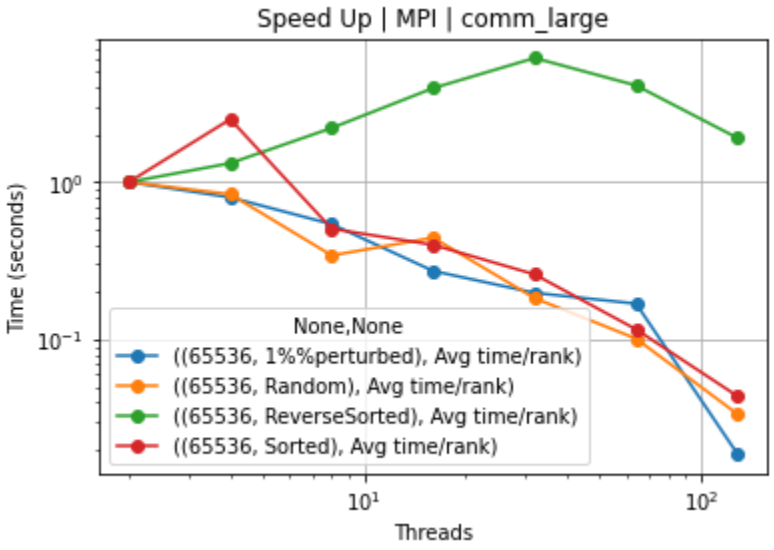


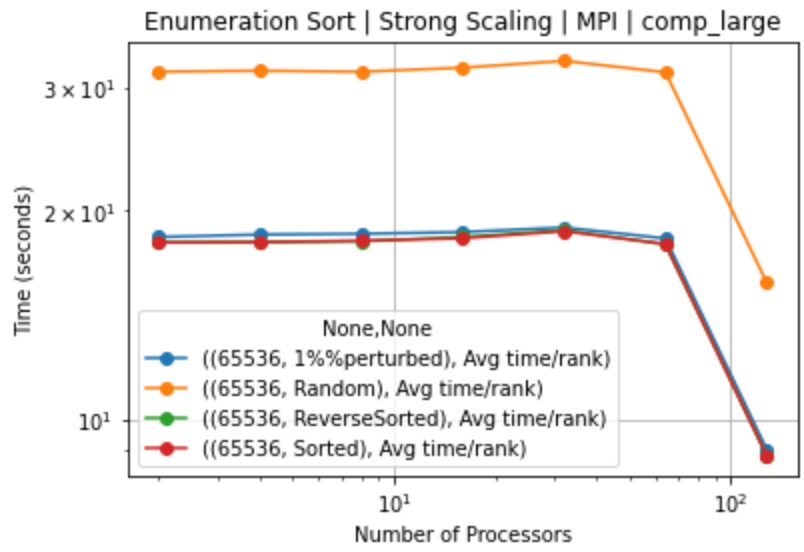
The Weak Scaling for CUDA follows a similar pattern for both comm_large and comp_large, taking more time as the number of processors increases, and doesn't seem to be flattening out like MPI.

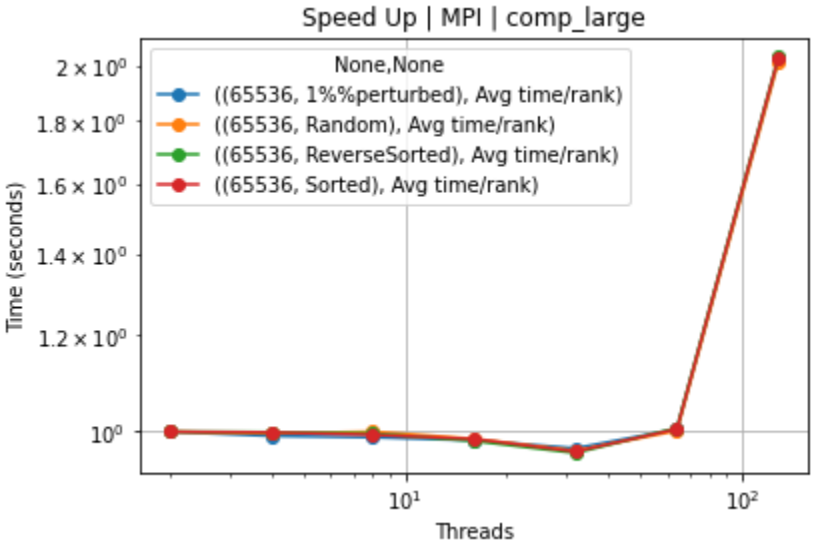
Enumeration Sort MPI:

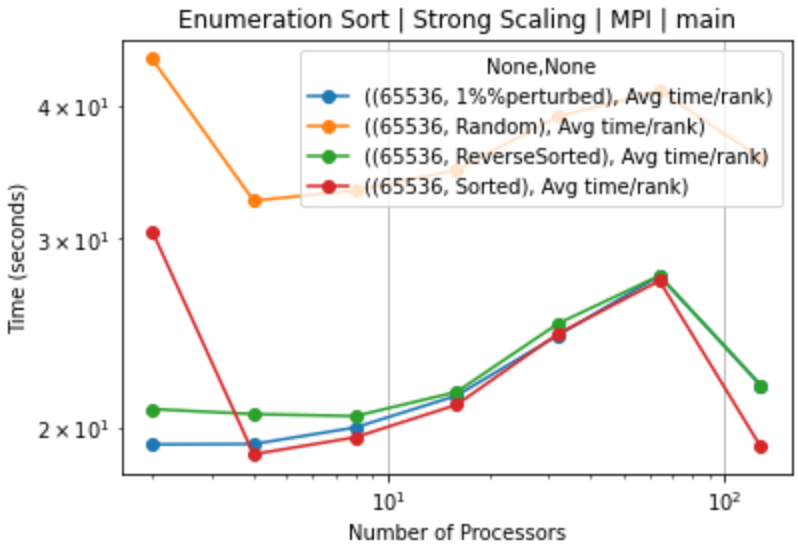
Strong Scaling Graphs

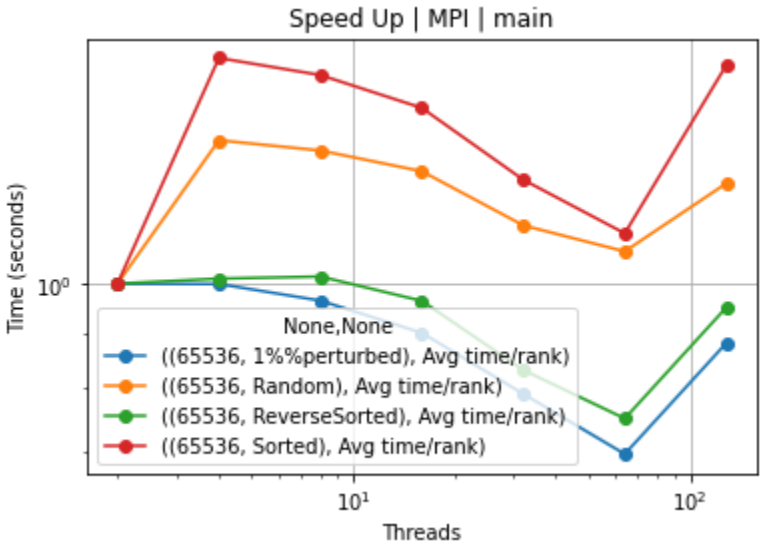




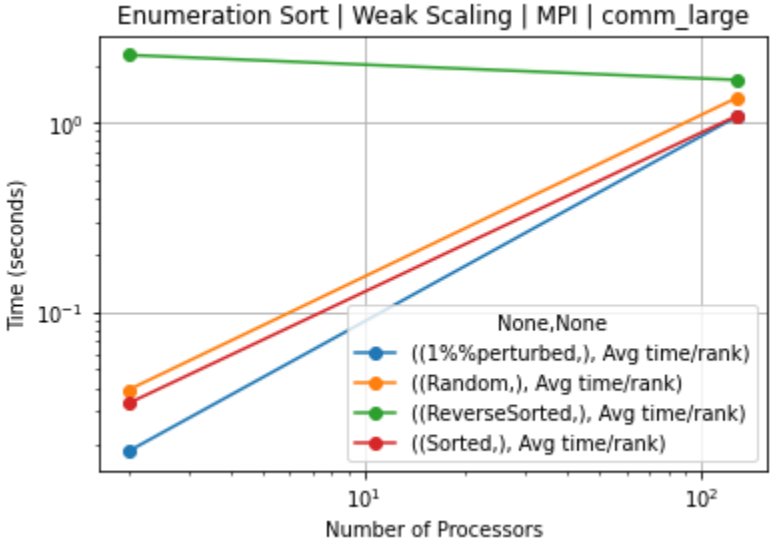

Enumeration Sort | Strong Scaling | MPI | comp_large
Time (seconds)
3×10^1
2×10^1
10^1
10^1
10^2
Number of Processors
None,None
((65536, 1%%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, ReverseSorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

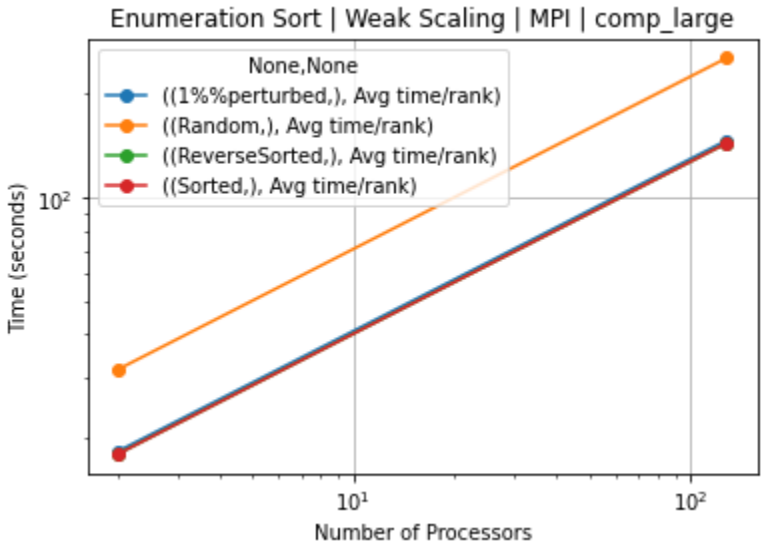

Speed Up | MPI | comp_large
Time (seconds)
2×10^0
1.8×10^0
1.6×10^0
1.4×10^0
1.2×10^0
10^0
10^1
10^2
Threads
None,None
((65536, 1%%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, ReverseSorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

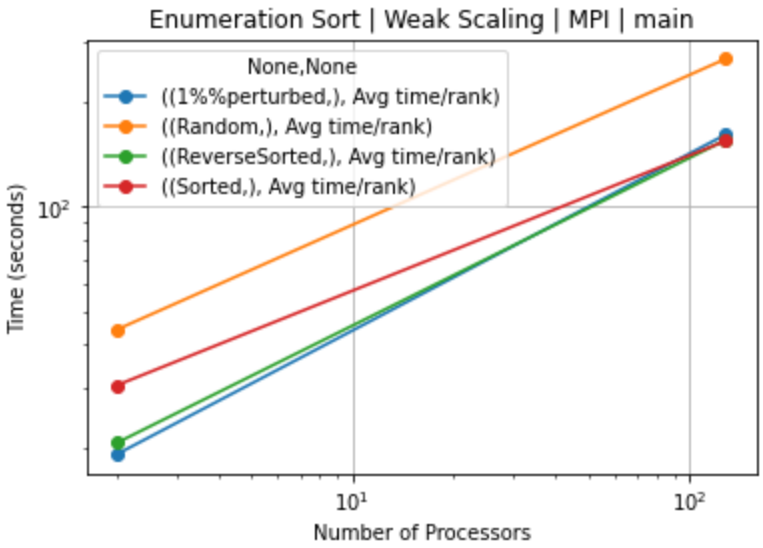

Enumeration Sort | Strong Scaling | MPI | main
Time (seconds)
4×10^1
3×10^1
2×10^1
10^1
10^2
Number of Processors
None,None
((65536, 1%%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, ReverseSorted), Avg time/rank)
((65536, Sorted), Avg time/rank)


Speed Up | MPI | main
Time (seconds)
10^0
10^1
10^2
Threads
None,None
((65536, 1%%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, ReverseSorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

Weak Scaling Graphs



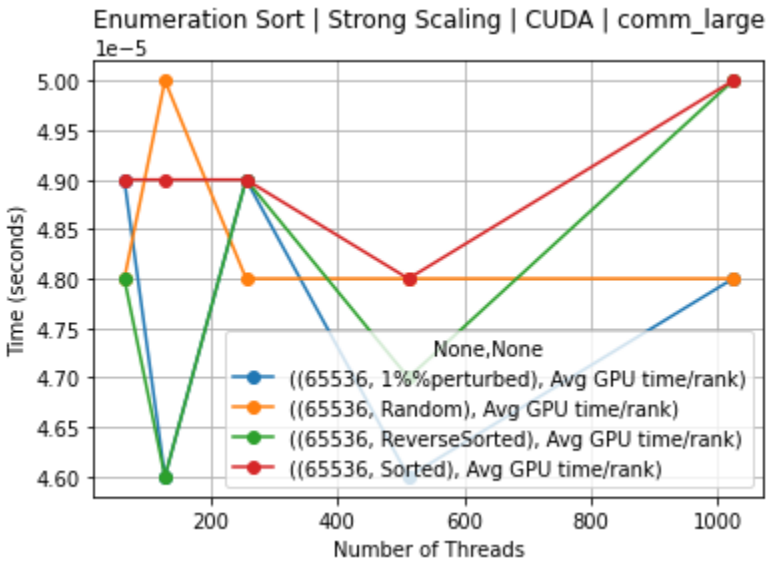


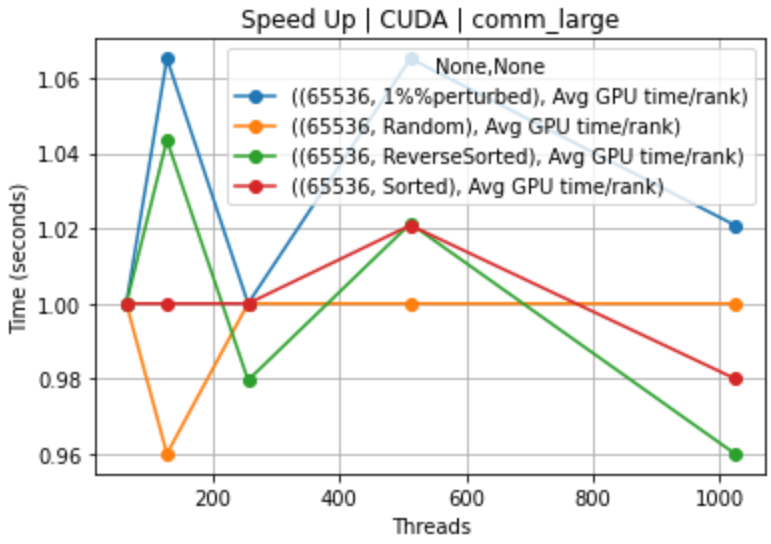


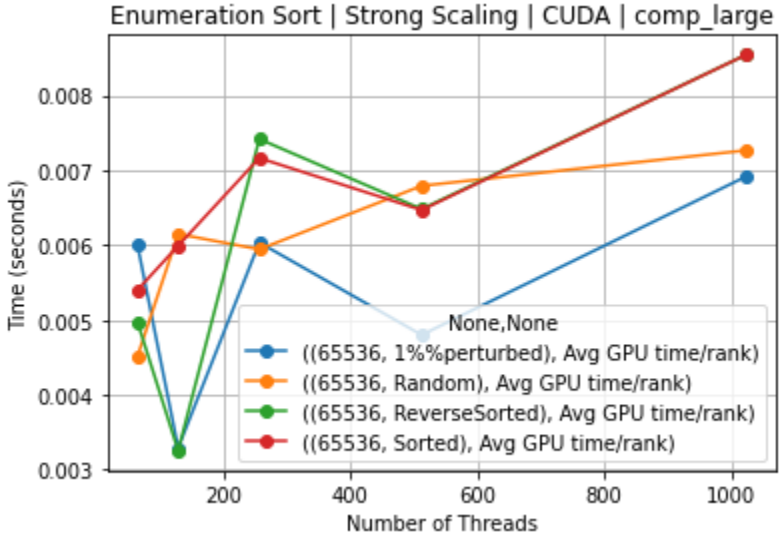
The current implementation of Enumeration Sort in MPI has many weak points. The execution time per processor stays constant with an increase in the number of processors. A reduction in runtime is also notable when increasing the number of nodes. The weak scaling graphs of MPI compare the input size with performance based on the processors. The strong scaling graphs compare the average time for different inputs. The comp graphs of strong and weak scaling show an increase in communication with correlation to the processors being increased. The main graphs show a decrease in performance as the input size is increased. In the comm_large graphs, there is no clear trend being shown while all the other graphs show a clear trend. In terms of the input types, random seemed to perform the worse based on the graphs.

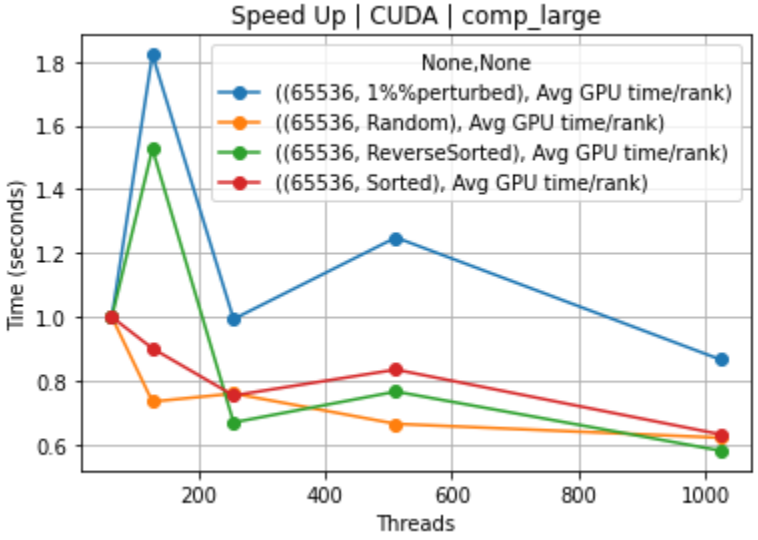
Enumeration Sort CUDA:

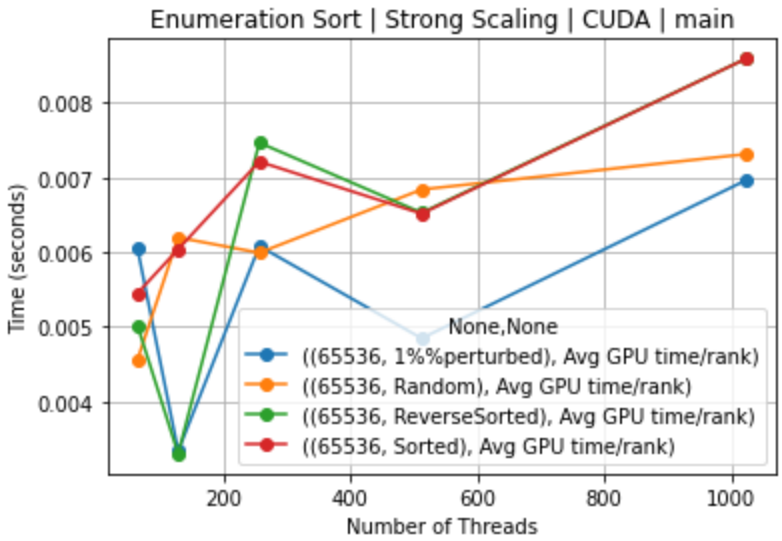
Strong Scaling Graphs

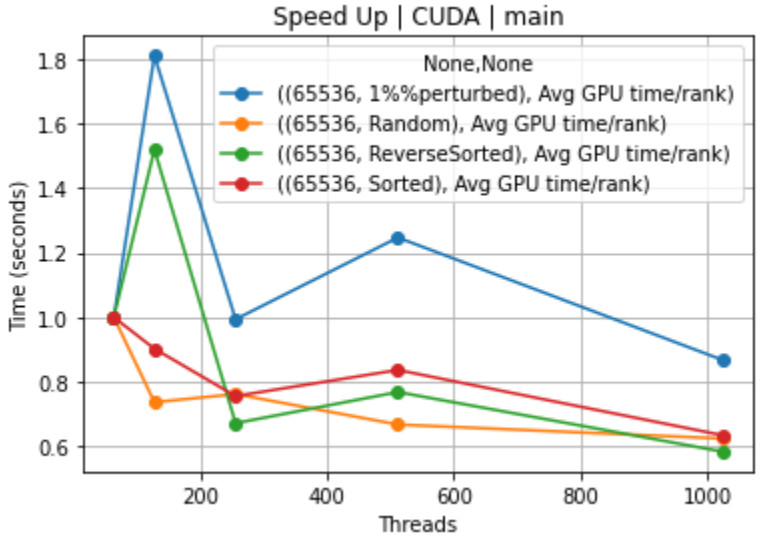



Speed Up | CUDA | comm_large
None,None
((65536, 1%%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, ReverseSorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
Threads
1.06
1.04
1.02
1.00
0.98
0.96
200
400
600
800
1000

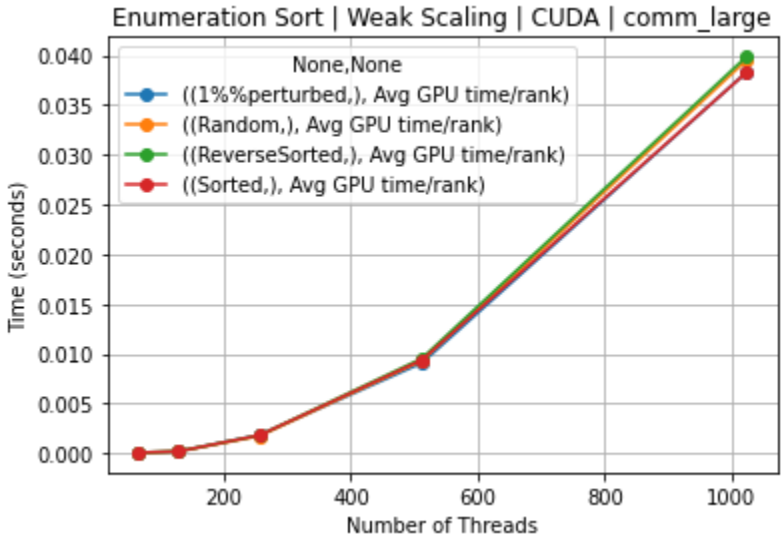

Enumeration Sort | Strong Scaling | CUDA | comp_large
None,None
((65536, 1%%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, ReverseSorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
Number of Threads
0.008
0.007
0.006
0.005
0.004
0.003
200
400
600
800
1000

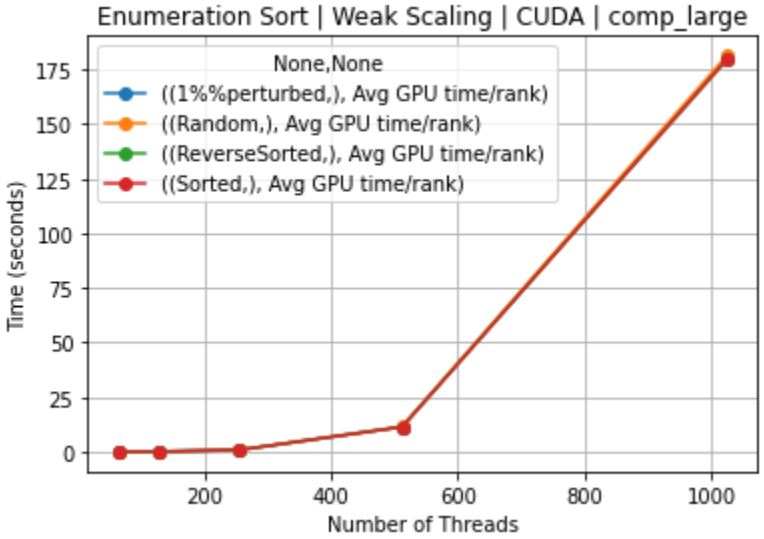

Speed Up | CUDA | comp_large
None,None
((65536, 1%%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, ReverseSorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
Threads
1.8
1.6
1.4
1.2
1.0
0.8
0.6
200
400
600
800
1000

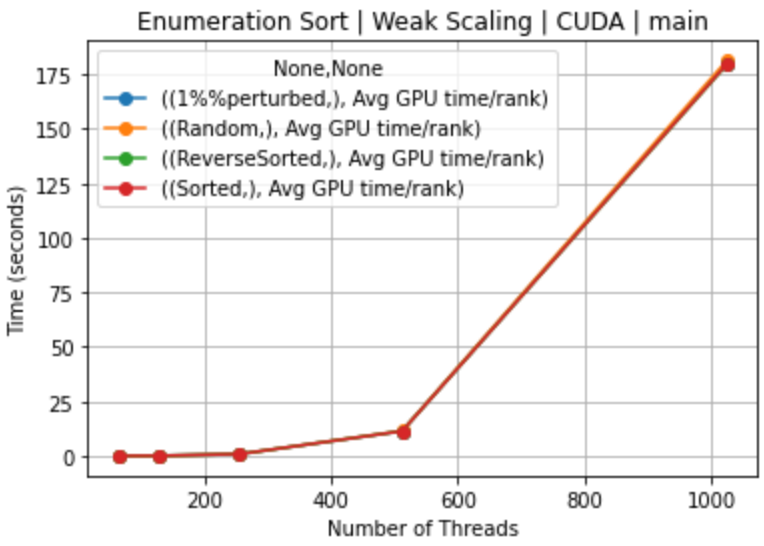

Enumeration Sort | Strong Scaling | CUDA | main
Time (seconds)
0.008
0.007
0.006
0.005
0.004
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, ReverseSorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)


Speed Up | CUDA | main
Time (seconds)
1.8
1.6
1.4
1.2
1.0
0.8
0.6
200
400
600
800
1000
Threads
None,None
((65536, 1%%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, ReverseSorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

Weak Scaling Graphs



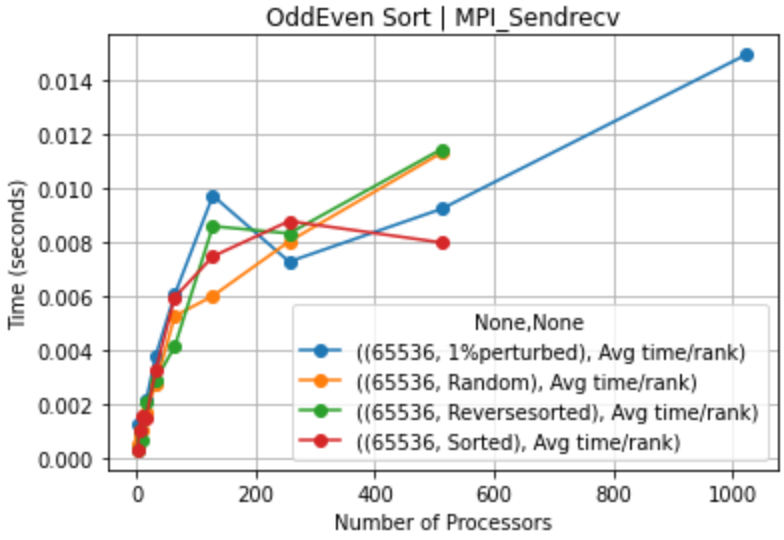


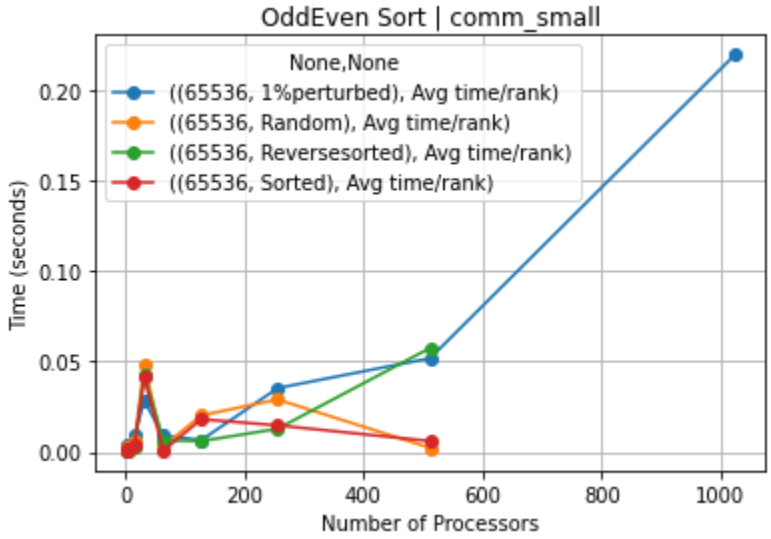


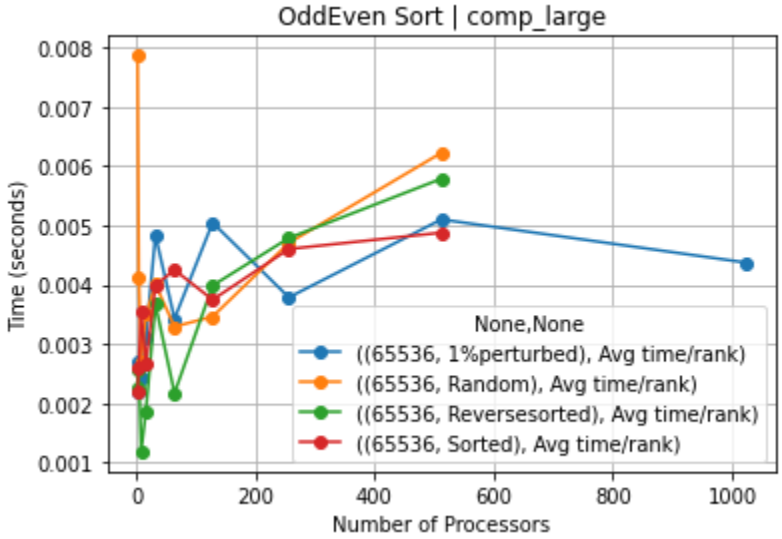
The current implementation of Enumeration Sort in CUDA shows that the increase in the number of threads does not lead to a decrease in time. This may be attributed to deficiencies of the enumeration sort algorithm. There is no speedup and weak and strong scaling are not shown for the current implementation. In addition, there are no evident performance improvements for the strong scaling graphs. This may be due to the implementation and how the arrays are accounted for. The strong scaling does not show a clear trend in CUDA which can be due to limitations for the algorithm. However, the weak scaling graphs performed relatively well in CUDA.

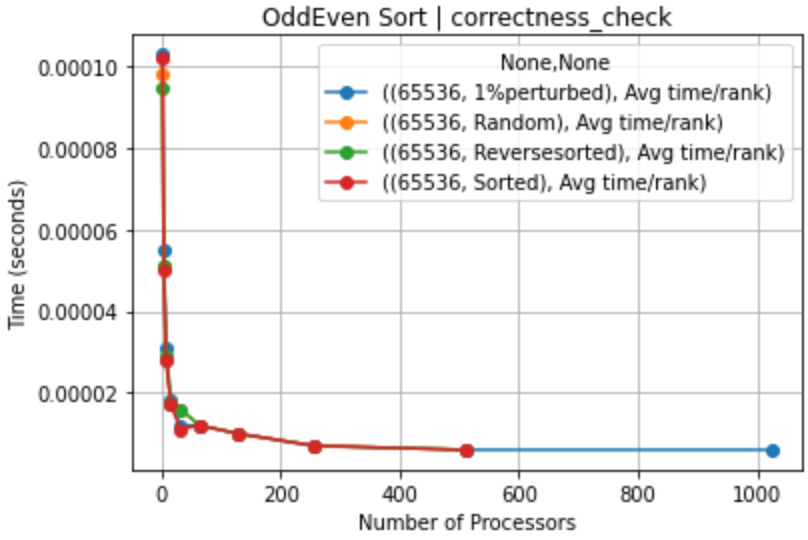
OddEven Sort MPI:

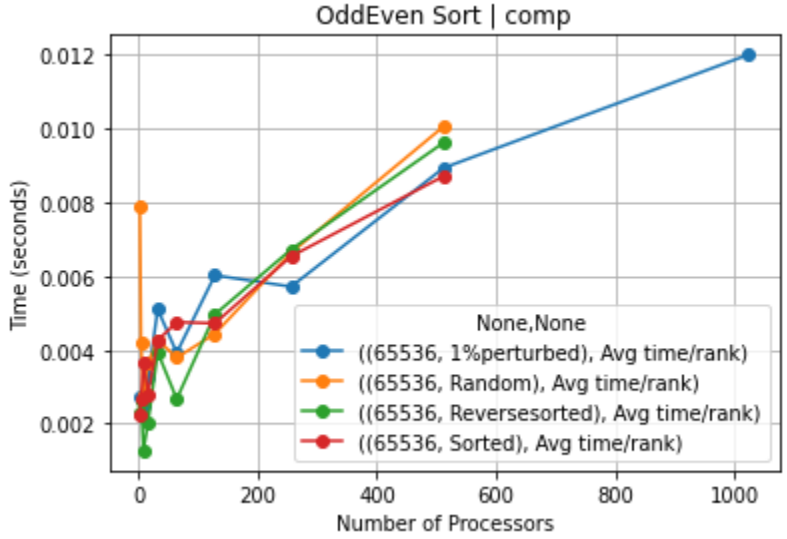
Strong Scaling 65536

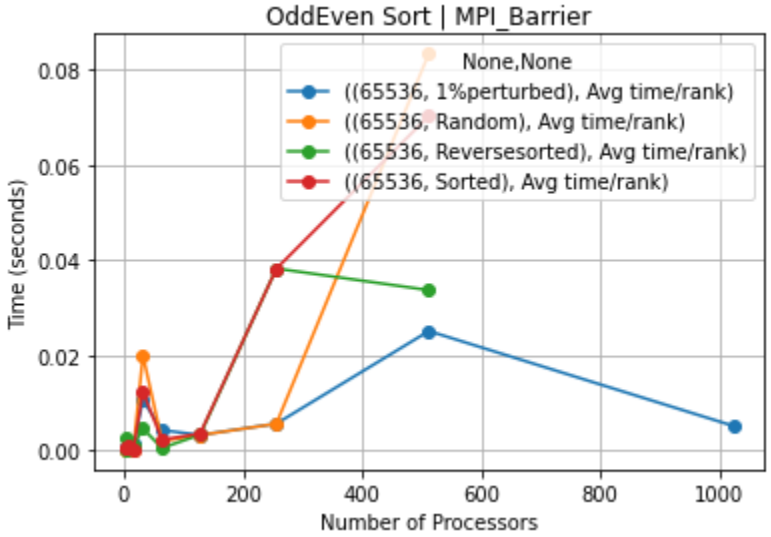

OddEven Sort | MPI_Sendrecv
Time (seconds)
Number of Processors
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)


OddEven Sort | comm_small
Time (seconds)
Number of Processors
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

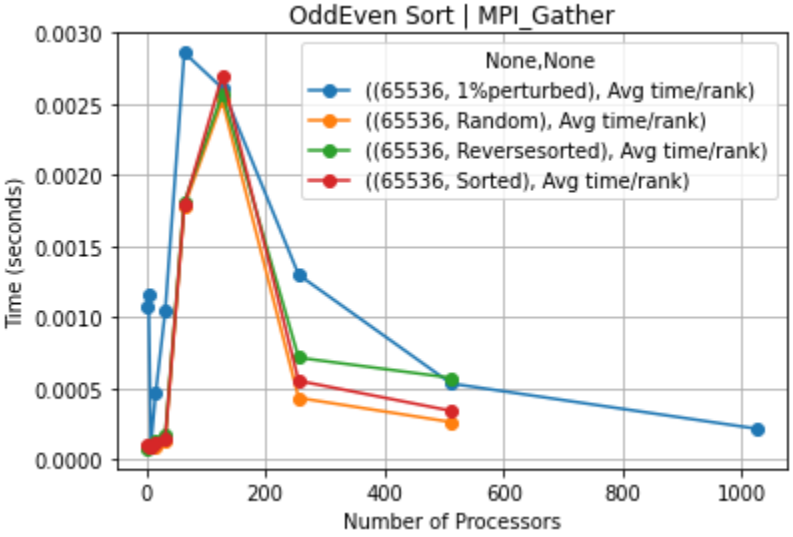

OddEven Sort | comp_large
Time (seconds)
Number of Processors
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

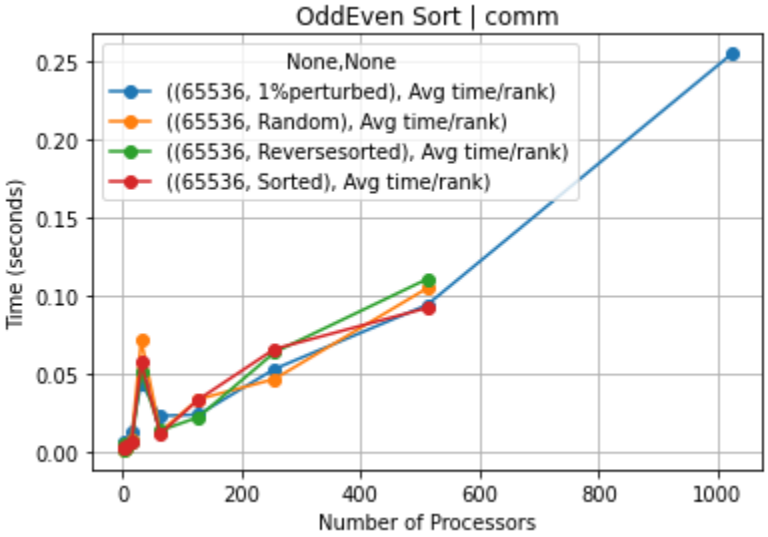

OddEven Sort | correctness_check
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
0.00010
0.00008
0.00006
0.00004
0.00002
0
200
400
600
800
1000
Number of Processors

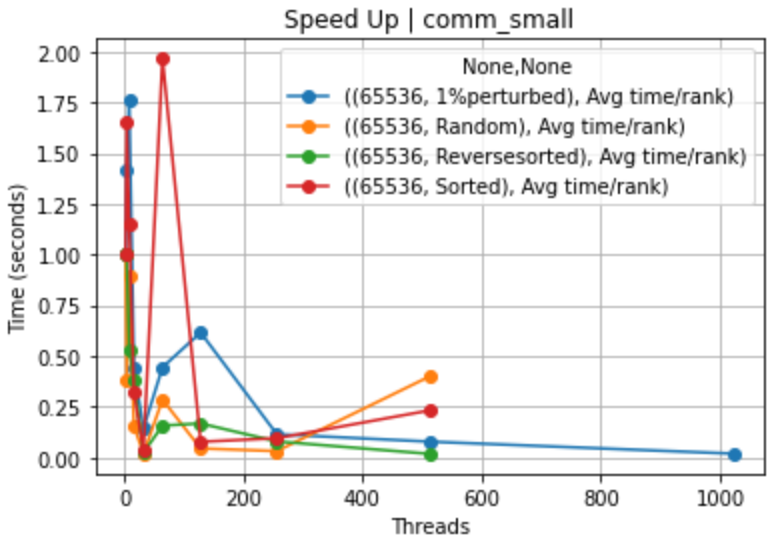

OddEven Sort | comp
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
0.012
0.010
0.008
0.006
0.004
0.002
0
200
400
600
800
1000
Number of Processors

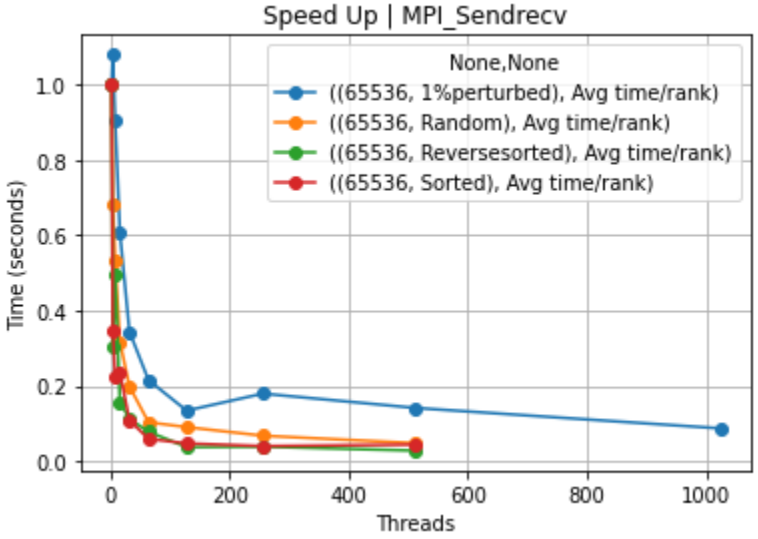

OddEven Sort | MPI_Barrier
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
0.08
0.06
0.04
0.02
0.00
0
200
400
600
800
1000
Number of Processors

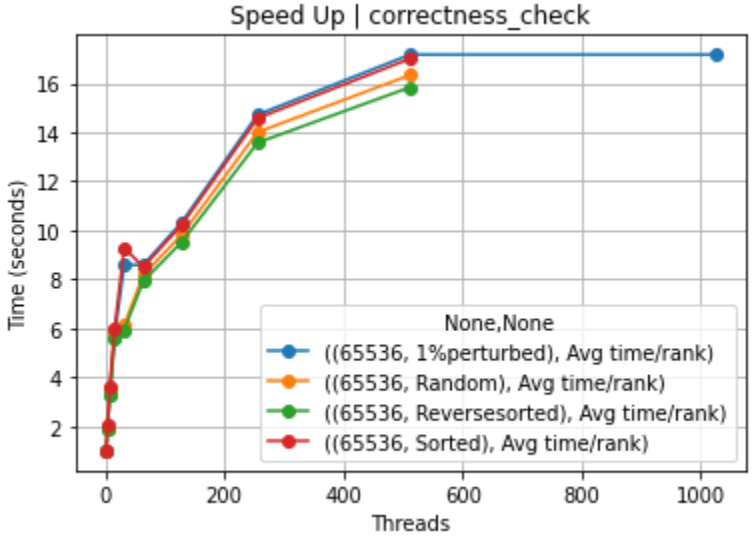

OddEven Sort | main
Time (seconds)
Number of Processors
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

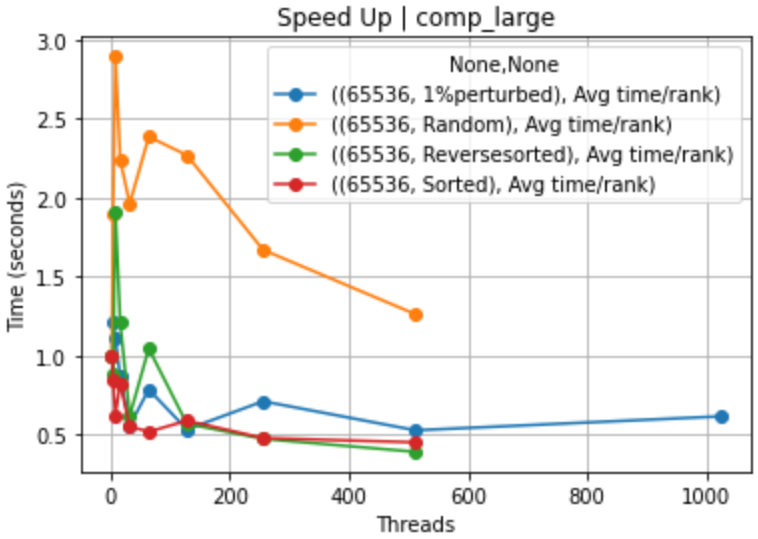

OddEven Sort | MPI_Gather
Time (seconds)
Number of Processors
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

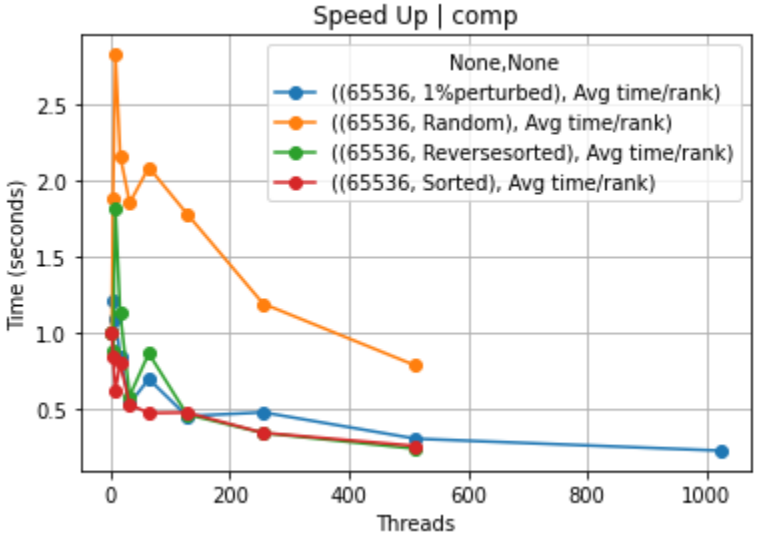

OddEven Sort | comm
Time (seconds)
Number of Processors
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

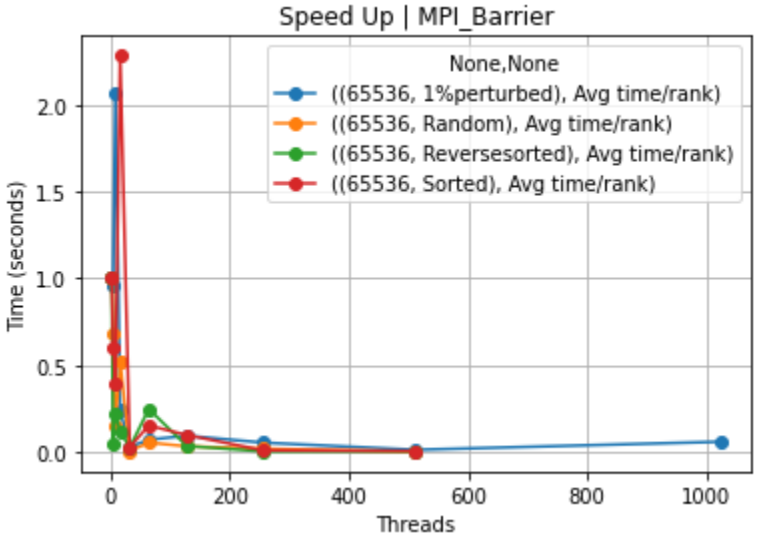

Speed Up | comm_small
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0
200
400
600
800
1000

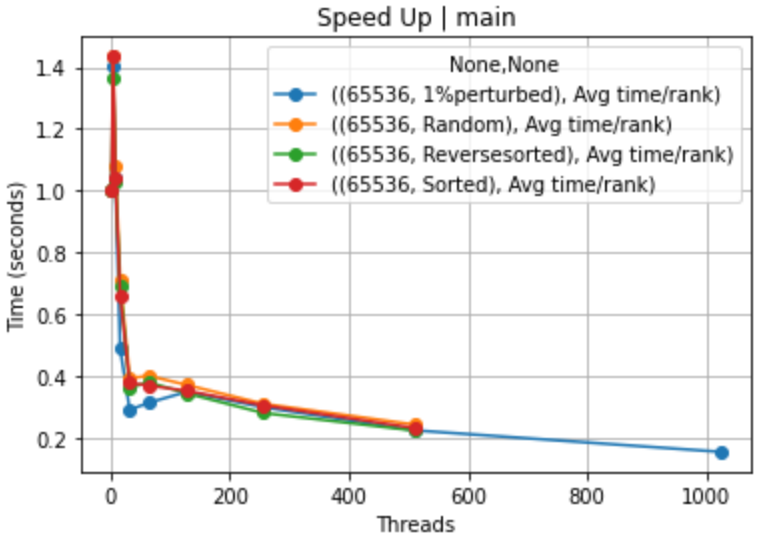

Speed Up | MPI_Sendrecv
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

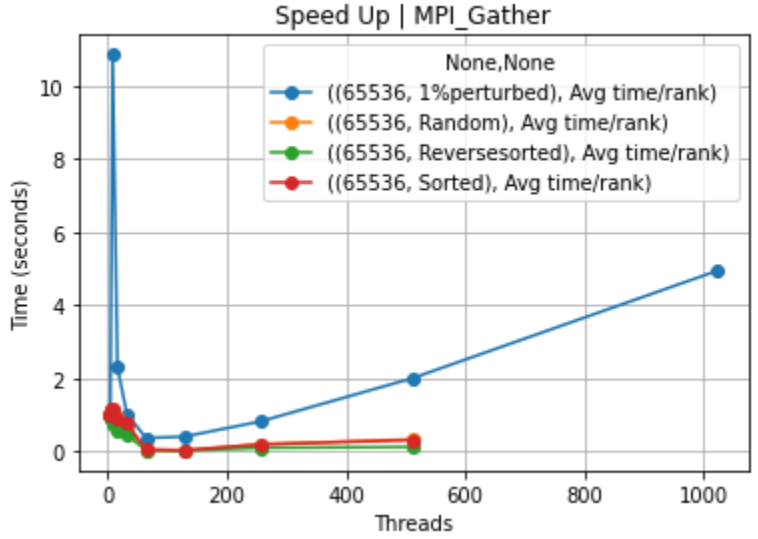

Speed Up | correctness_check
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
2
4
6
8
10
12
14
16
0
200
400
600
800
1000

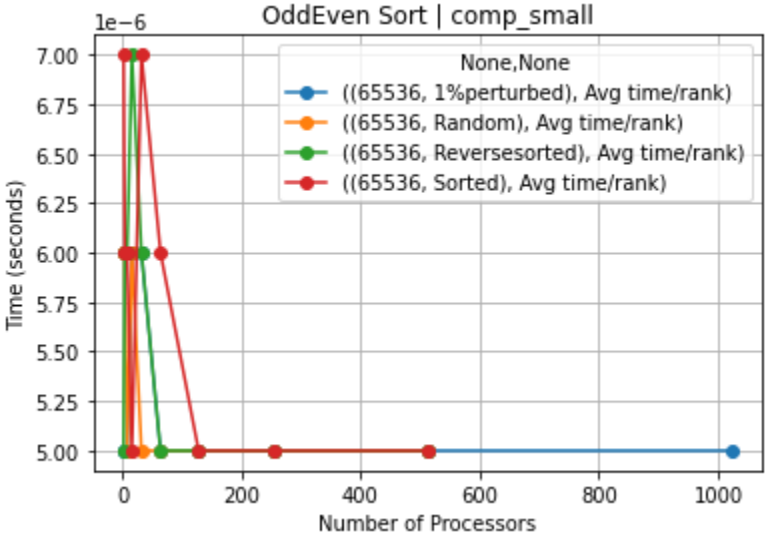

Speed Up | comp_large
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
3.0
2.5
2.0
1.5
1.0
0.5
0
200
400
600
800
1000

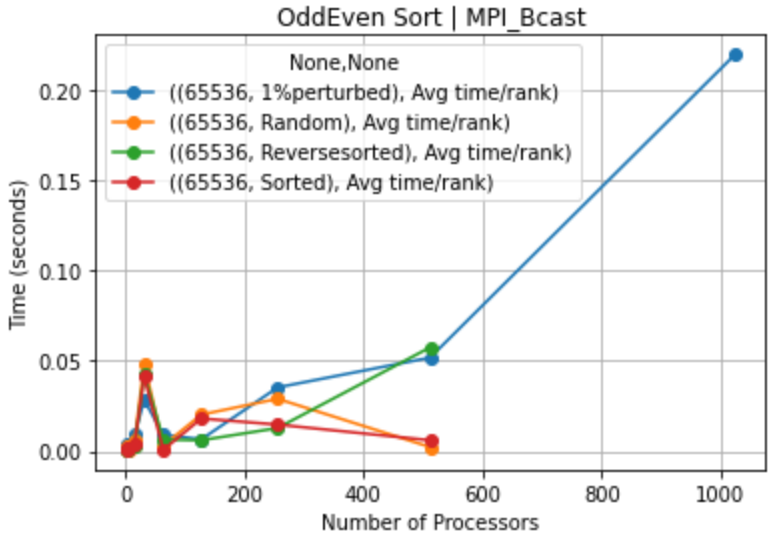

Speed Up | comp
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
2.5
2.0
1.5
1.0
0.5
0
200
400
600
800
1000

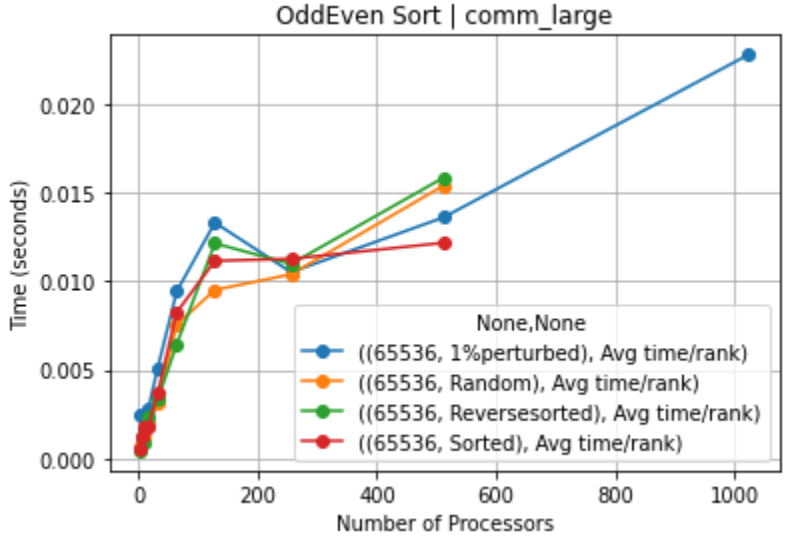

Speed Up | MPI_Barrier
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

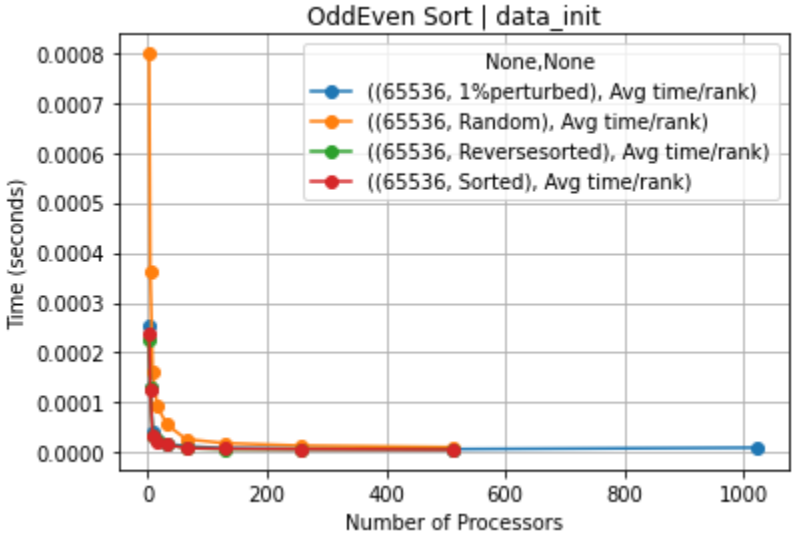

Speed Up | main
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads

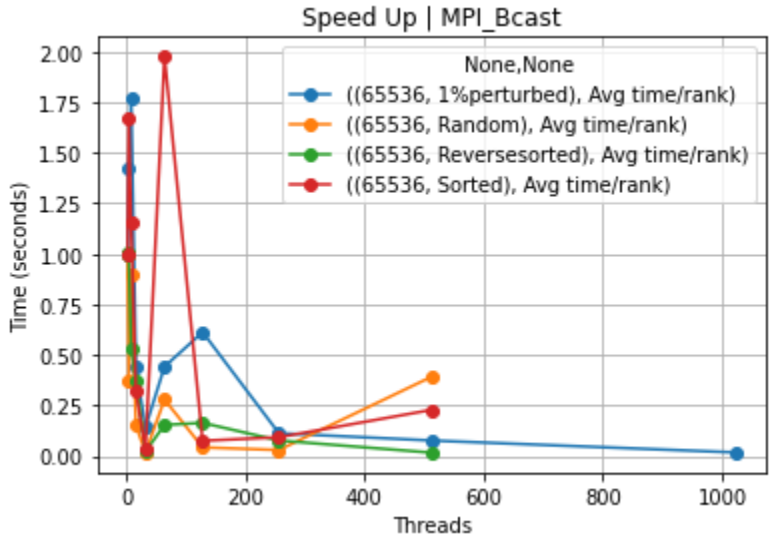

Speed Up | MPI_Gather
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads

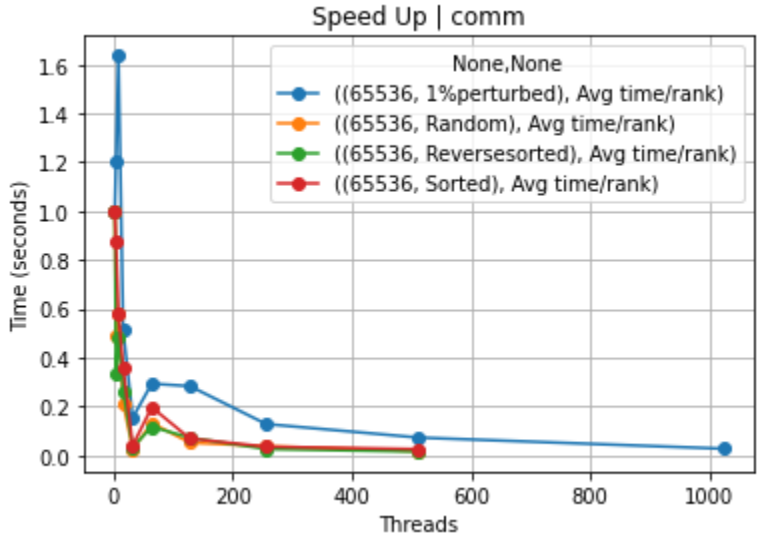

OddEven Sort | comp_small
1e−6
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Number of Processors

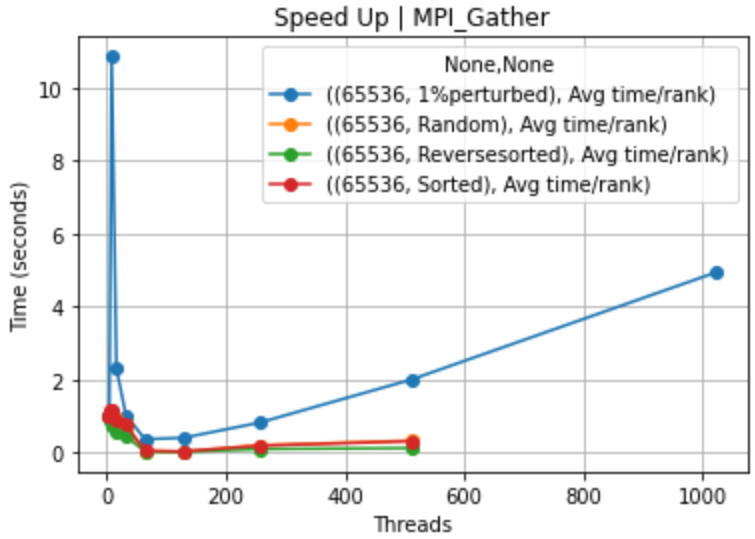

OddEven Sort | MPI_Bcast
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
0.00
0.05
0.10
0.15
0.20
0
200
400
600
800
1000
Number of Processors

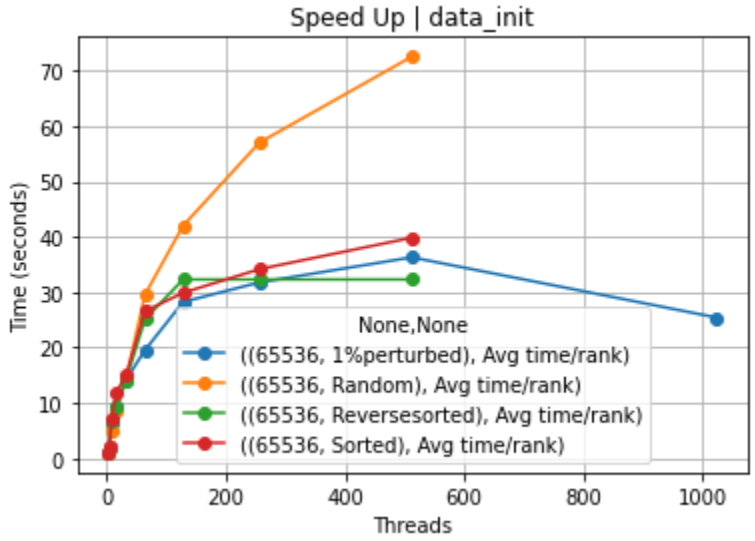

OddEven Sort | comm_large
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
0.000
0.005
0.010
0.015
0.020
0
200
400
600
800
1000
Number of Processors

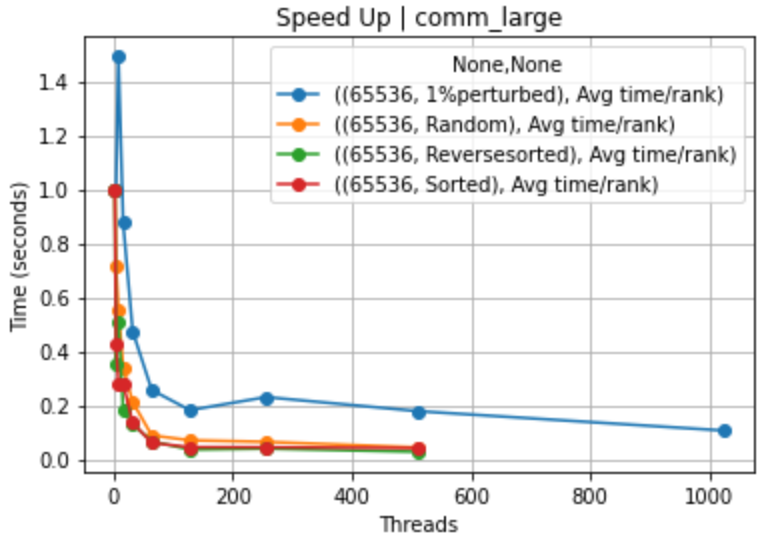

OddEven Sort | data_init
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0
200
400
600
800
1000
Number of Processors

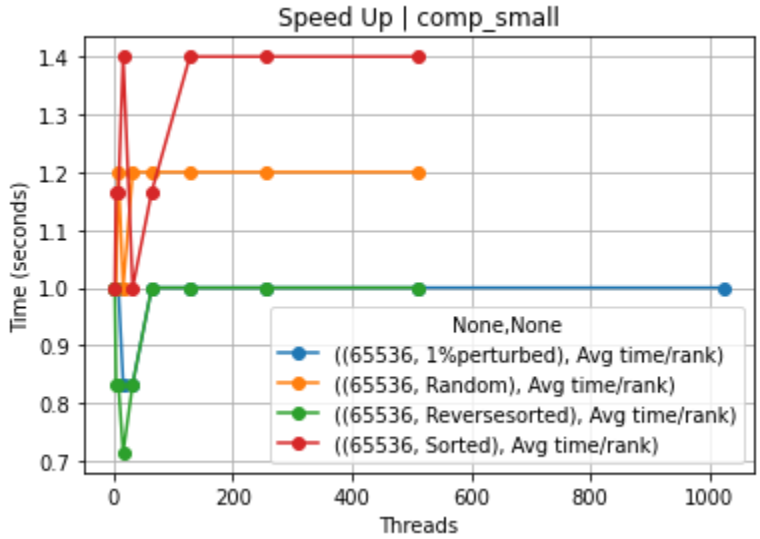

Speed Up | MPI_Bcast
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0
200
400
600
800
1000


Speed Up | comm
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
200
400
600
800
1000


Speed Up | MPI_Gather
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
0
2
4
6
8
10
0
200
400
600
800
1000

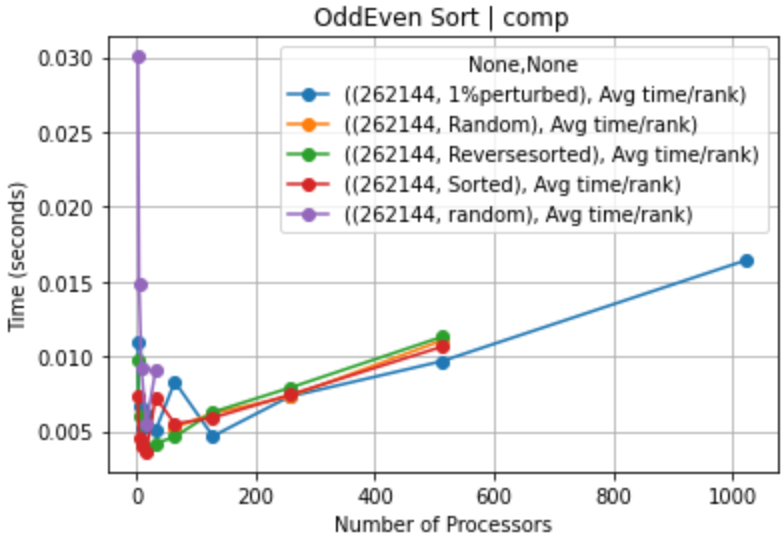

Speed Up | data_init
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
0
10
20
30
40
50
60
70
0
200
400
600
800
1000

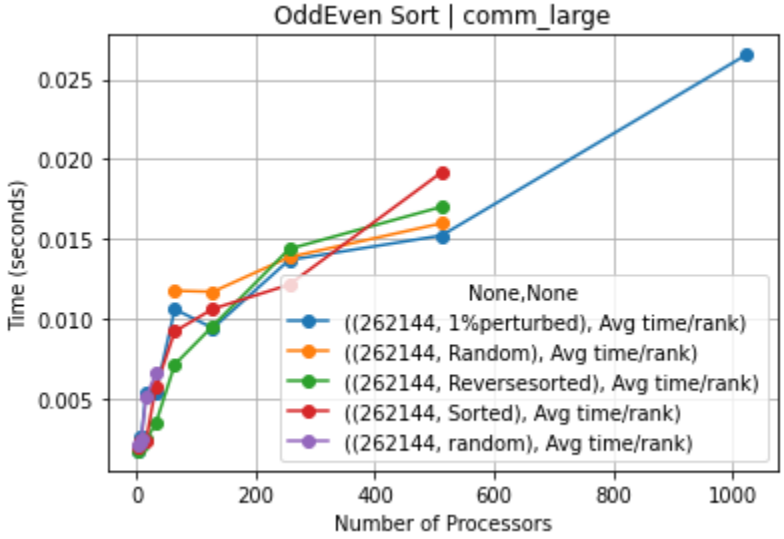


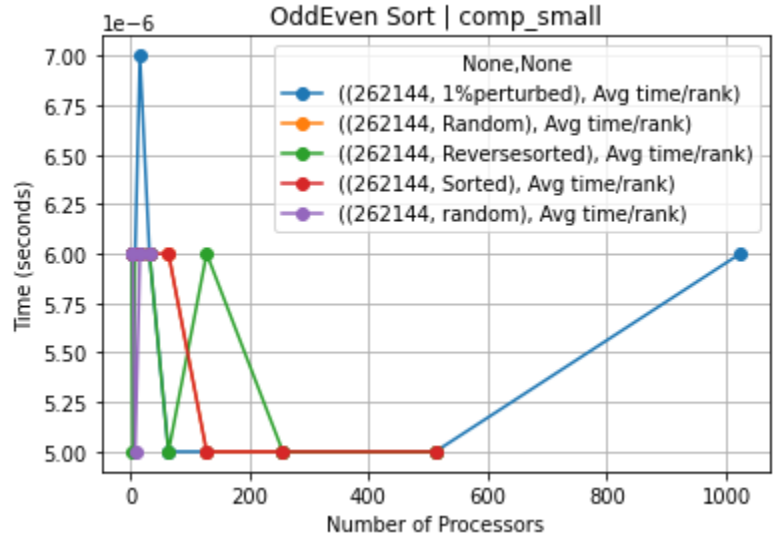


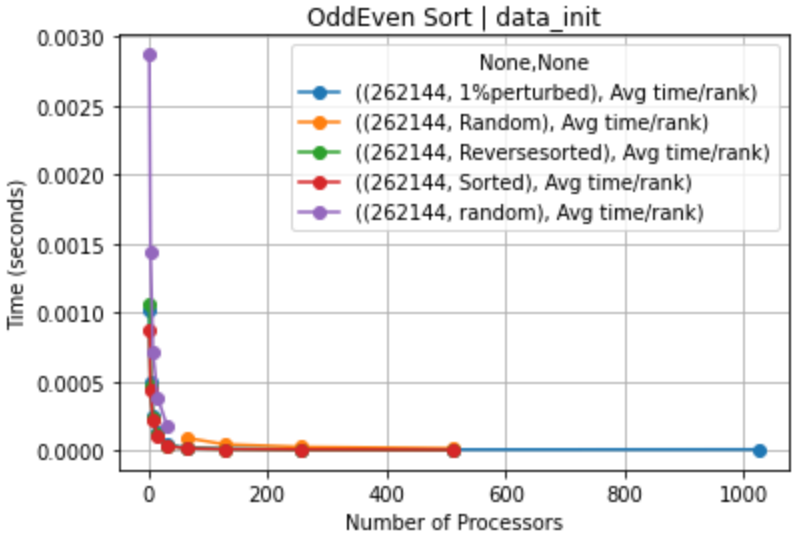
We see a linear increase in the time for large computation as the number of processes increases. This is most likely due to the fact that we are operating with a relatively small number of processes, increasing the overhead. We also observe a high communication time relative to everything else. This is because my implementation of the algorithm requires processes to transmit at most half of the entire array during each iteration of the algorithm.

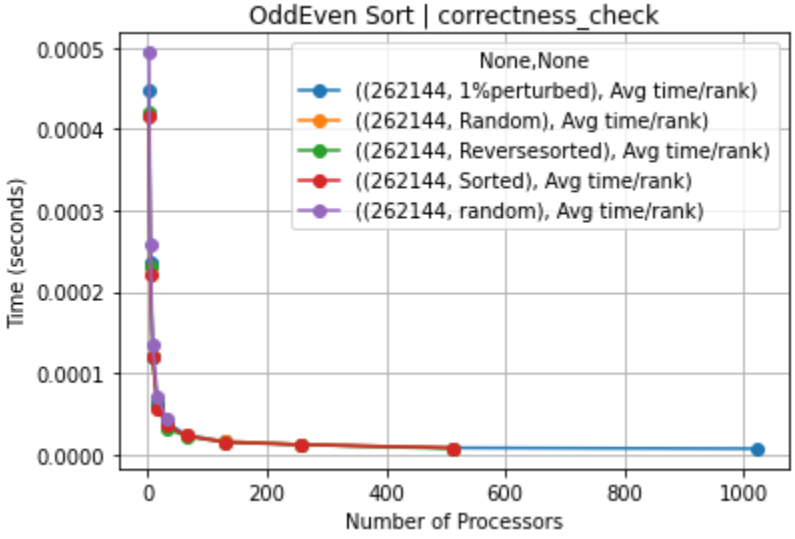
Strong Scaling 262144

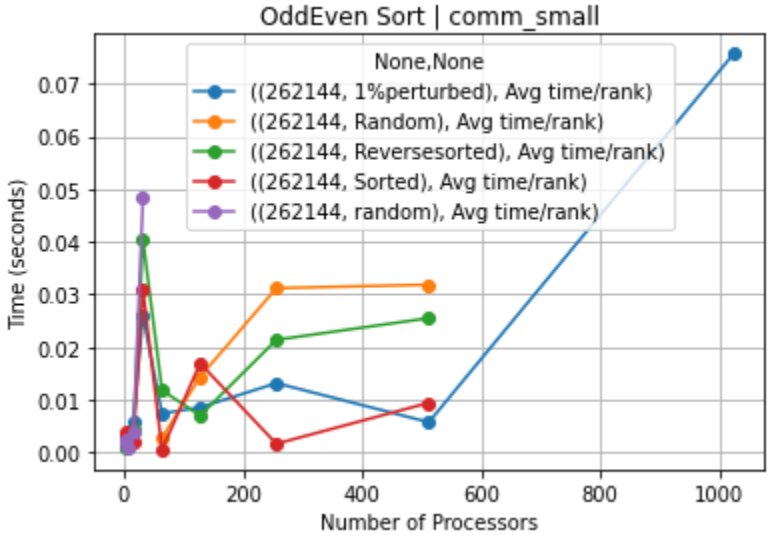

OddEven Sort | comp
Time (seconds)
Number of Processors
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)

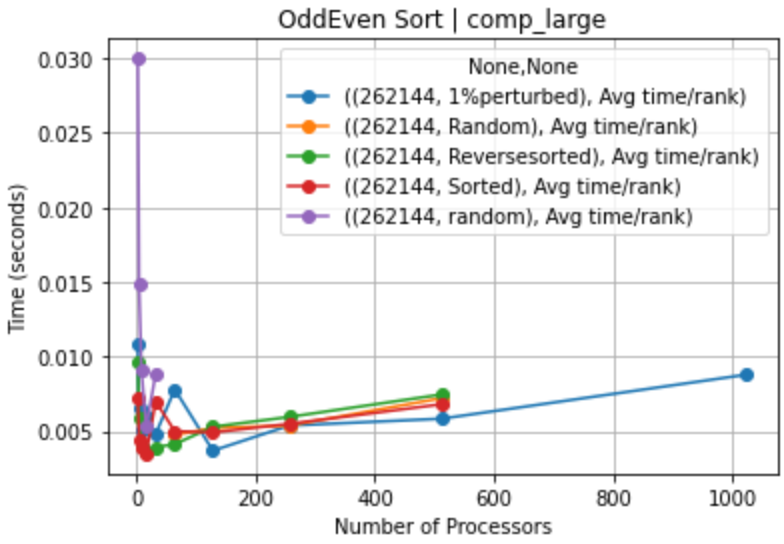

OddEven Sort | comm_large
Time (seconds)
Number of Processors
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)

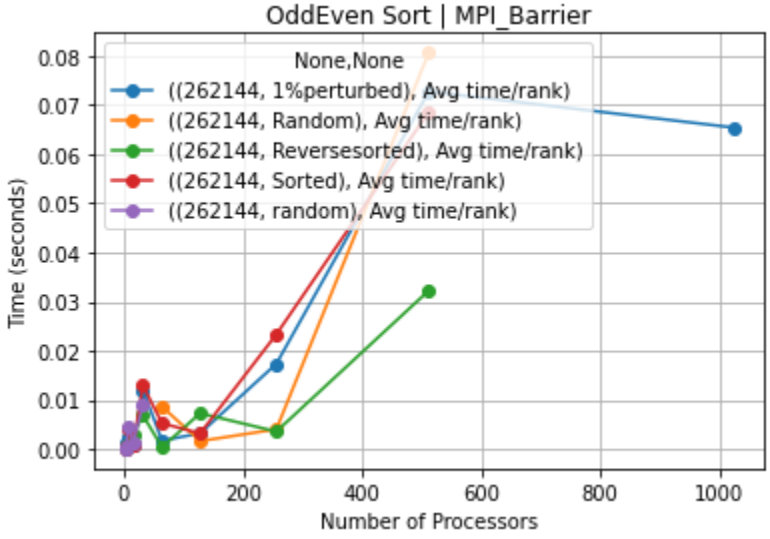

OddEven Sort | comp_small
1e−6
Time (seconds)
Number of Processors
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)

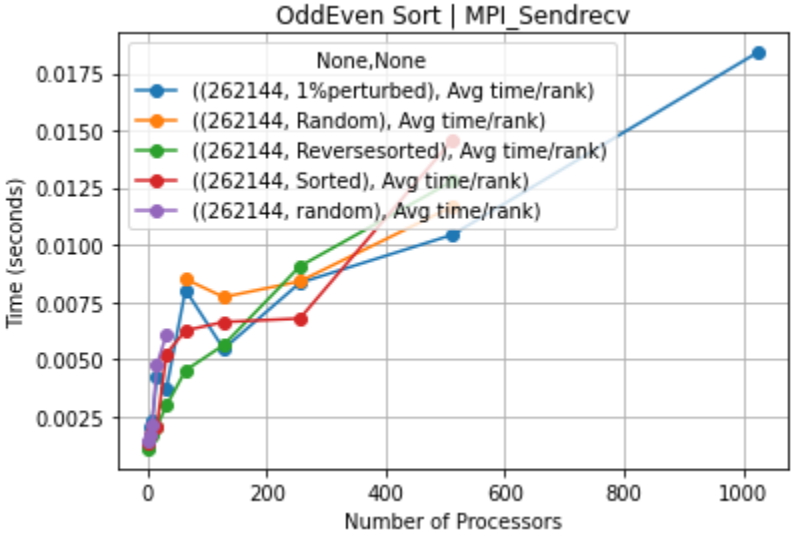

OddEven Sort | data_init
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Number of Processors

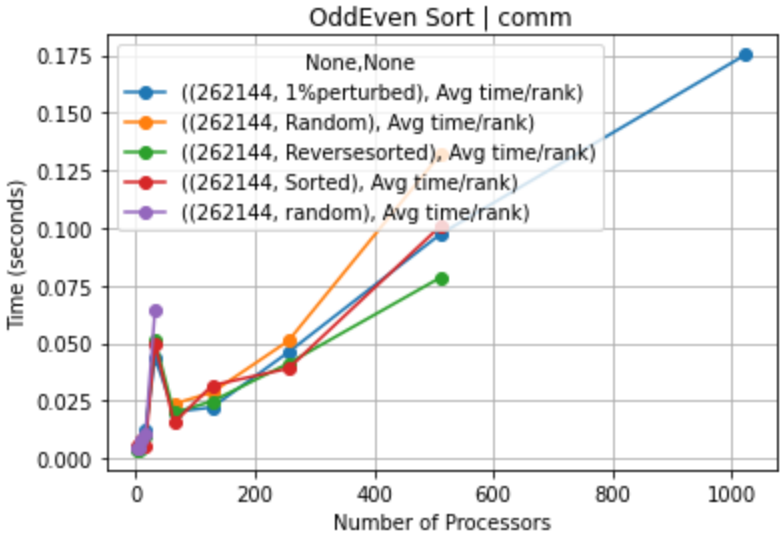

OddEven Sort | correctness_check
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Number of Processors

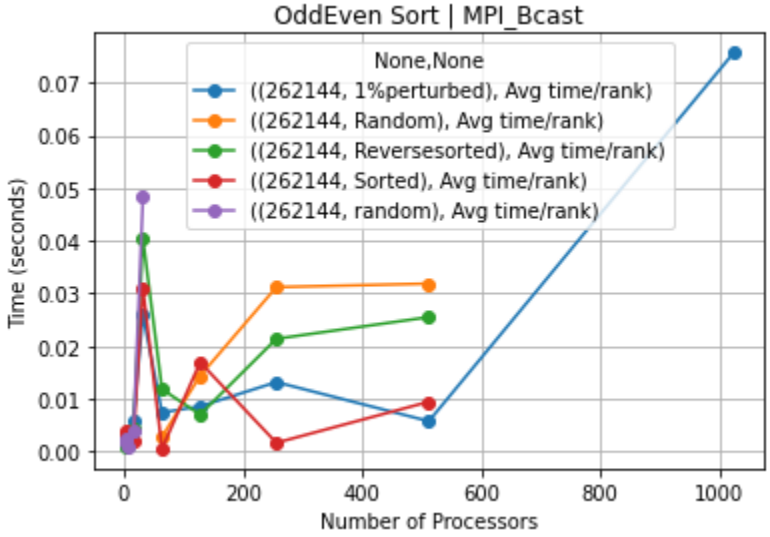

OddEven Sort | comm_small
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Number of Processors

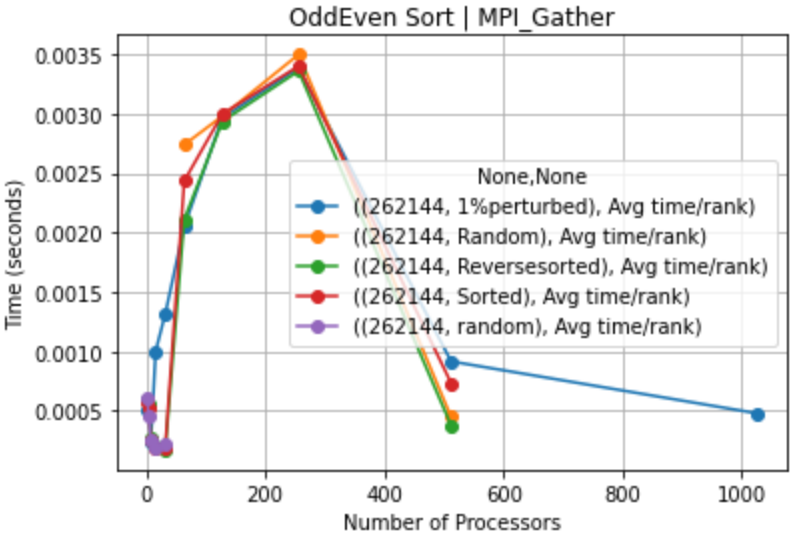

OddEven Sort | comp_large
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Number of Processors

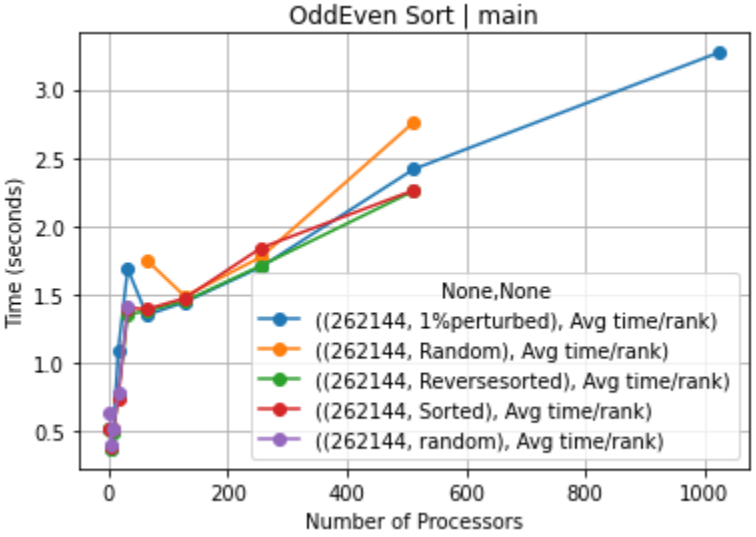

OddEven Sort | MPI_Barrier
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Number of Processors

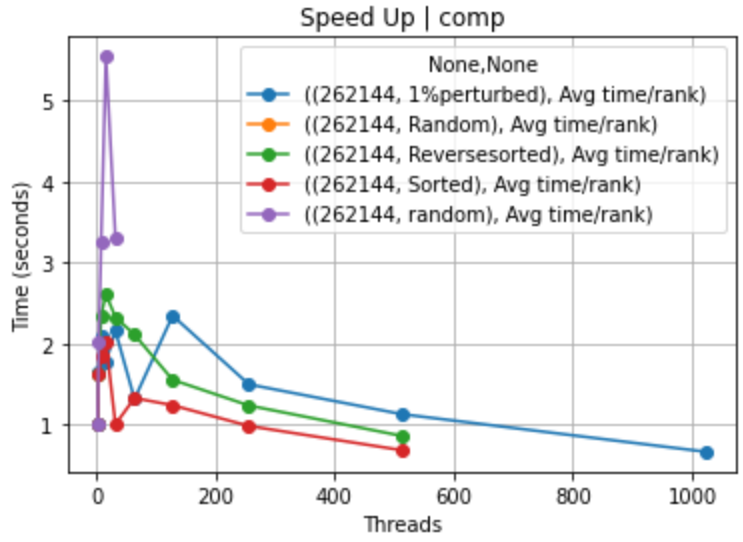

OddEven Sort | MPI_Sendrecv
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Number of Processors

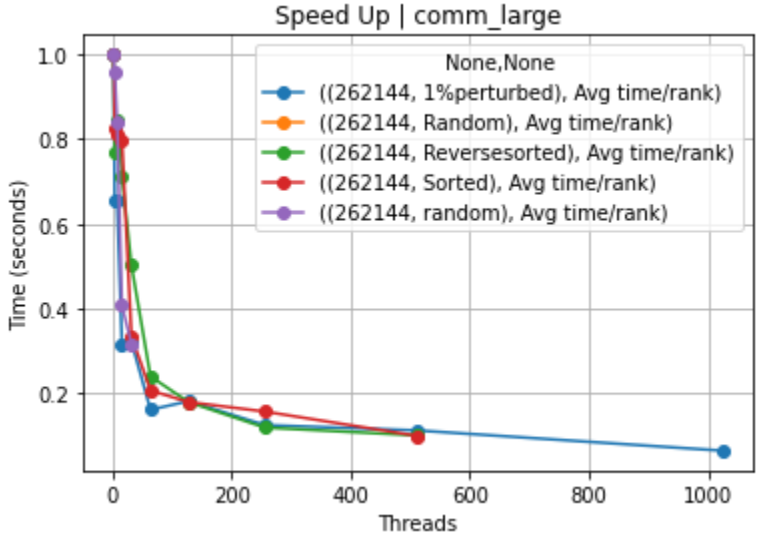

OddEven Sort | comm
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Number of Processors
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
200
400
600
800
1000

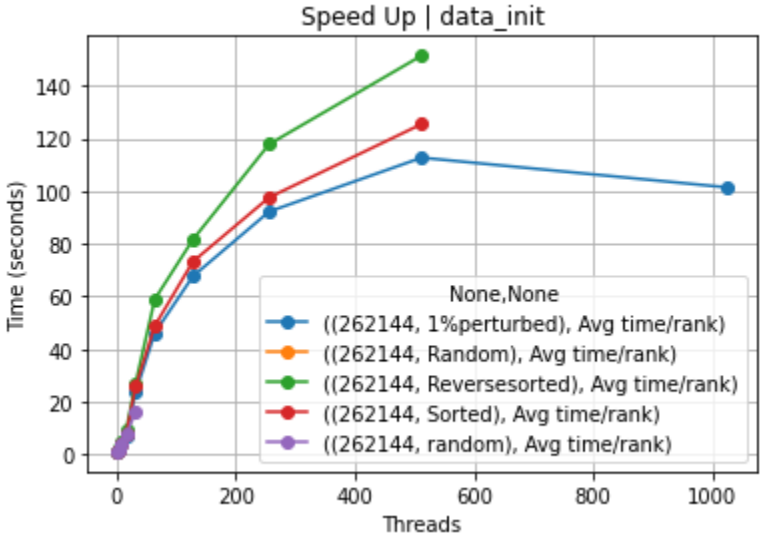

OddEven Sort | MPI_Bcast
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Number of Processors
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
200
400
600
800
1000

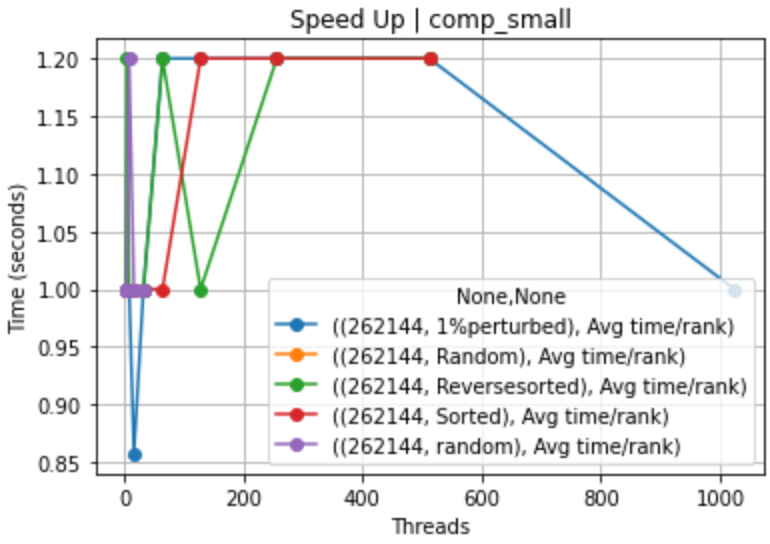

OddEven Sort | MPI_Gather
Time (seconds)
Number of Processors
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)

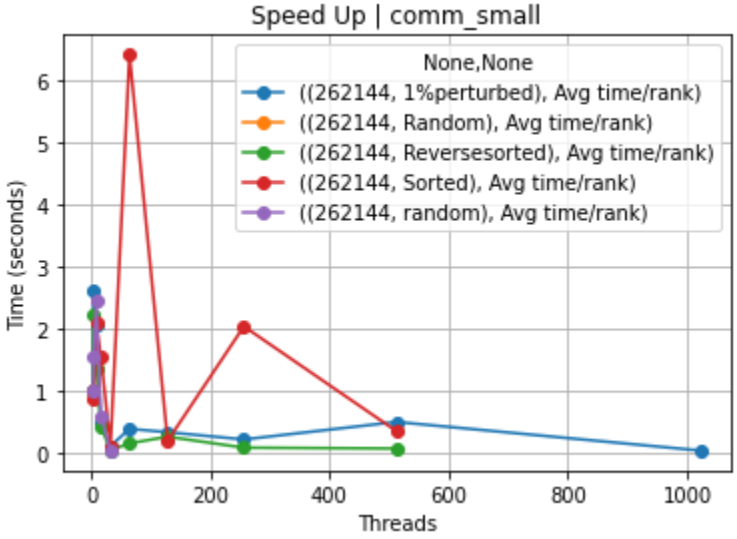

OddEven Sort | main
Time (seconds)
Number of Processors
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)

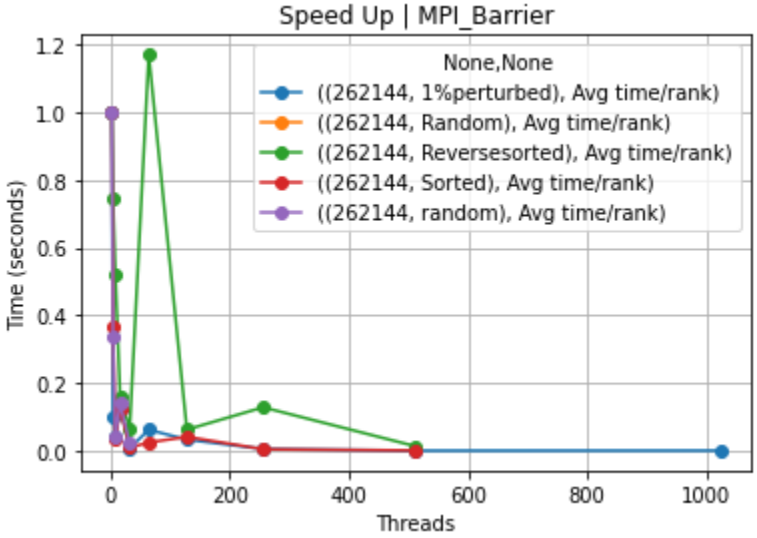

Speed Up | comp
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Threads
1
2
3
4
5
0
200
400
600
800
1000

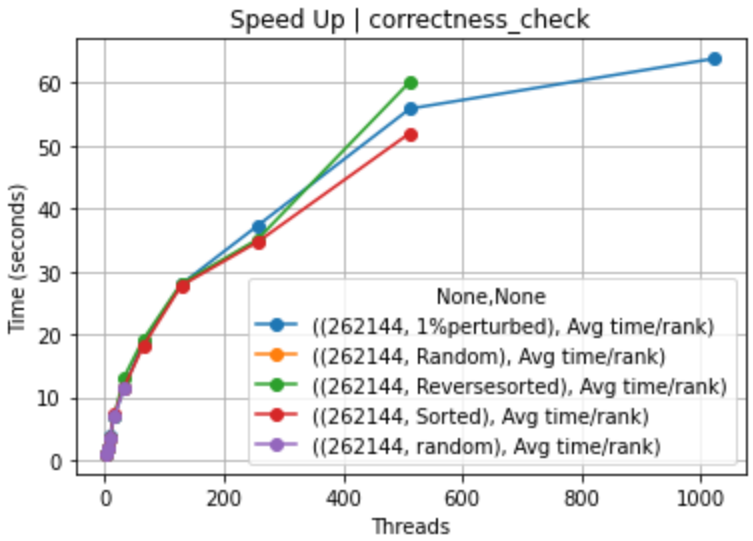

Speed Up | comm_large
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Threads
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

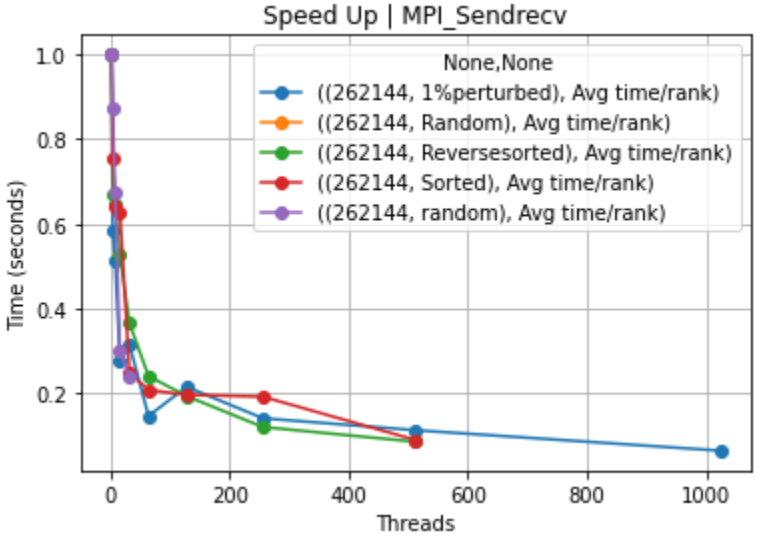

Speed Up | data_init
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Threads
0
20
40
60
80
100
120
140
0
200
400
600
800
1000

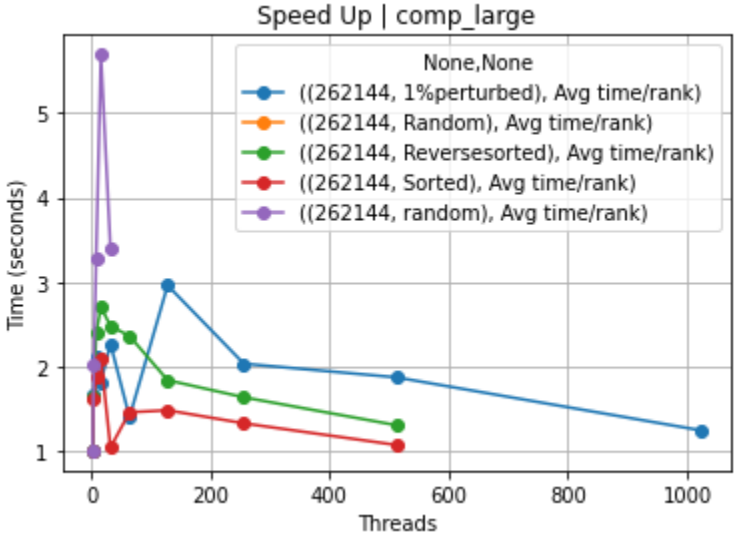

Speed Up | comp_small
Time (seconds)
Threads
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)

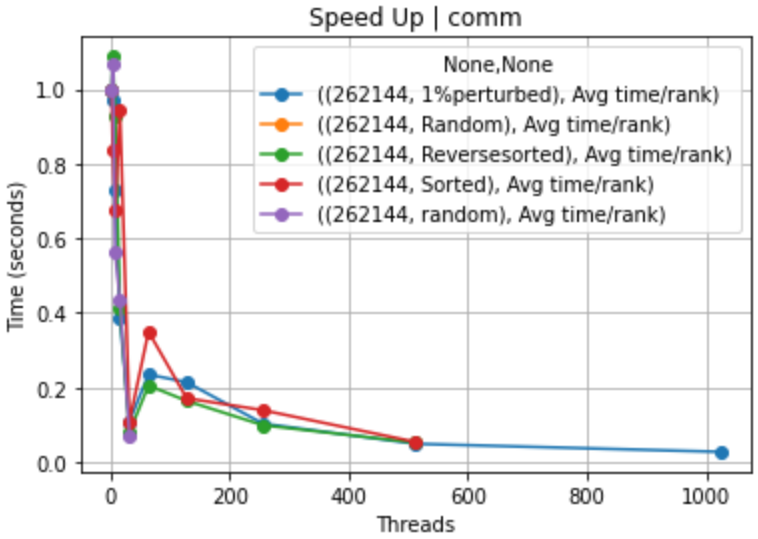

Speed Up | comm_small
Time (seconds)
Threads
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)

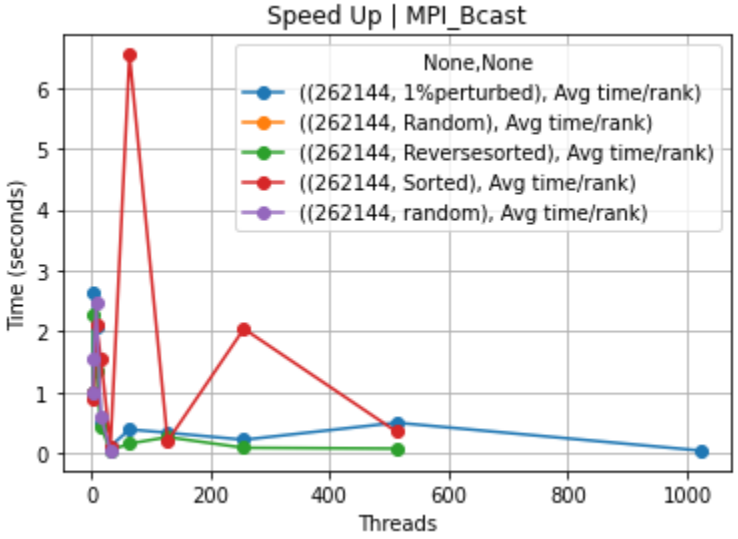

Speed Up | MPI_Barrier
Time (seconds)
Threads
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)

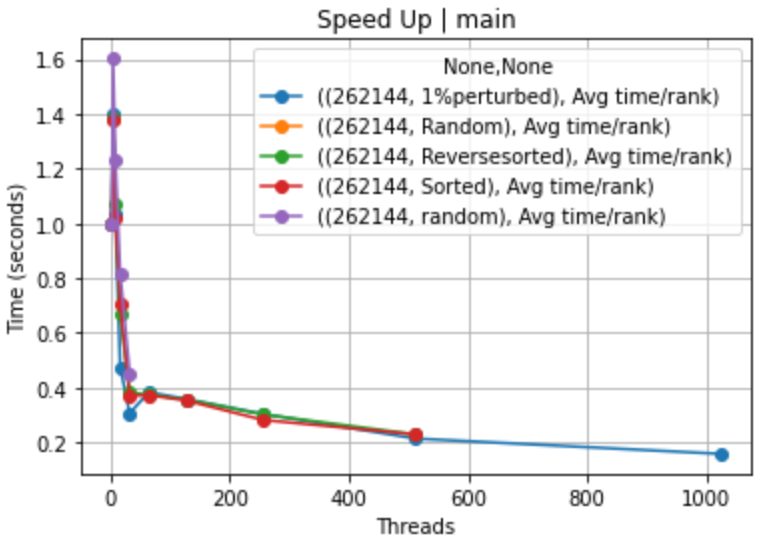

Speed Up | correctness_check
Time (seconds)
Threads
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)

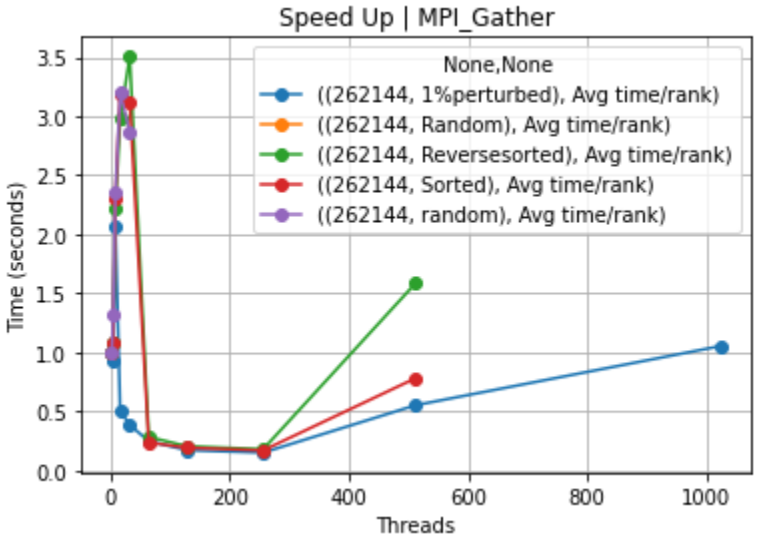

Speed Up | MPI_Sendrecv
Time (seconds)
Threads
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)


Speed Up | comp_large
Time (seconds)
Threads
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)

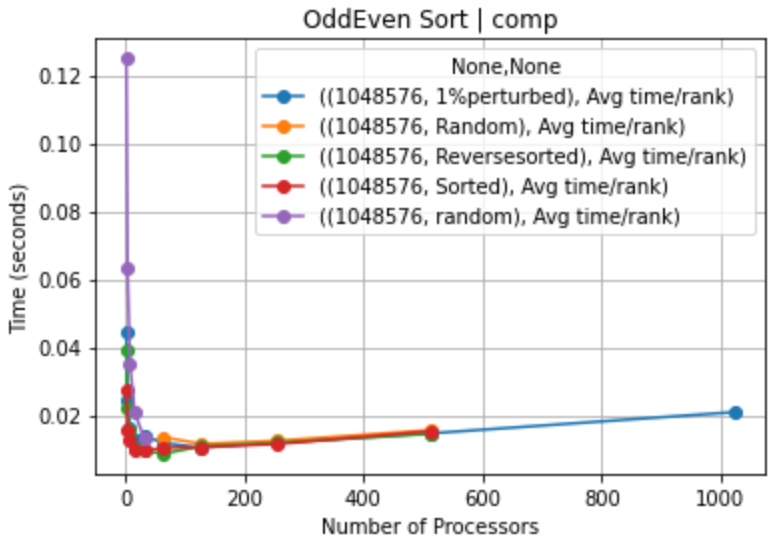

Speed Up | comm
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Threads
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

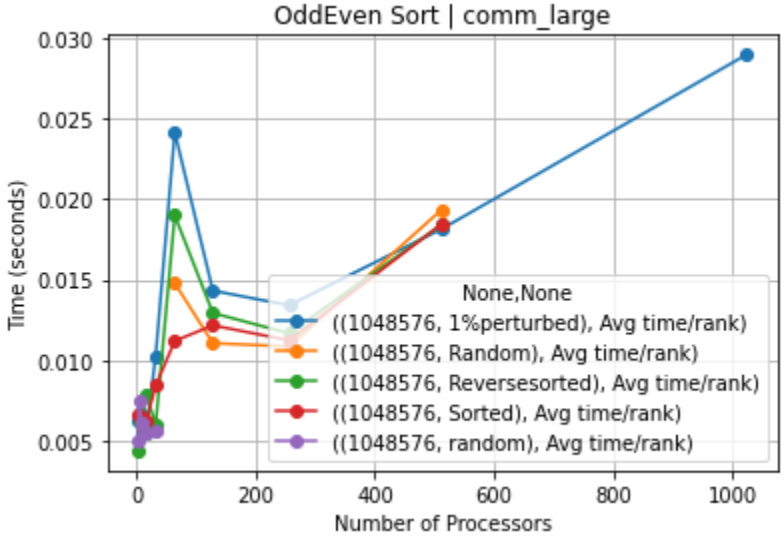

Speed Up | MPI_Bcast
None,None
((262144, 1%perturbed), Avg time/rank)
((262144, Random), Avg time/rank)
((262144, Reversesorted), Avg time/rank)
((262144, Sorted), Avg time/rank)
((262144, random), Avg time/rank)
Time (seconds)
Threads
0
1
2
3
4
5
6
0
200
400
600
800
1000

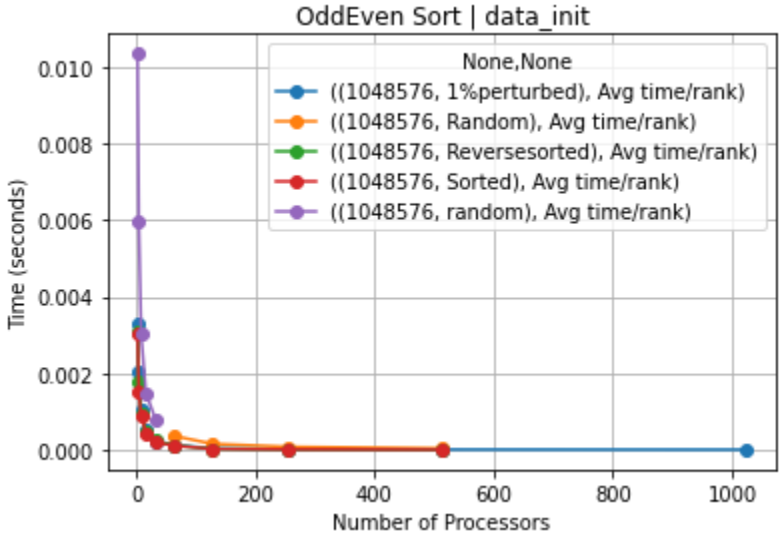


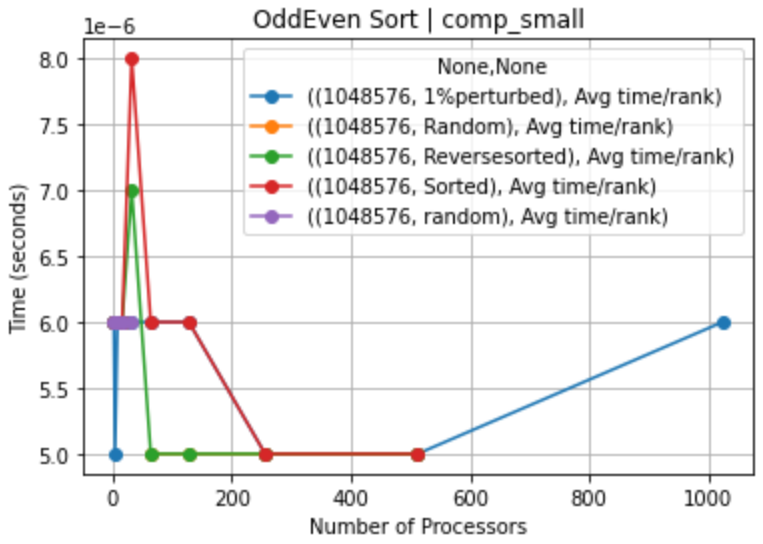


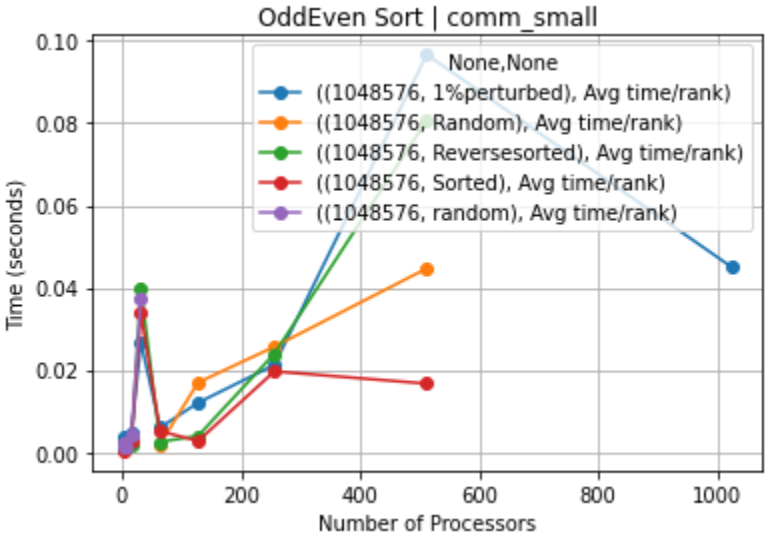
Strong Scaling 1048576

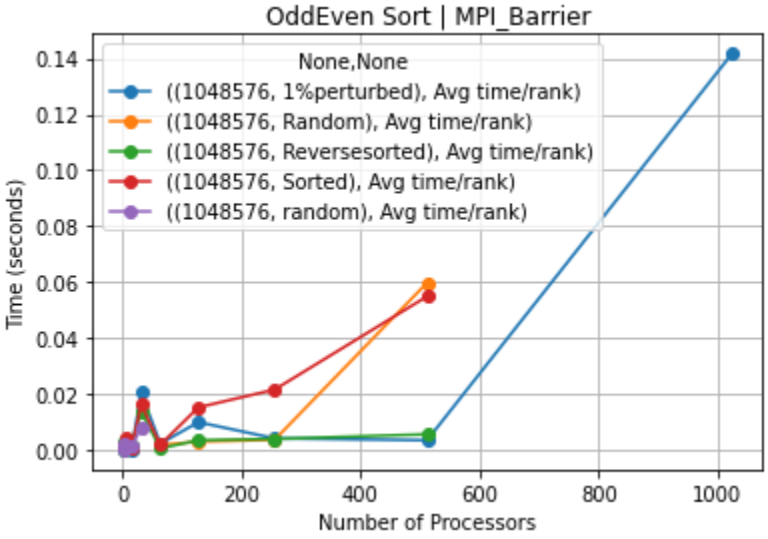

OddEven Sort | comp
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Number of Processors

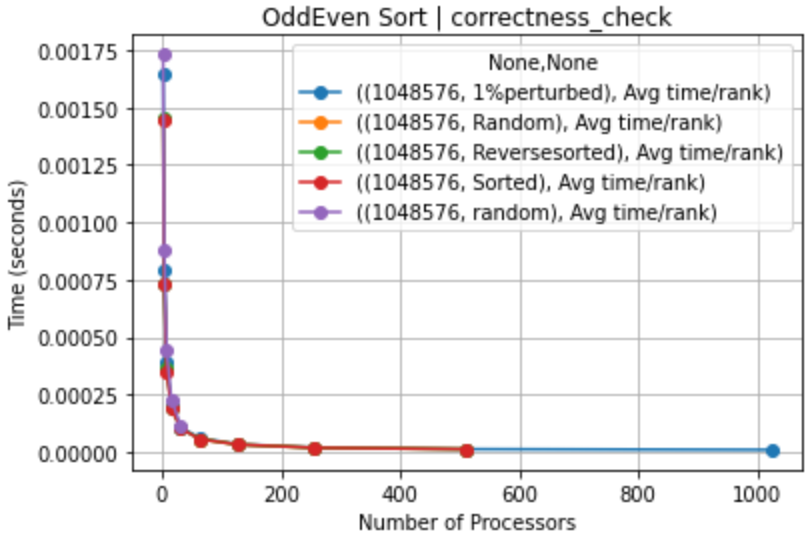

OddEven Sort | comm_large
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Number of Processors

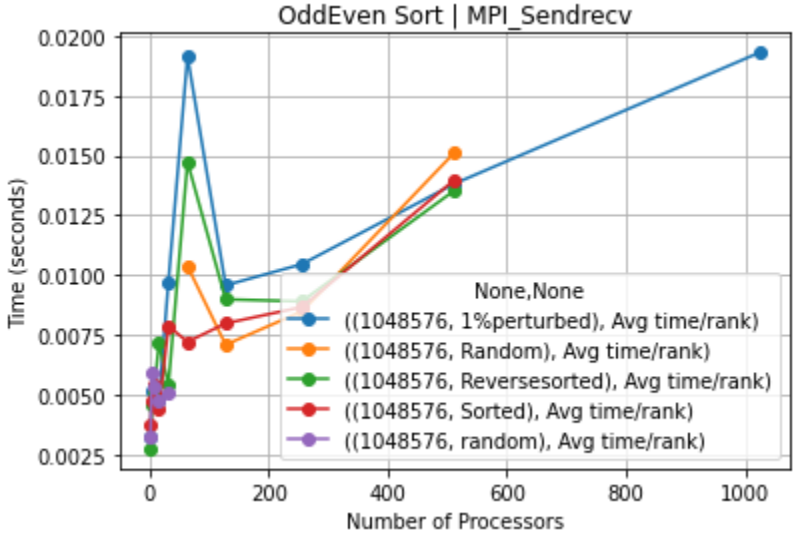

OddEven Sort | data_init
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Number of Processors

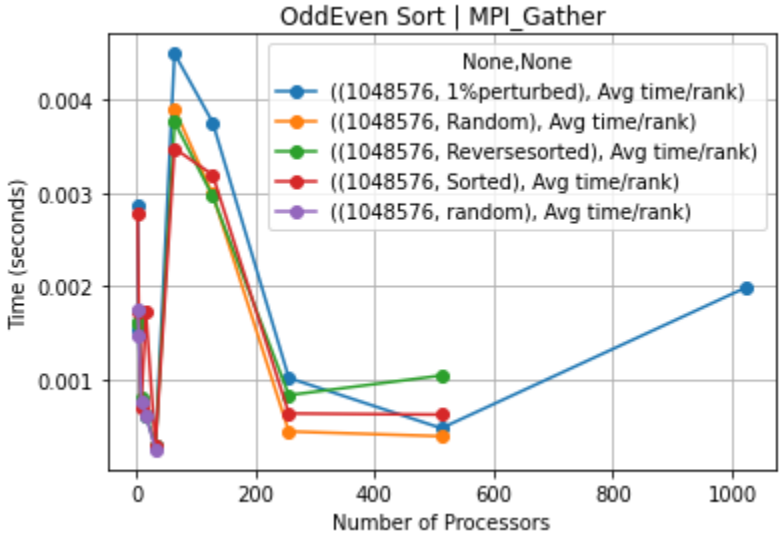

OddEven Sort | comp_small
1e−6
8.0
7.5
7.0
6.5
6.0
5.5
5.0
Time (seconds)
0
200
400
600
800
1000
Number of Processors
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

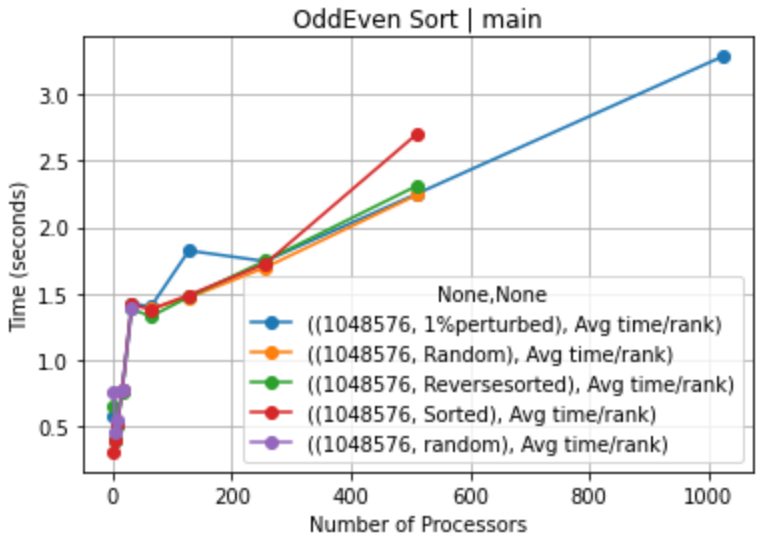

OddEven Sort | comm_small
0.10
0.08
0.06
0.04
0.02
0.00
Time (seconds)
0
200
400
600
800
1000
Number of Processors
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

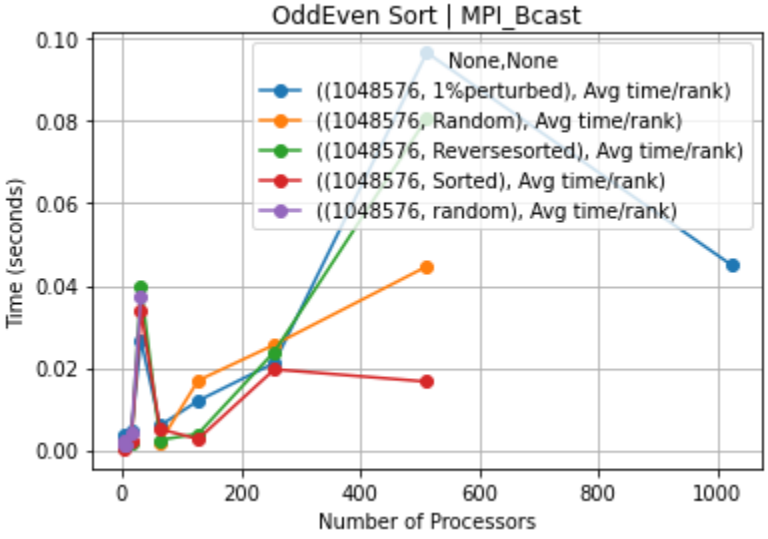

OddEven Sort | MPI_Barrier
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Time (seconds)
0
200
400
600
800
1000
Number of Processors
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

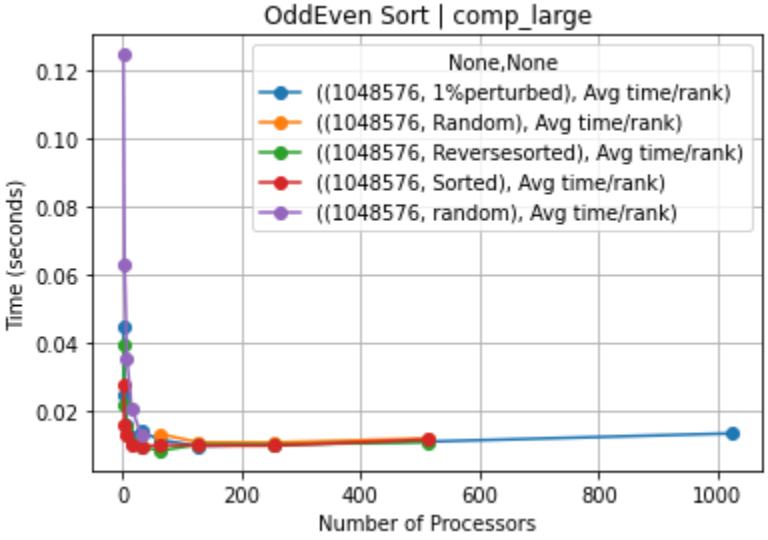

OddEven Sort | correctness_check
Time (seconds)
Number of Processors
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

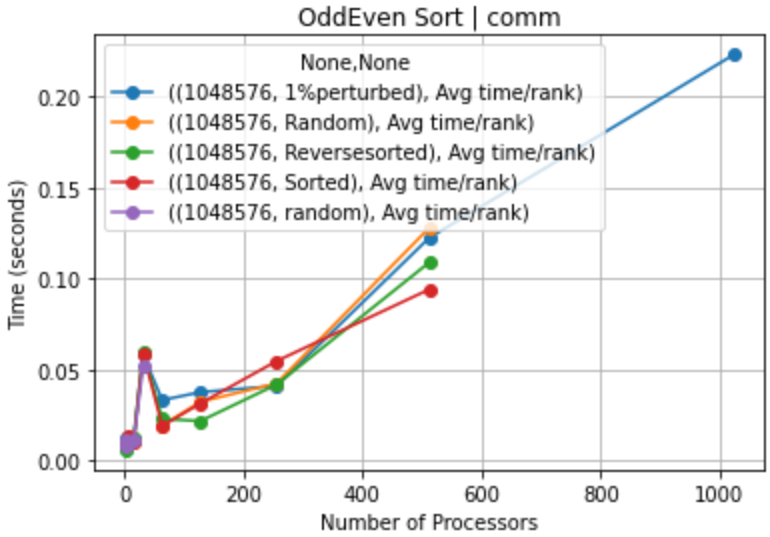

OddEven Sort | MPI_Sendrecv
Time (seconds)
Number of Processors
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

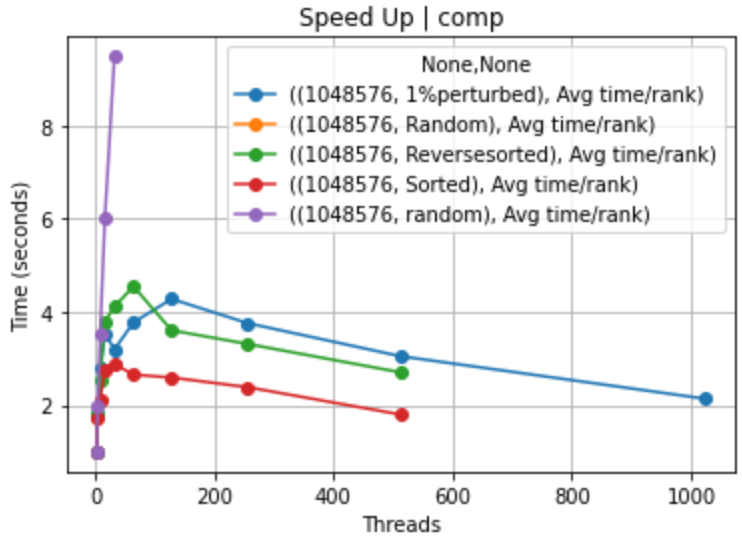

OddEven Sort | MPI_Gather
Time (seconds)
Number of Processors
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

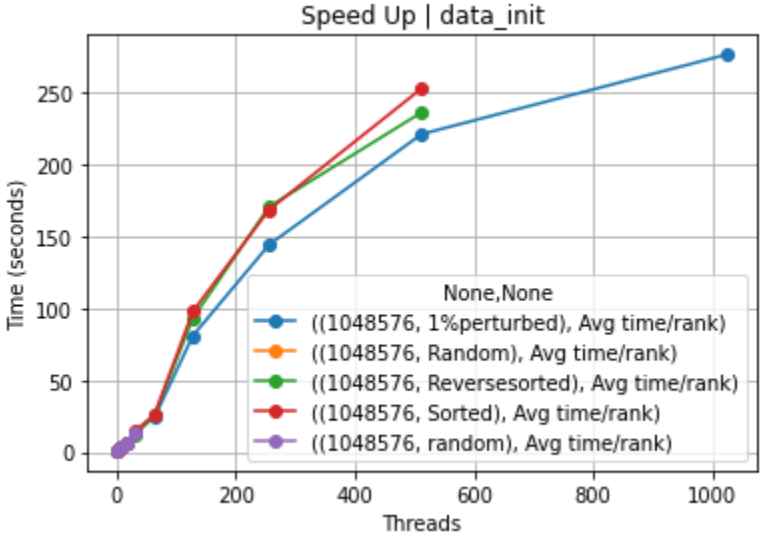

OddEven Sort | main
Time (seconds)
Number of Processors
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

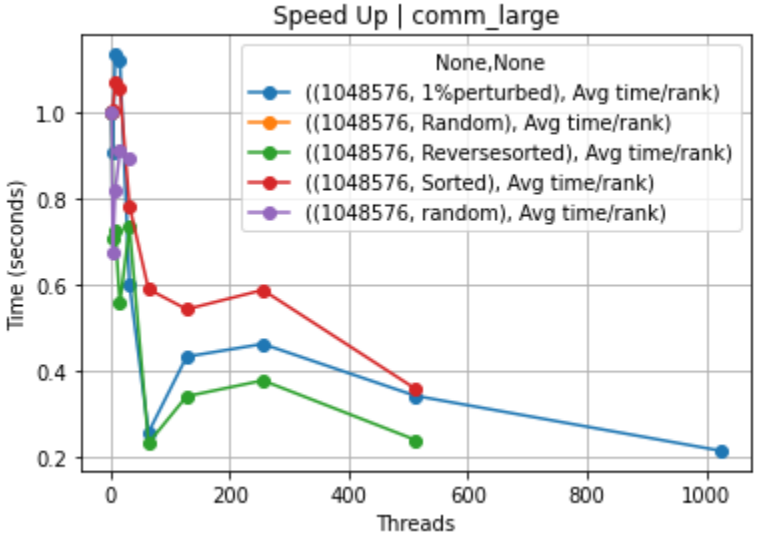

OddEven Sort | MPI_Bcast
Time (seconds)
Number of Processors
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

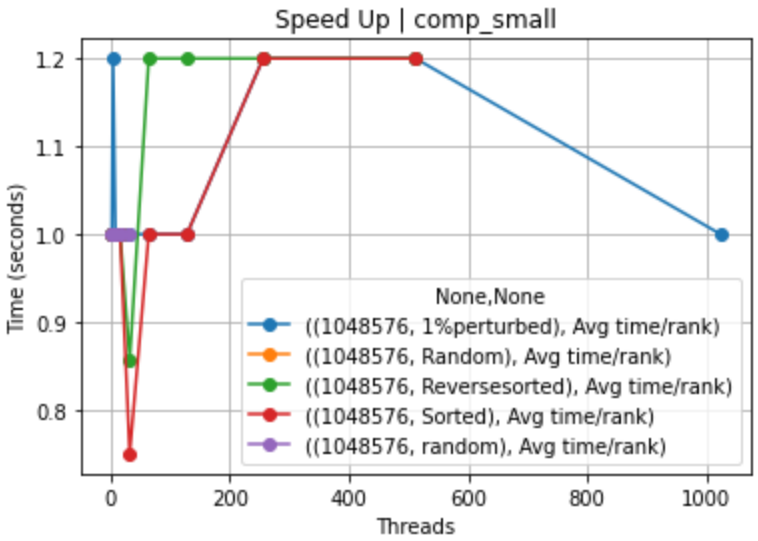

OddEven Sort | comp_large
Time (seconds)
Number of Processors
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

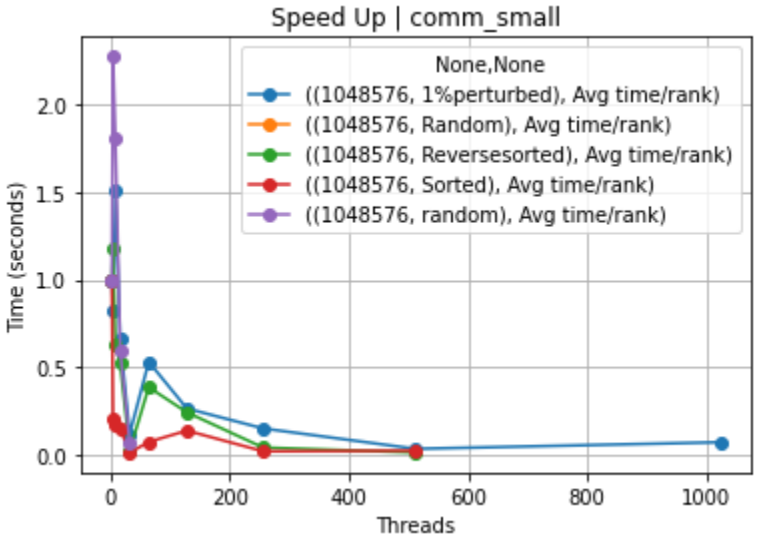

OddEven Sort | comm
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
0.00
0.05
0.10
0.15
0.20
0
200
400
600
800
1000
Number of Processors

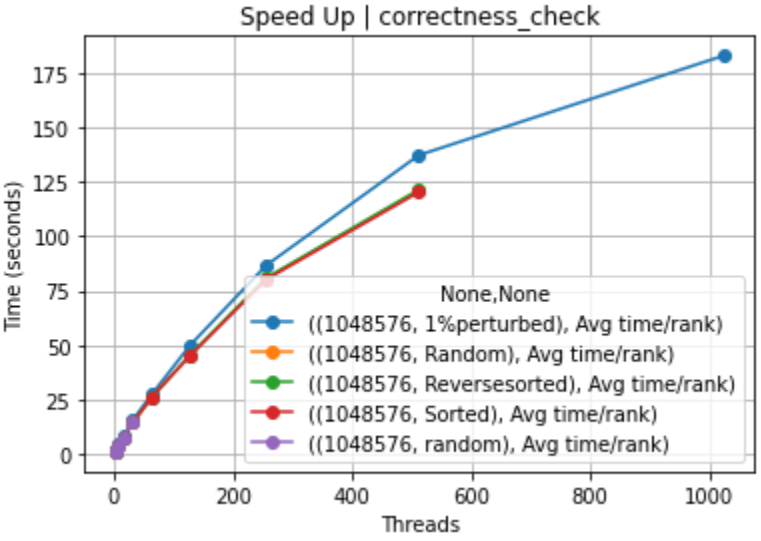

Speed Up | comp
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Threads
0
200
400
600
800
1000
2
4
6
8

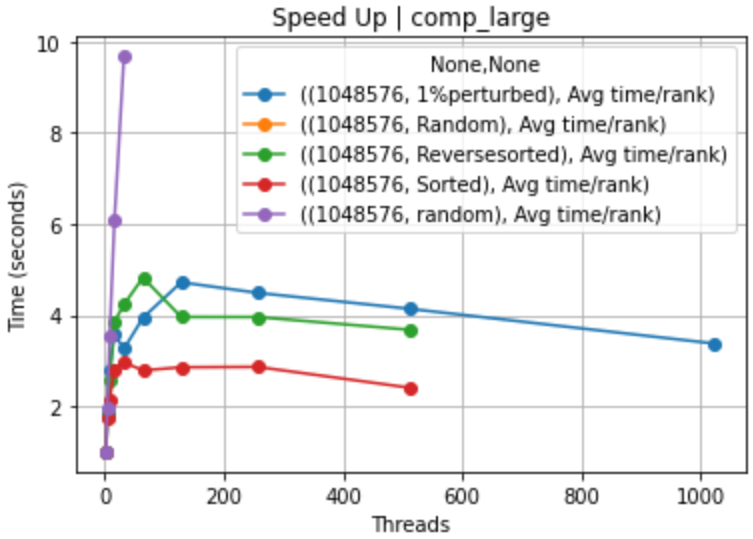

Speed Up | data_init
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Threads
0
200
400
600
800
1000
0
50
100
150
200
250

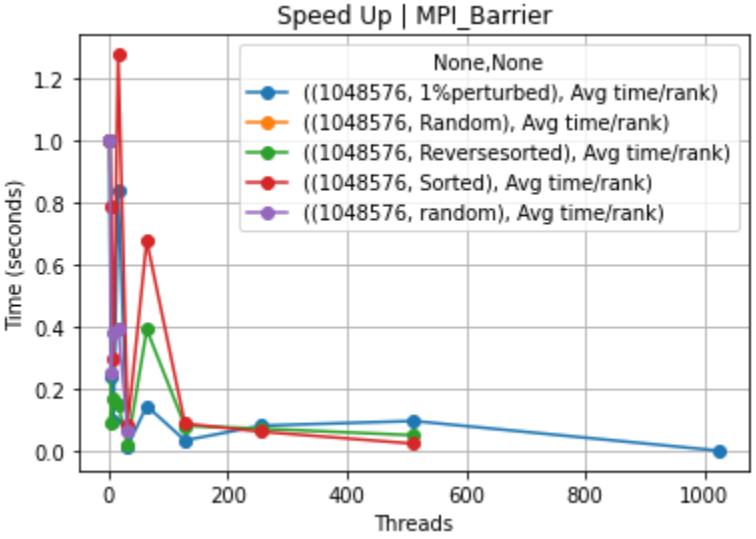

Speed Up | comm_large
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Threads
0
200
400
600
800
1000
0.2
0.4
0.6
0.8
1.0

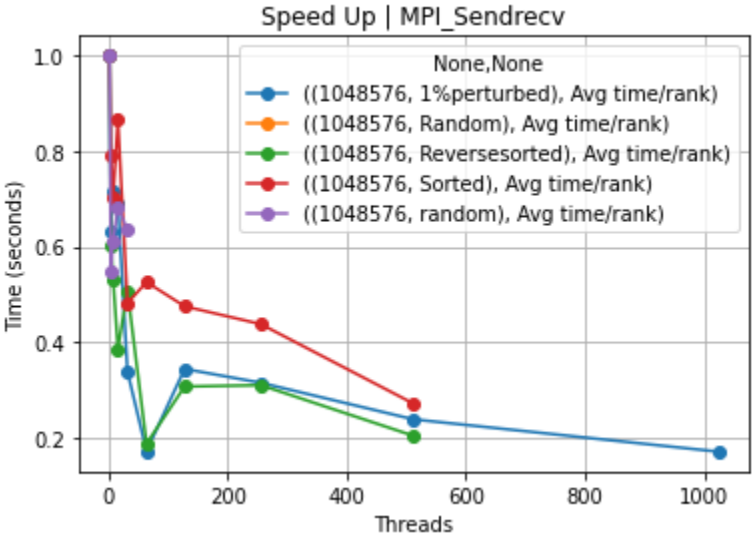

Speed Up | comp_small
Time (seconds)
Threads
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

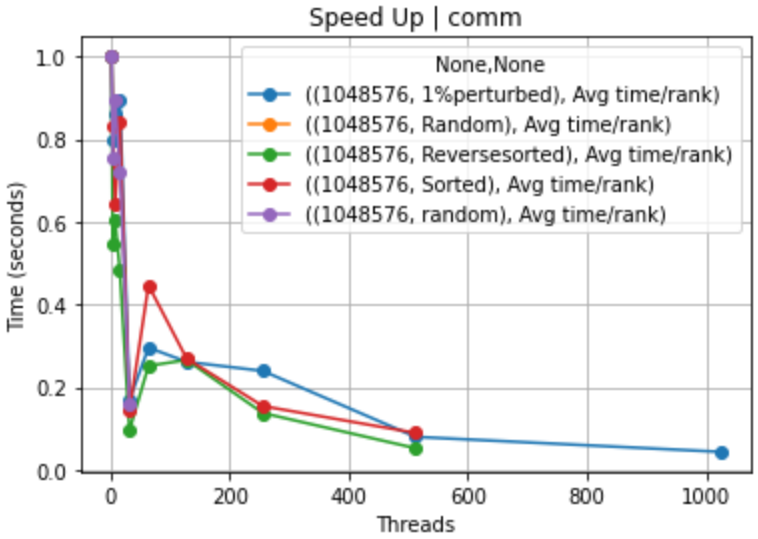

Speed Up | comm_small
Time (seconds)
Threads
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

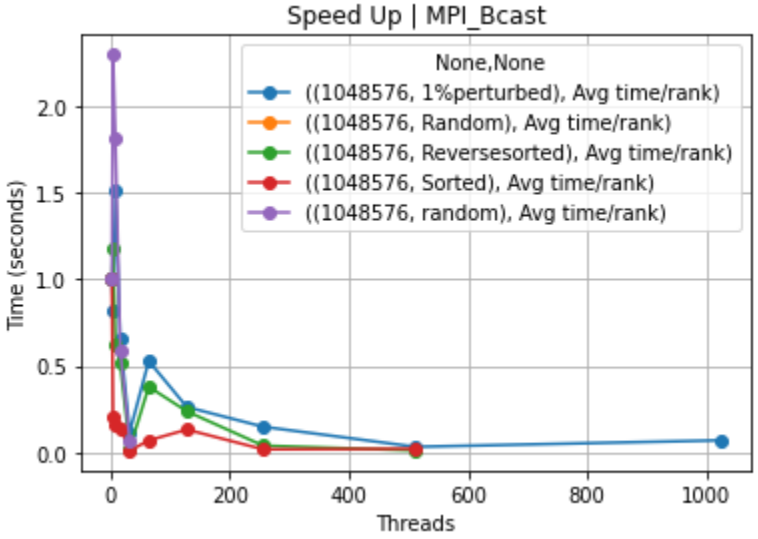

Speed Up | correctness_check
Time (seconds)
Threads
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)

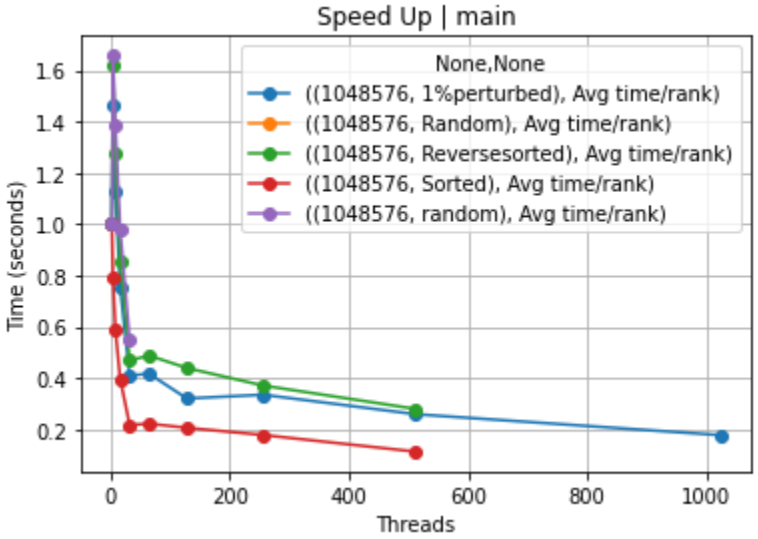

Speed Up | comp_large
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Threads

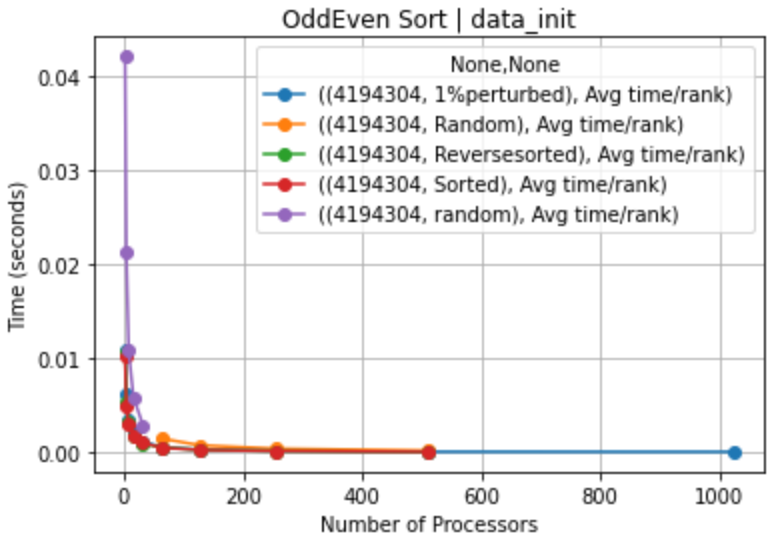

Speed Up | MPI_Barrier
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Threads

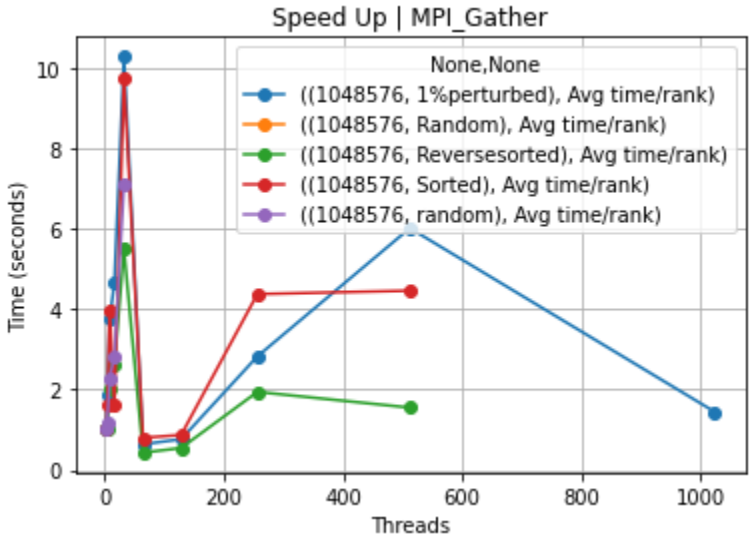

Speed Up | MPI_Sendrecv
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Threads


Speed Up | comm
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Threads
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

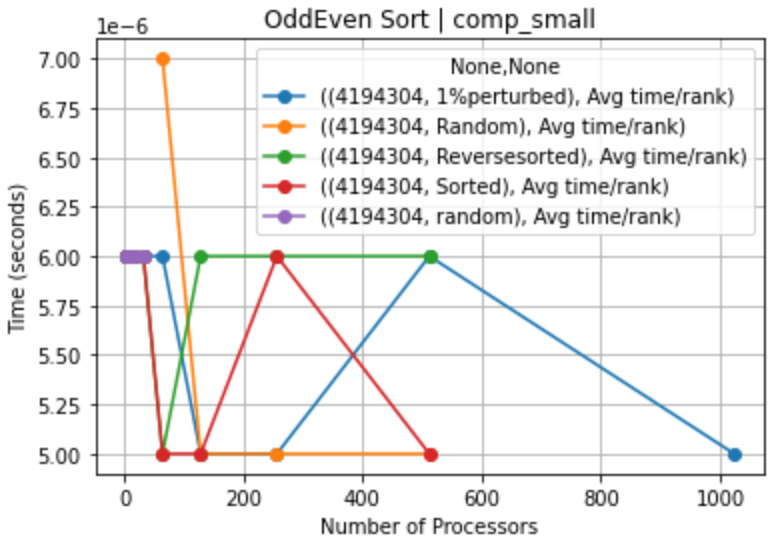

Speed Up | MPI_Bcast
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Threads
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
1000

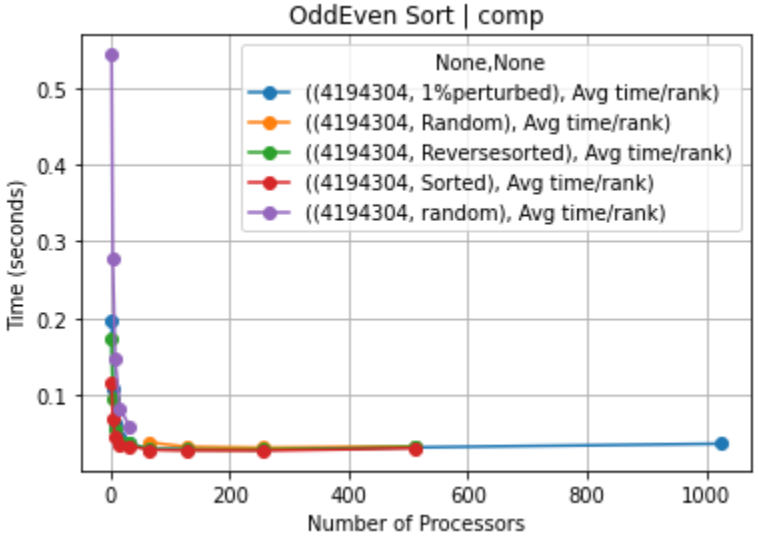

Speed Up | main
None,None
((1048576, 1%perturbed), Avg time/rank)
((1048576, Random), Avg time/rank)
((1048576, Reversesorted), Avg time/rank)
((1048576, Sorted), Avg time/rank)
((1048576, random), Avg time/rank)
Time (seconds)
Threads
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
200
400
600
800
1000

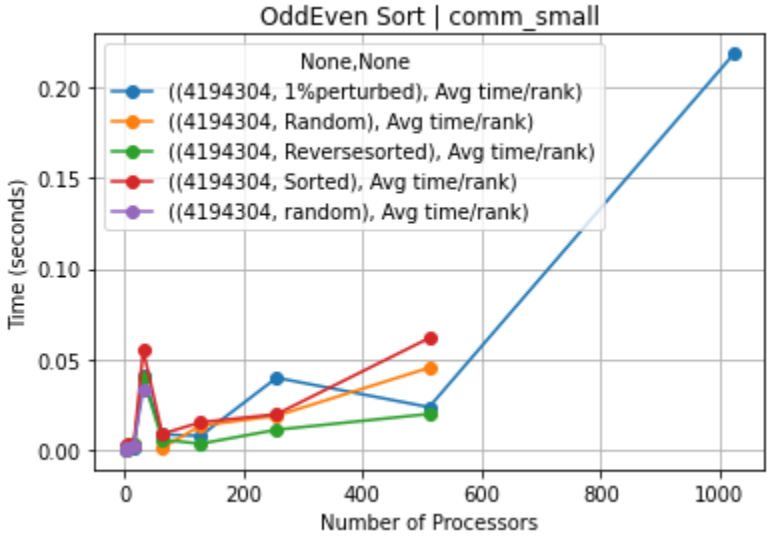


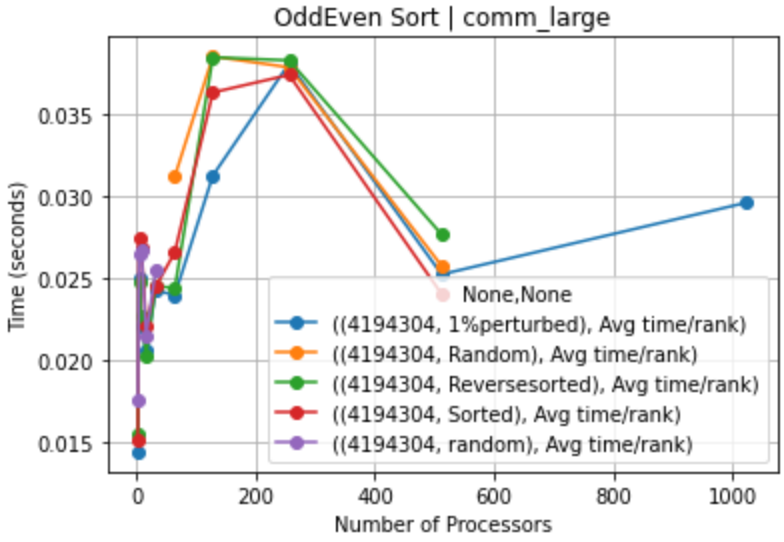


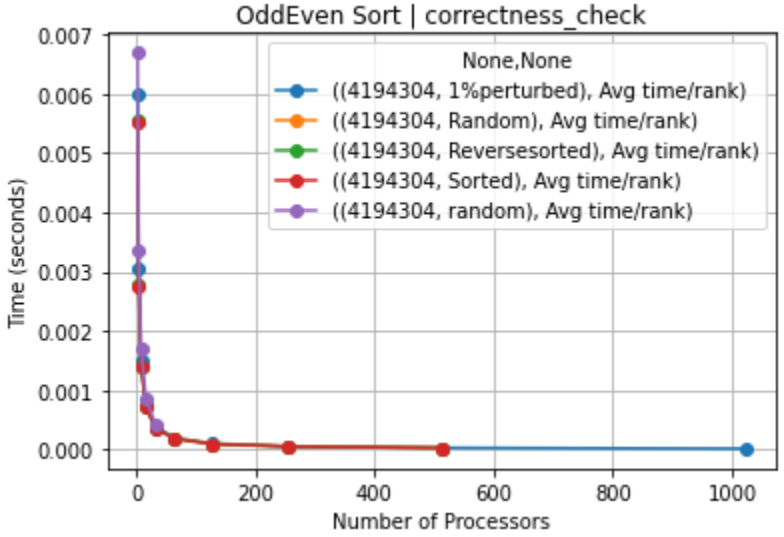
Strong Scaling 4194304

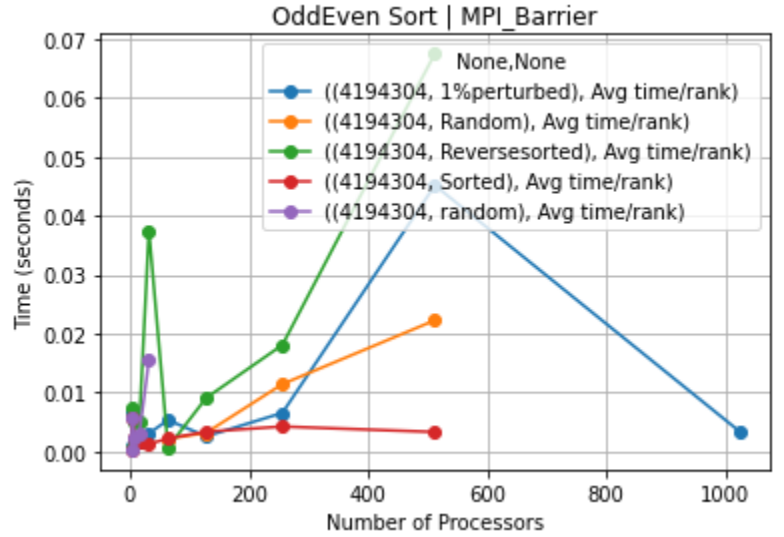

OddEven Sort | comp_small
1e−6
7.00
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
Time (seconds)
0
200
400
600
800
1000
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

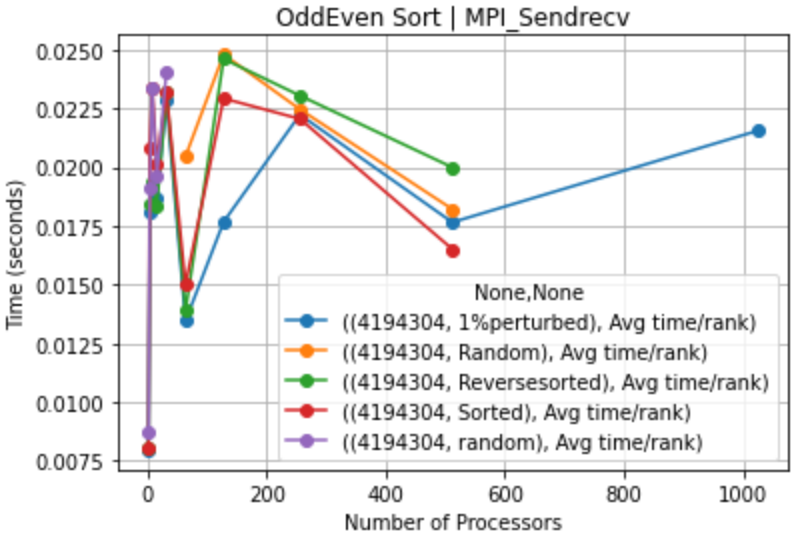

OddEven Sort | comp
0.5
0.4
0.3
0.2
0.1
Time (seconds)
0
200
400
600
800
1000
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

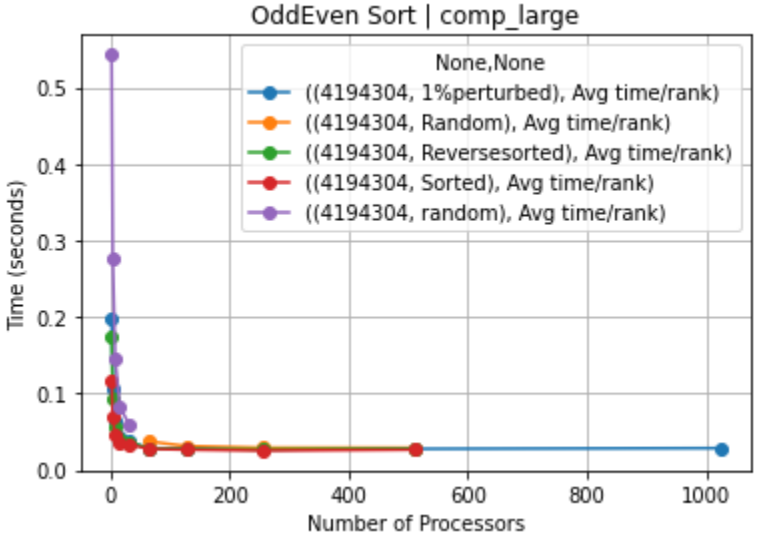

OddEven Sort | comm_small
0.20
0.15
0.10
0.05
0.00
Time (seconds)
0
200
400
600
800
1000
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

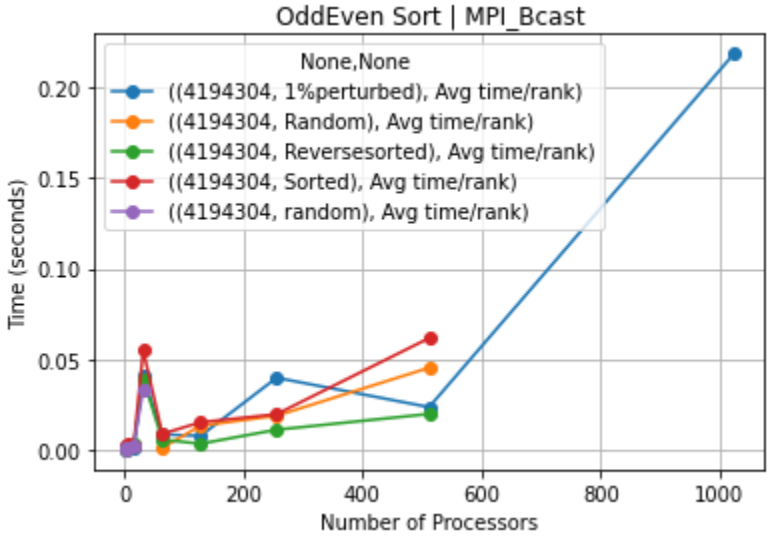

OddEven Sort | comm_large
Time (seconds)
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

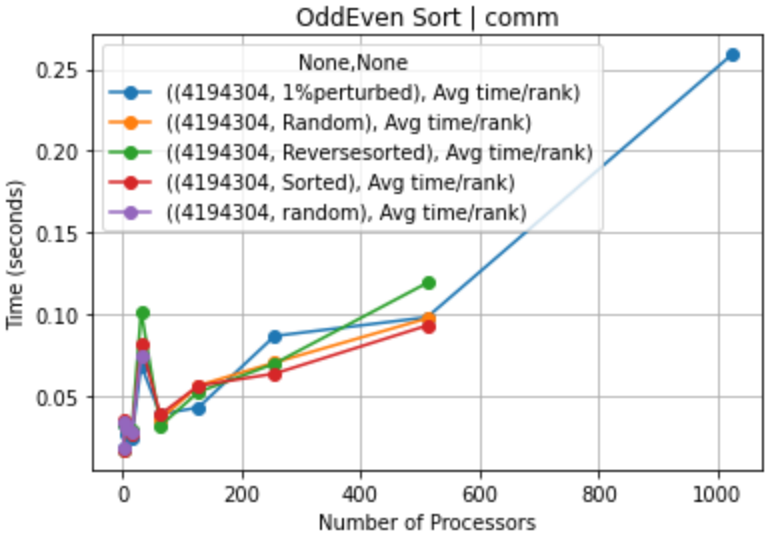

OddEven Sort | correctness_check
Time (seconds)
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

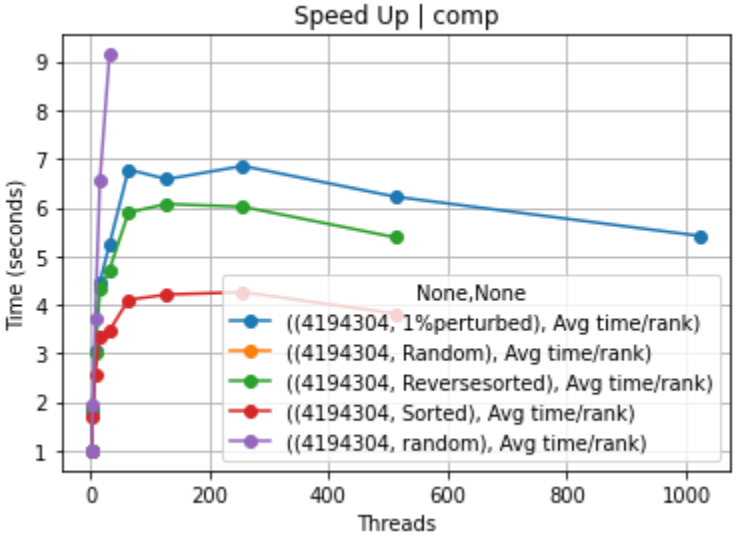

OddEven Sort | MPI_Barrier
Time (seconds)
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

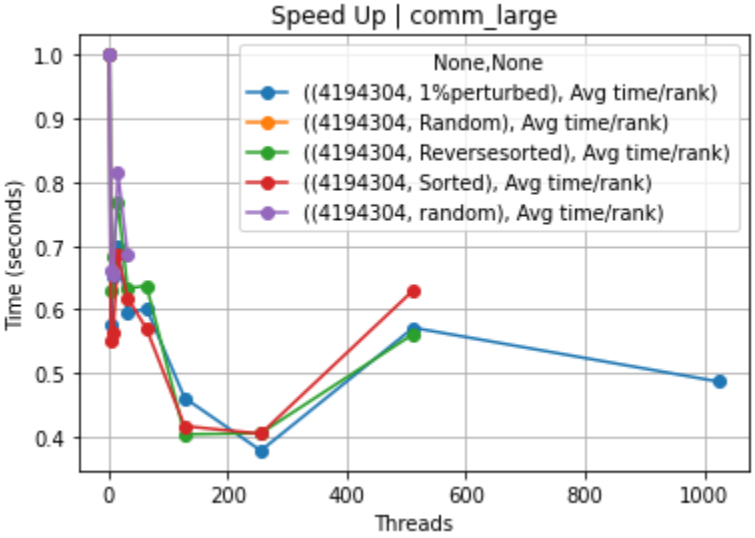

OddEven Sort | MPI_Sendrecv
Time (seconds)
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

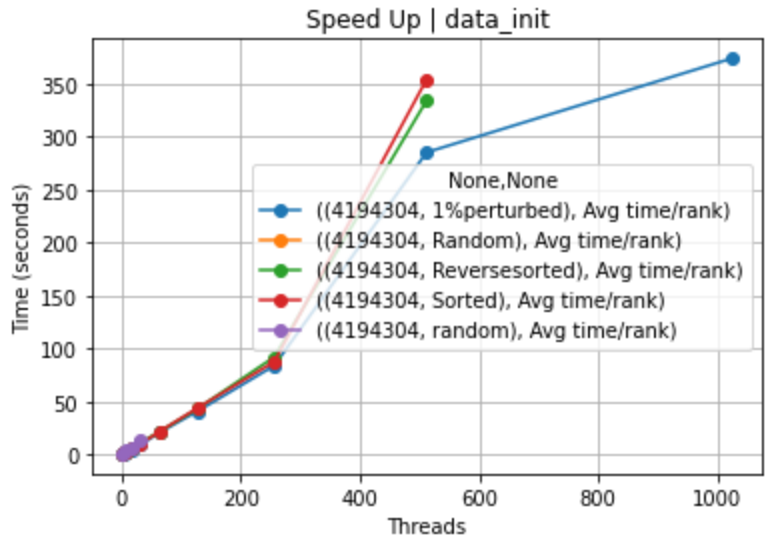

OddEven Sort | comp_large
Time (seconds)
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

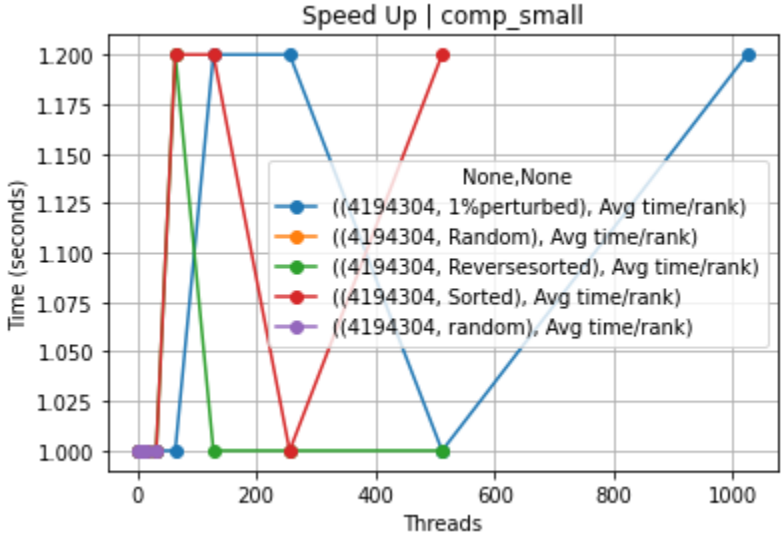

OddEven Sort | MPI_Bcast
Time (seconds)
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

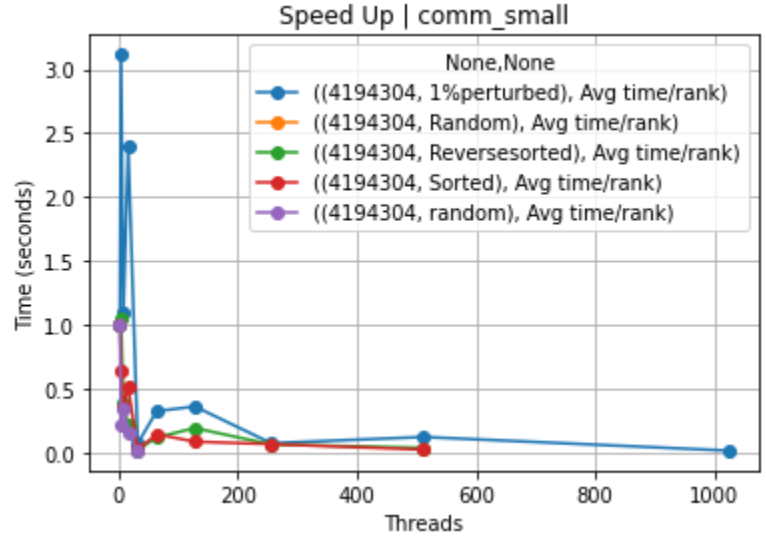

OddEven Sort | comm
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)
Time (seconds)
Number of Processors

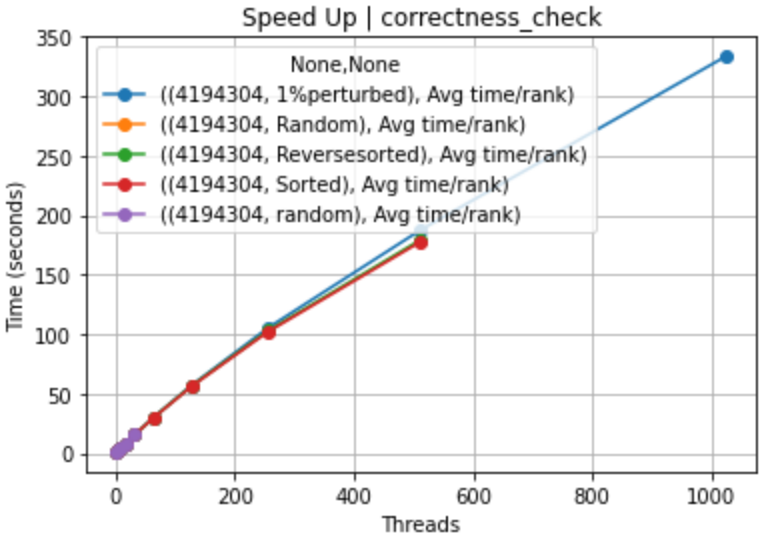

Speed Up | comp
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)
Time (seconds)
Threads

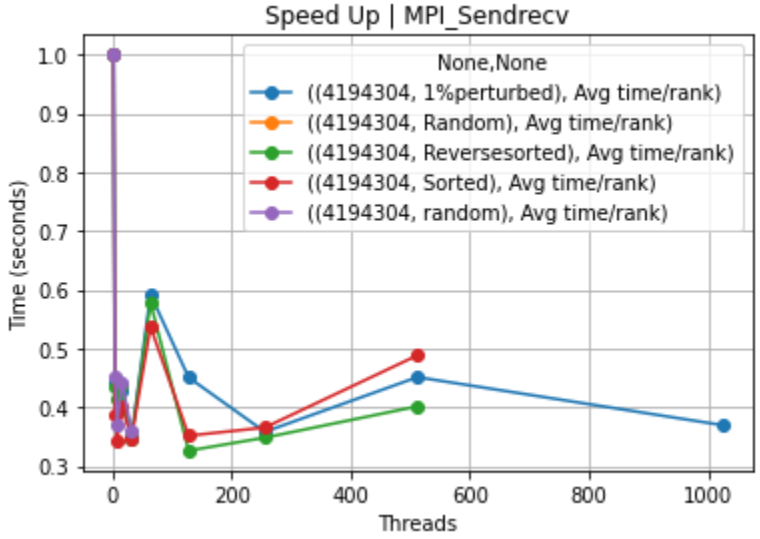

Speed Up | comm_large
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)
Time (seconds)
Threads

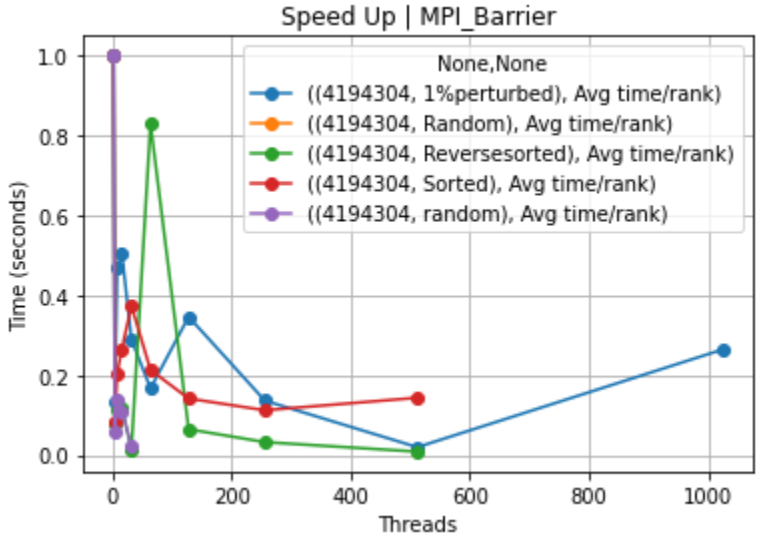

Speed Up | data_init
Time (seconds)
Threads
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

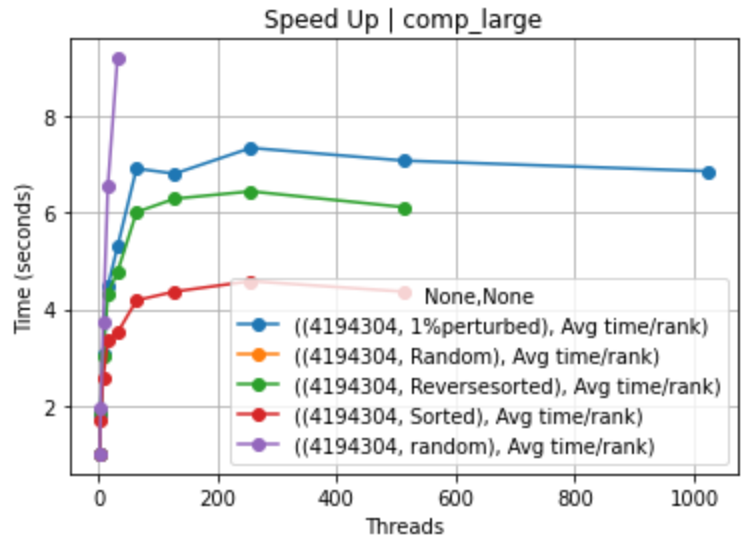

Speed Up | comp_small
Time (seconds)
Threads
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

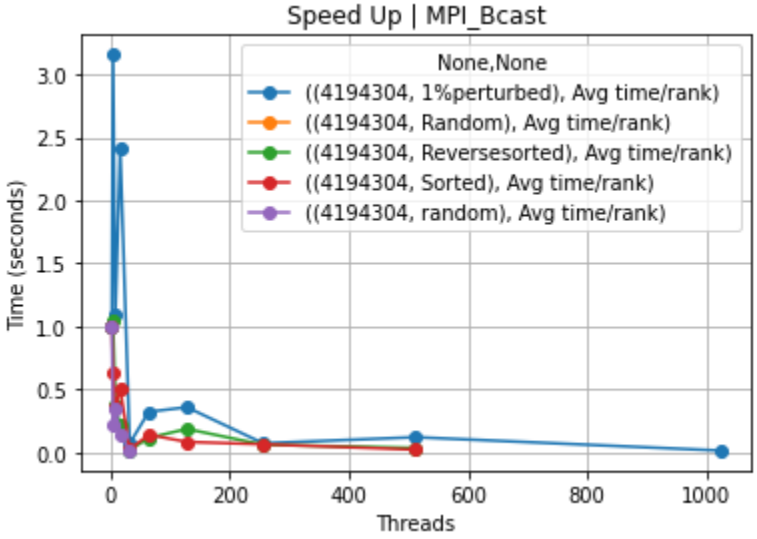

Speed Up | comm_small
Time (seconds)
Threads
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

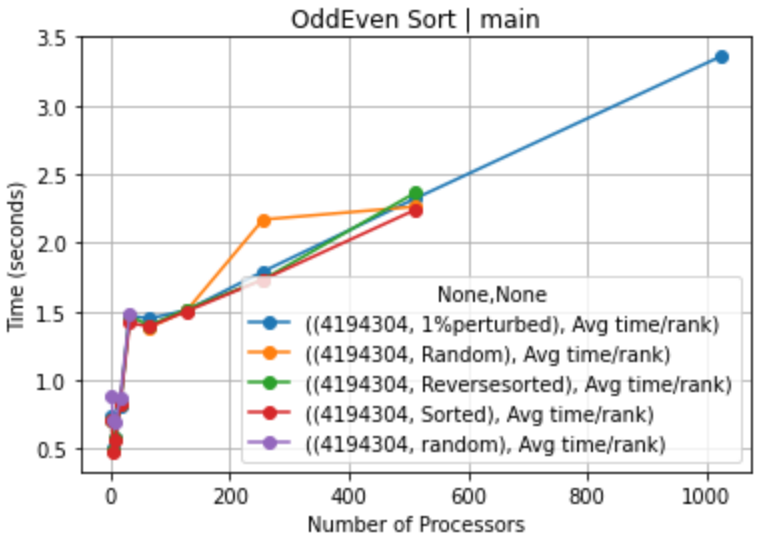

Speed Up | correctness_check
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)
Time (seconds)
Threads
0
50
100
150
200
250
300
350
0
200
400
600
800
1000

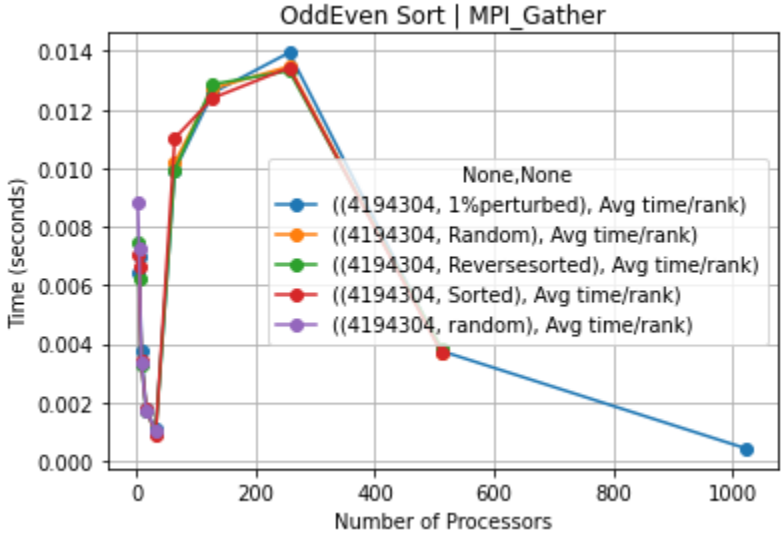

Speed Up | MPI_Sendrecv
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)
Time (seconds)
Threads
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0
200
400
600
800
1000

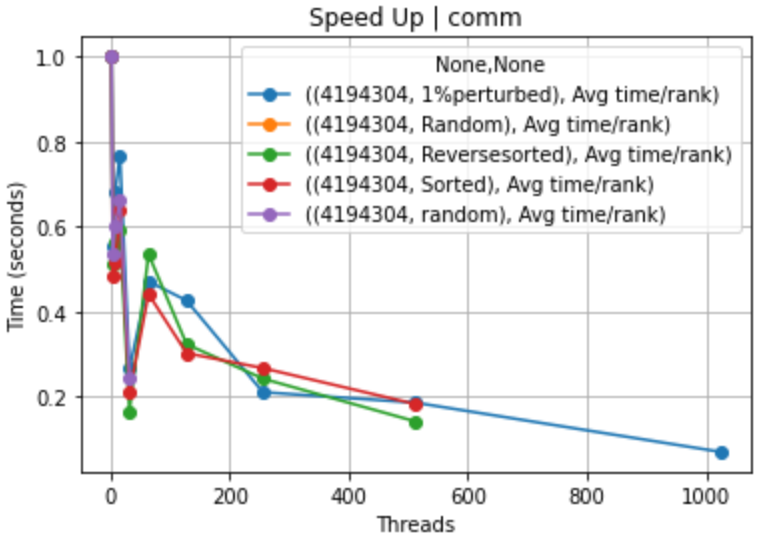

Speed Up | MPI_Barrier
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)
Time (seconds)
Threads
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

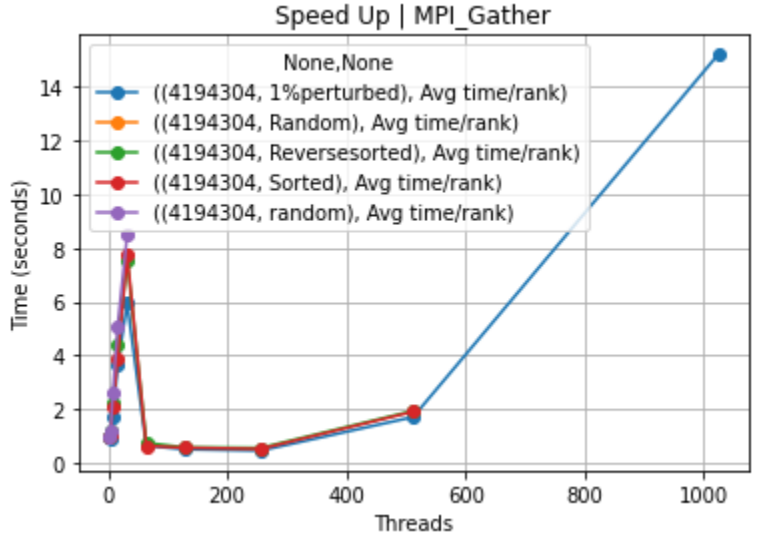

Speed Up | comp_large
Time (seconds)
Threads
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

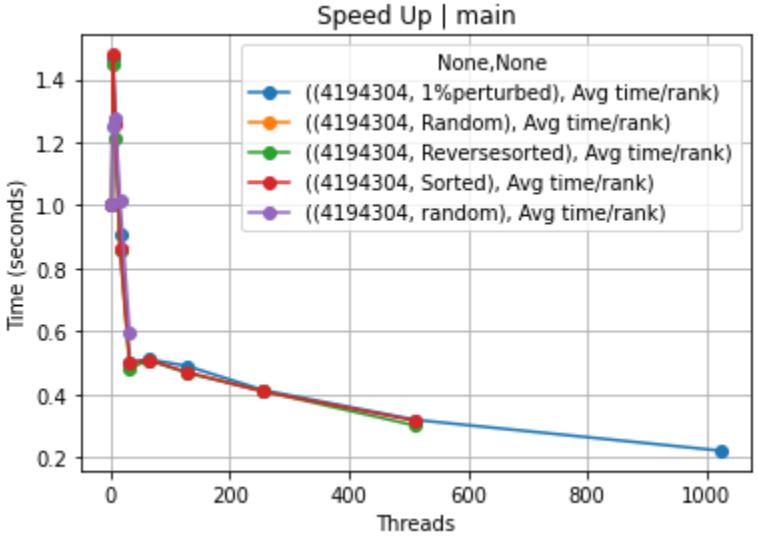

Speed Up | MPI_Bcast
Time (seconds)
Threads
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)


OddEven Sort | main
Time (seconds)
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

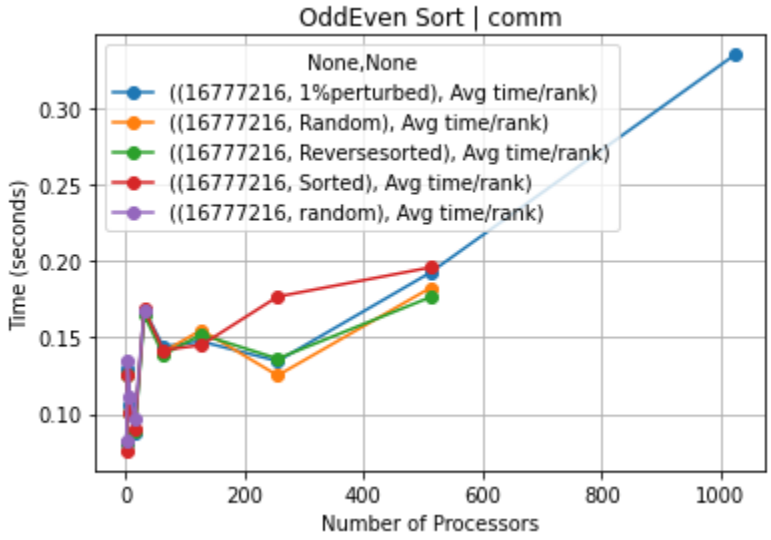

OddEven Sort | MPI_Gather
Time (seconds)
Number of Processors
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)

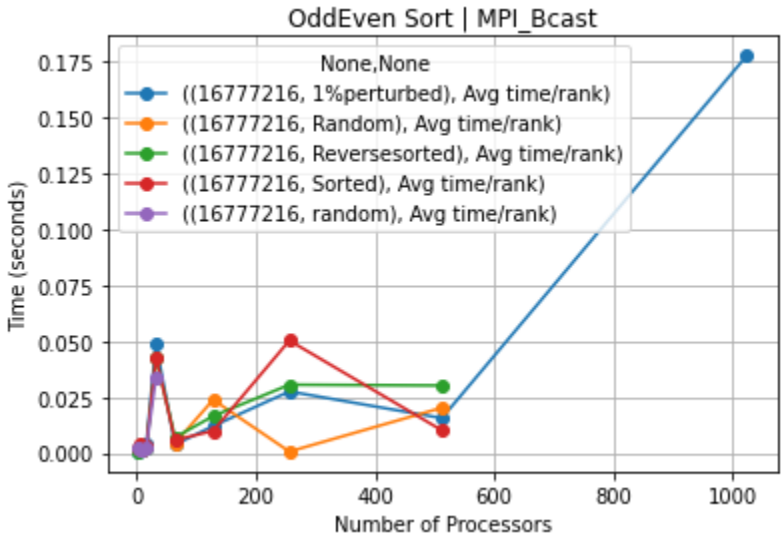

Speed Up | comm
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)
Time (seconds)
Threads
1.0
0.8
0.6
0.4
0.2
0
200
400
600
800
1000

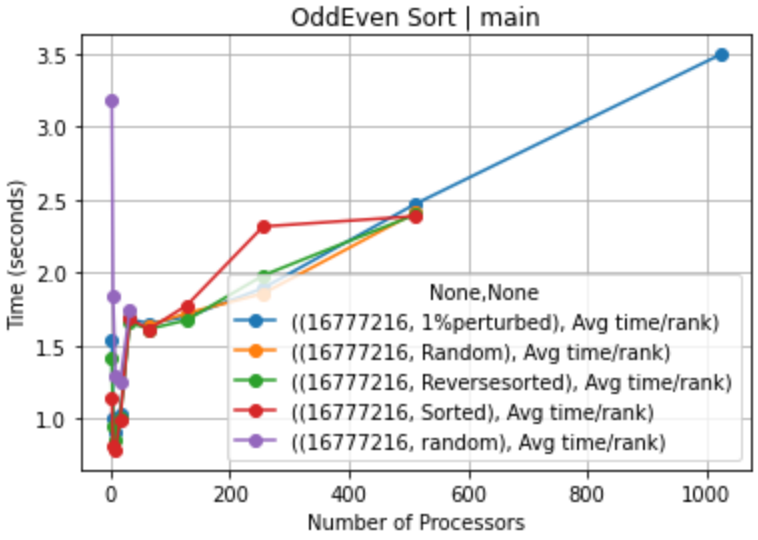

Speed Up | MPI_Gather
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)
Time (seconds)
Threads
14
12
10
8
6
4
2
0
0
200
400
600
800
1000

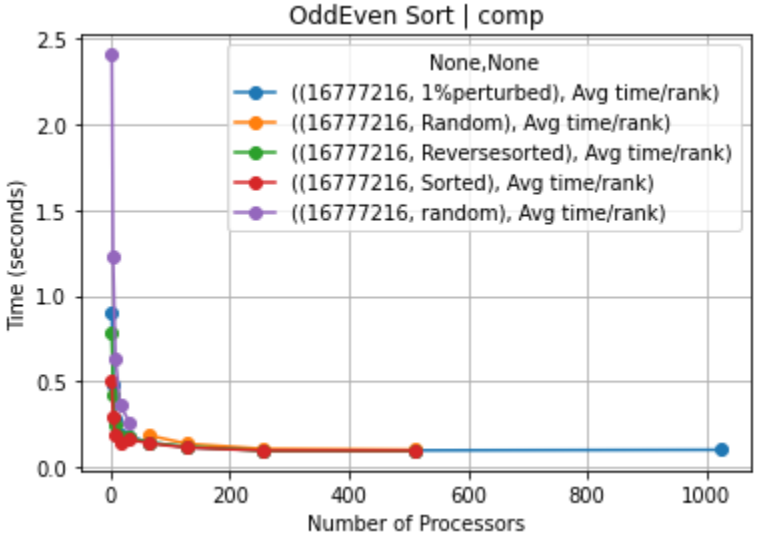

Speed Up | main
None,None
((4194304, 1%perturbed), Avg time/rank)
((4194304, Random), Avg time/rank)
((4194304, Reversesorted), Avg time/rank)
((4194304, Sorted), Avg time/rank)
((4194304, random), Avg time/rank)
Time (seconds)
Threads
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
200
400
600
800
1000

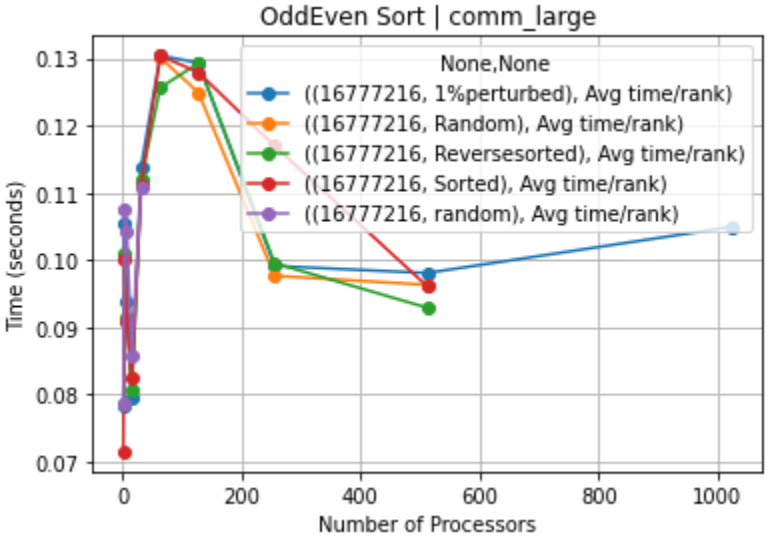
Strong Scaling 16777216

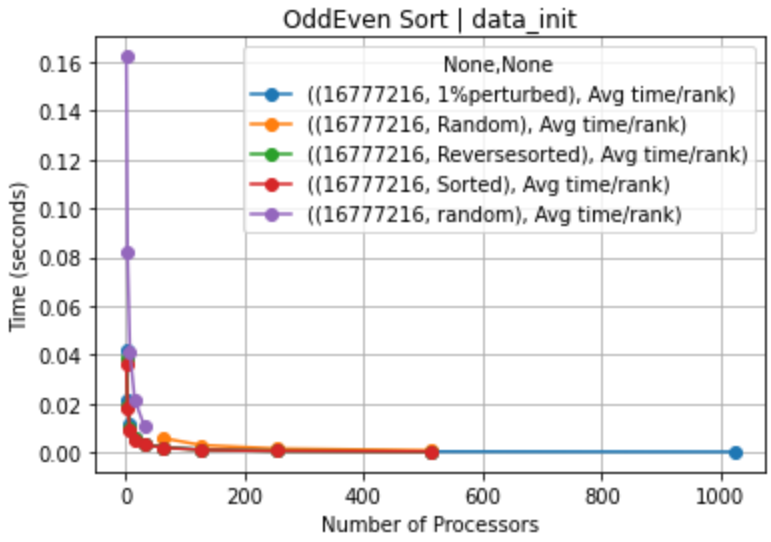


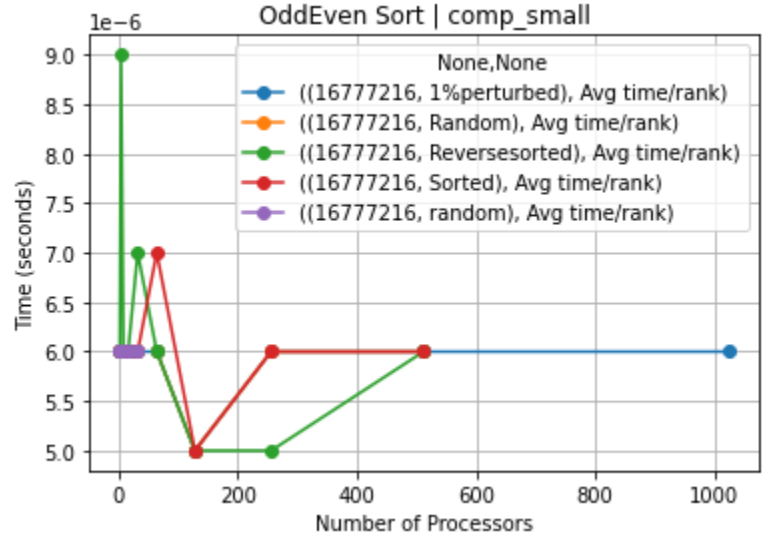


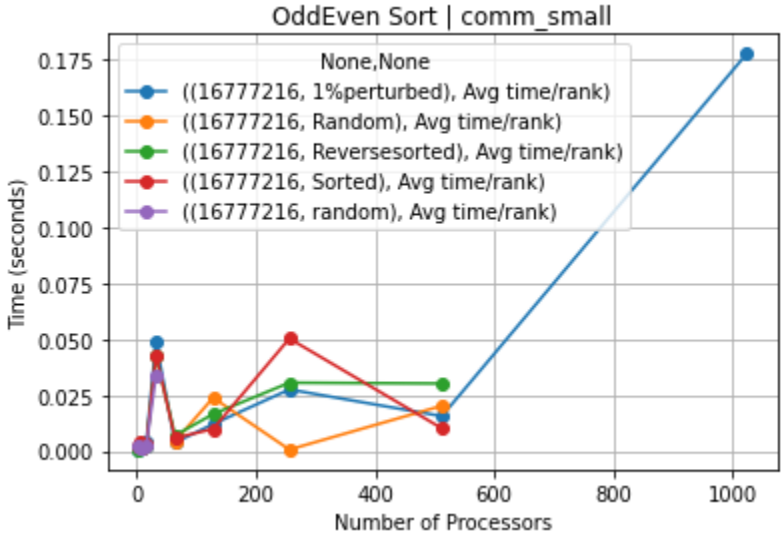

OddEven Sort | main
Time (seconds)
Number of Processors
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)

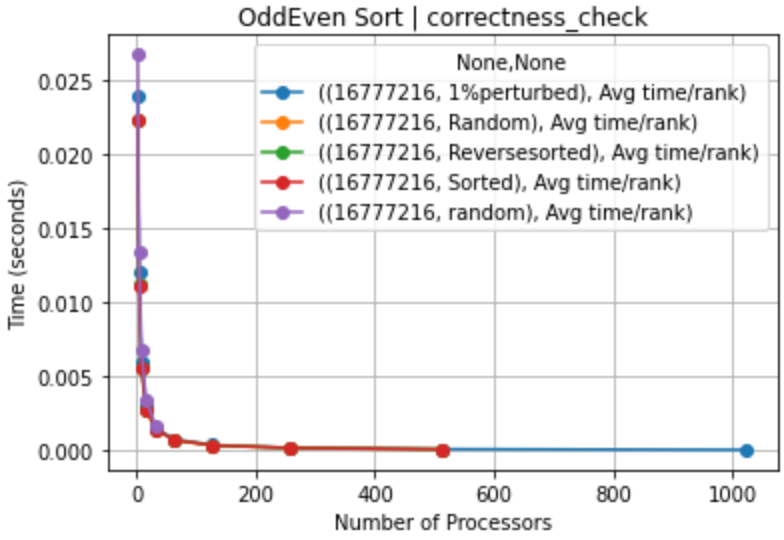

OddEven Sort | comp
Time (seconds)
Number of Processors
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)

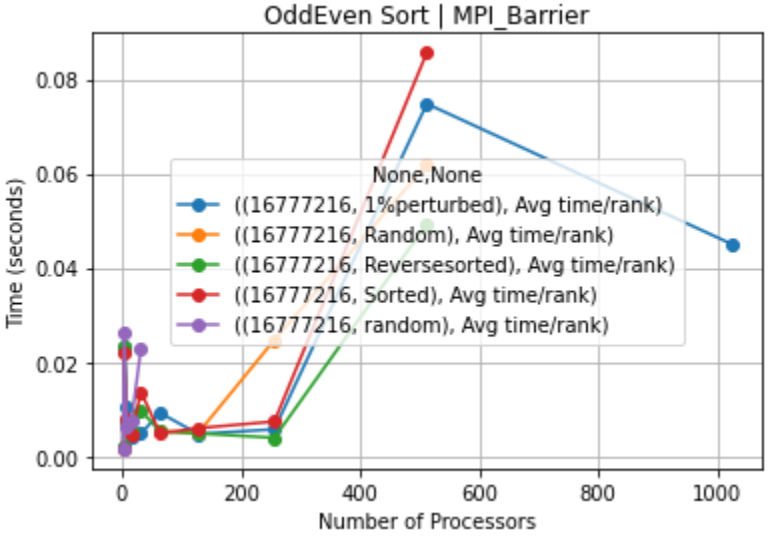

OddEven Sort | comm_large
Time (seconds)
Number of Processors
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)

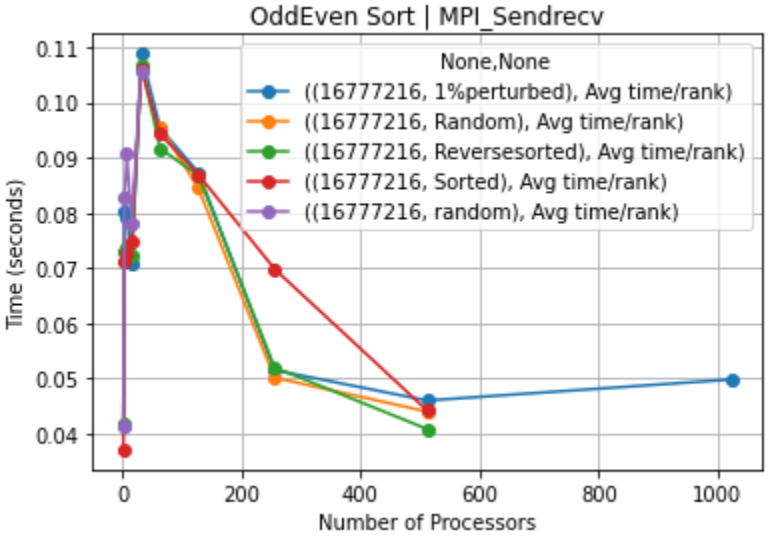

OddEven Sort | data_init
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Number of Processors

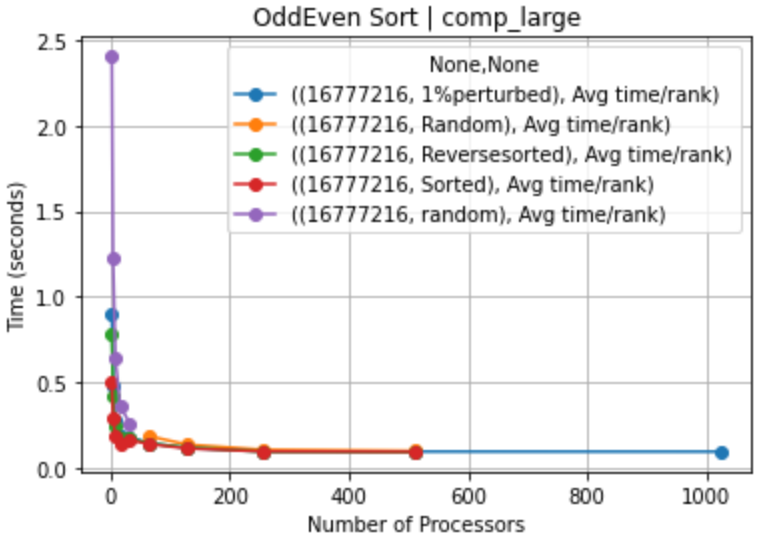

OddEven Sort | comp_small
1e−6
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Number of Processors

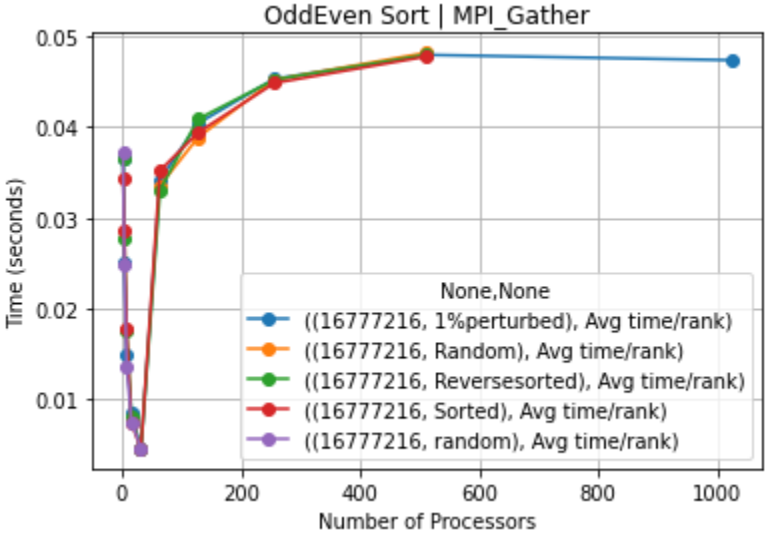

OddEven Sort | comm_small
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Number of Processors

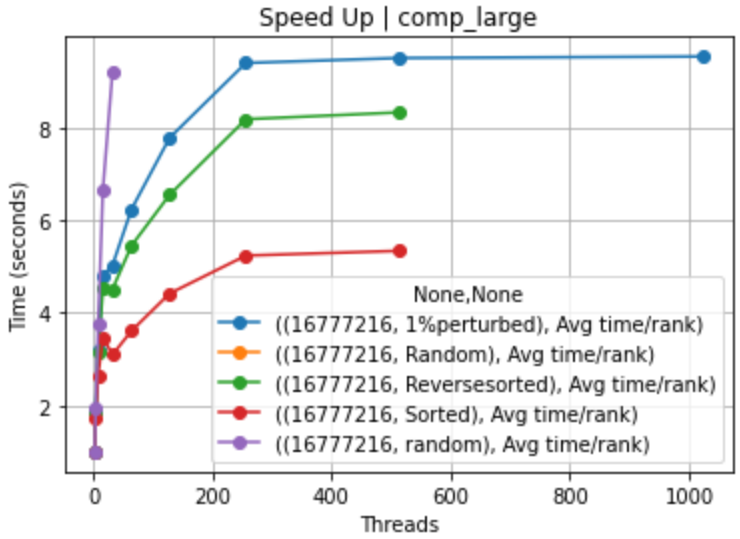

OddEven Sort | correctness_check
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Number of Processors

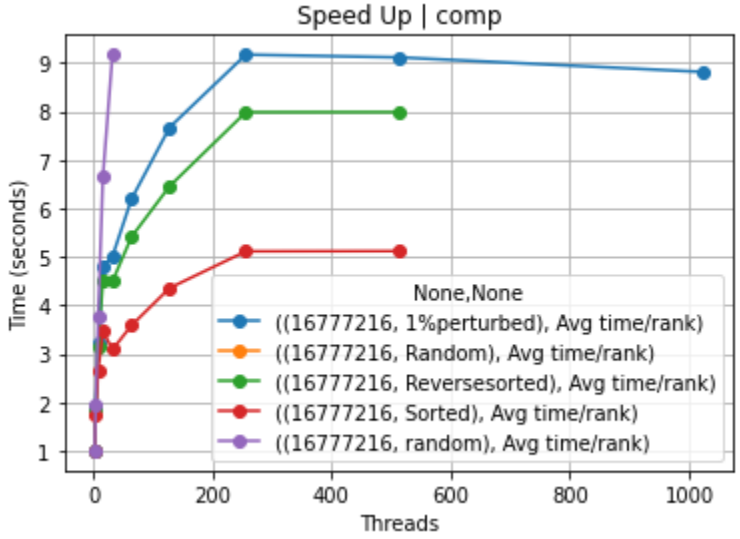

OddEven Sort | MPI_Barrier
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Number of Processors

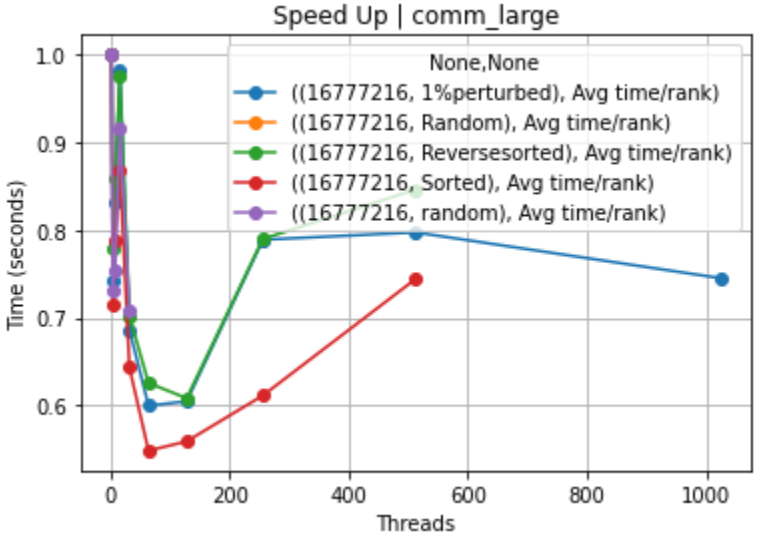

OddEven Sort | MPI_Sendrecv
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Number of Processors

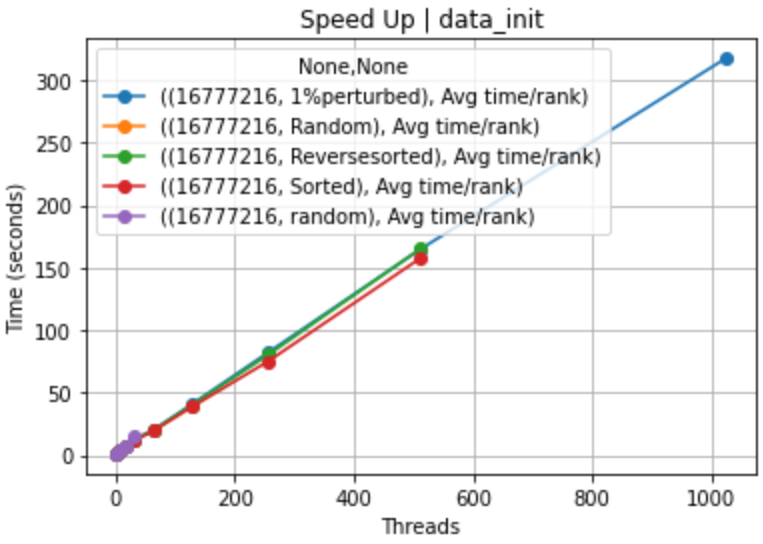

OddEven Sort | comp_large
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Number of Processors
0.0
0.5
1.0
1.5
2.0
2.5
0
200
400
600
800
1000

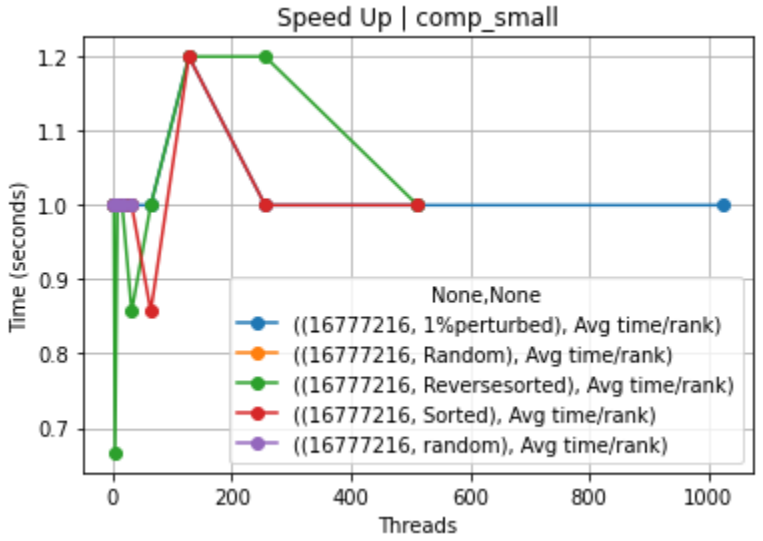

OddEven Sort | MPI_Gather
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Number of Processors
0.01
0.02
0.03
0.04
0.05
0
200
400
600
800
1000

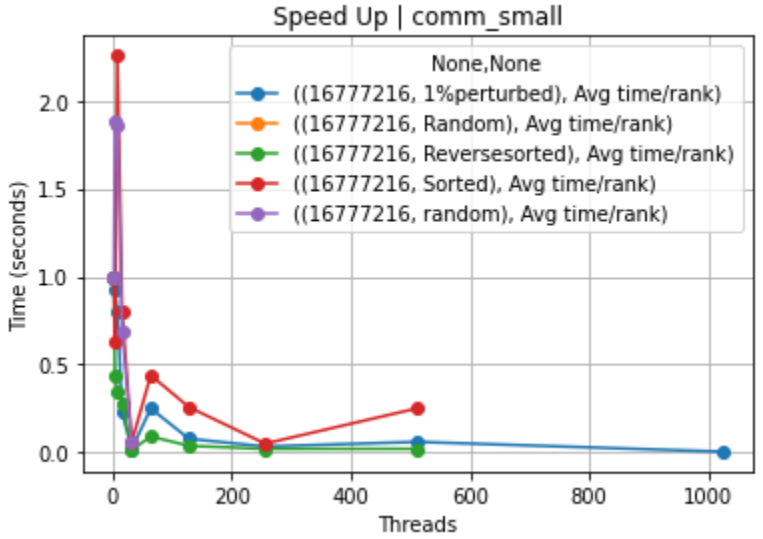

Speed Up | comp_large
Time (seconds)
Threads
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)

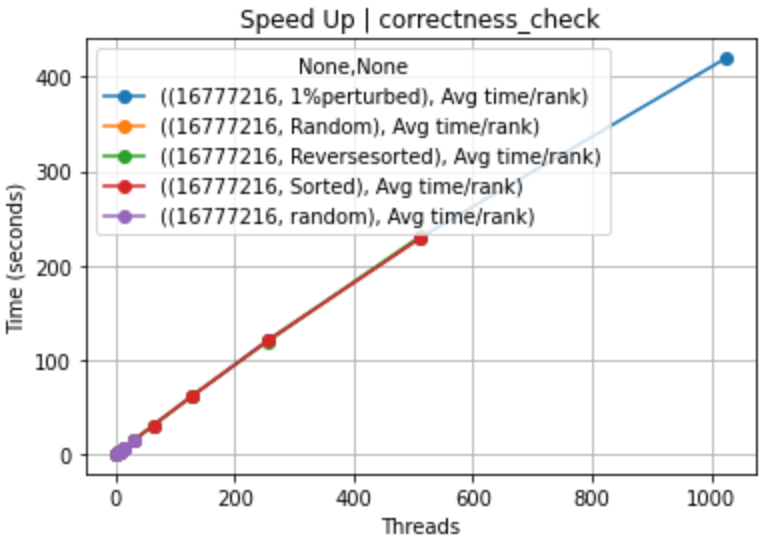

Speed Up | comp
Time (seconds)
Threads
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)

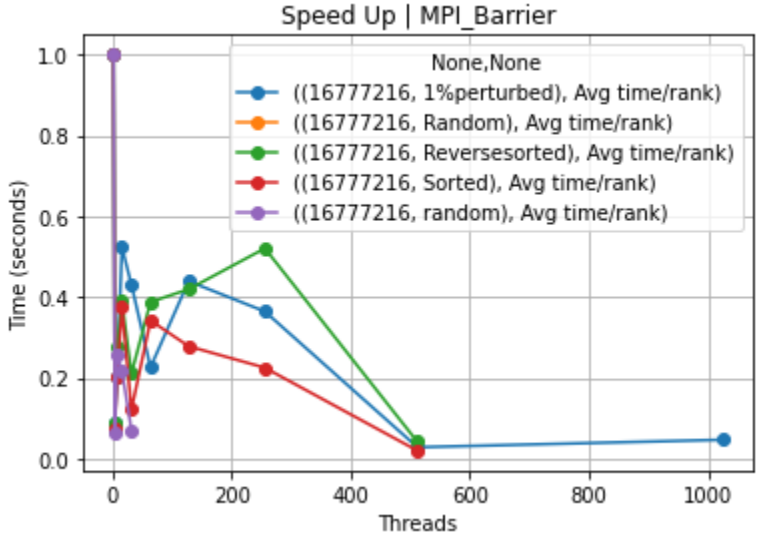

Speed Up | comm_large
Time (seconds)
Threads
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)

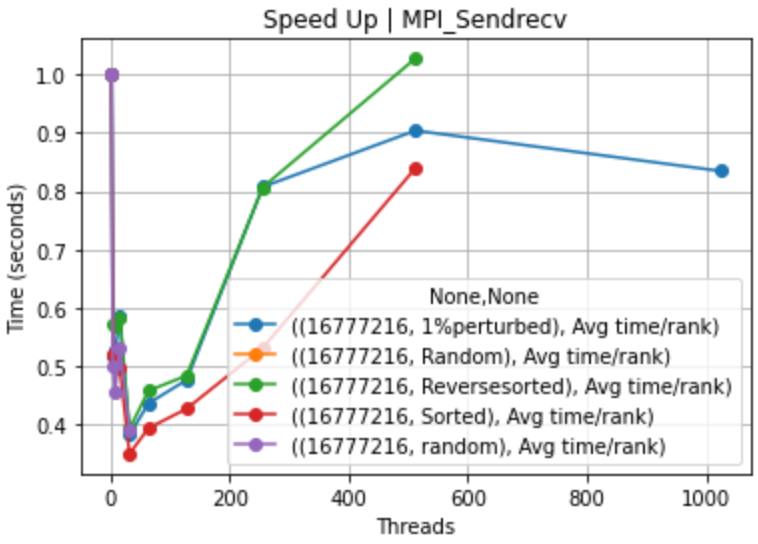

Speed Up | data_init
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Threads
0
50
100
150
200
250
300
0
200
400
600
800
1000

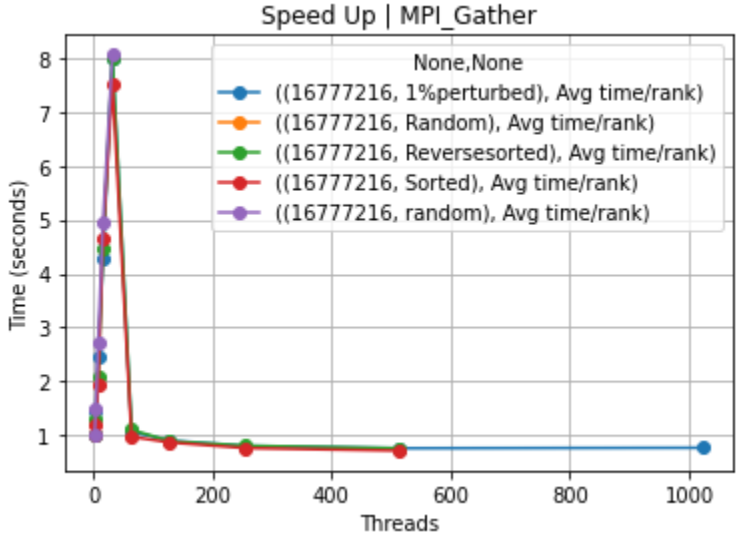

Speed Up | comp_small
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Threads
0.7
0.8
0.9
1.0
1.1
1.2
0
200
400
600
800
1000

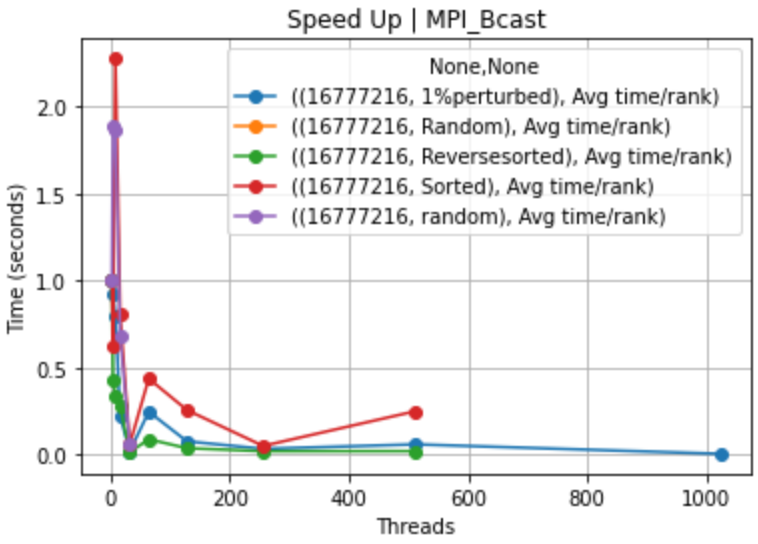

Speed Up | comm_small
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Threads
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
1000

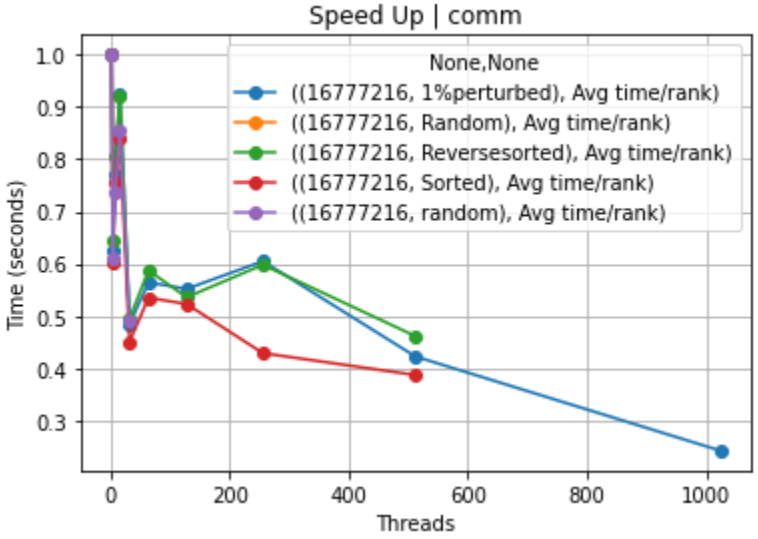

Speed Up | correctness_check
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Threads
0
100
200
300
400
0
200
400
600
800
1000

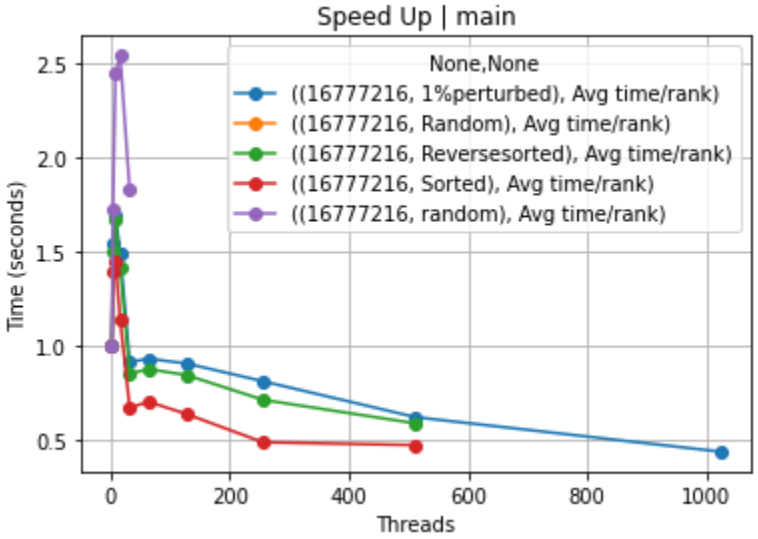

Speed Up | MPI_Barrier
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Threads
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000


Speed Up | MPI_Sendrecv
Time (seconds)
Threads
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)

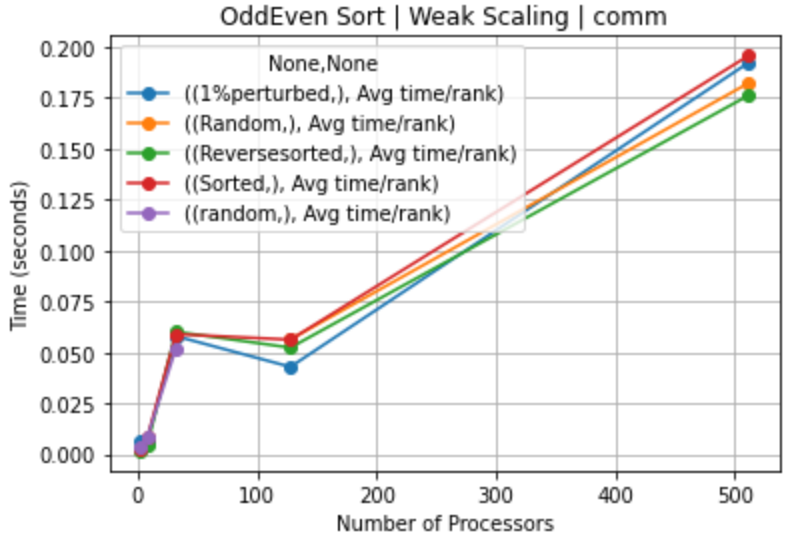

Speed Up | MPI_Gather
Time (seconds)
Threads
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)

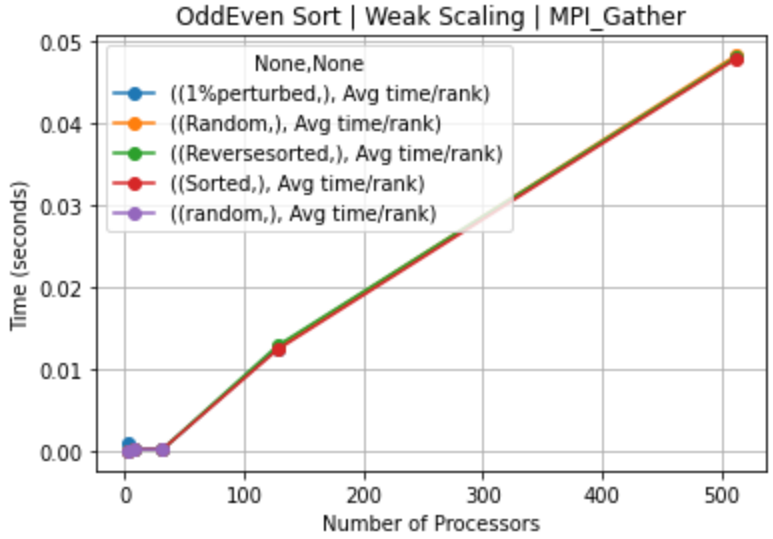

Speed Up | MPI_Bcast
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Threads

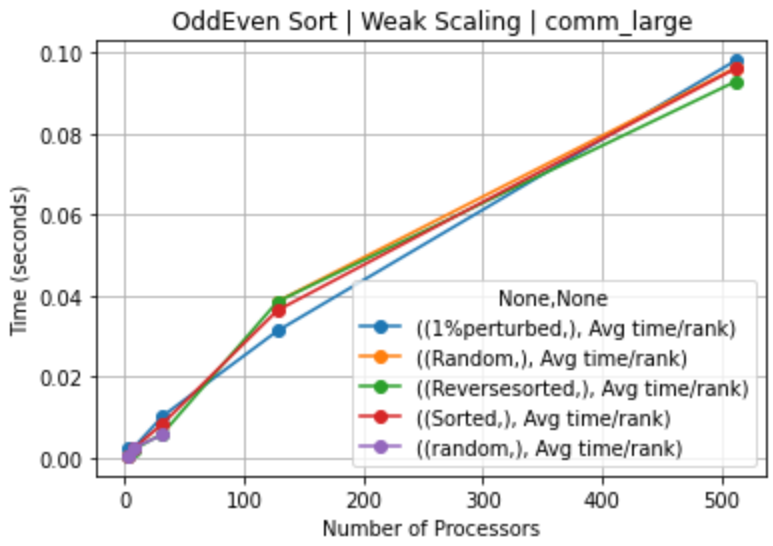

Speed Up | comm
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Threads

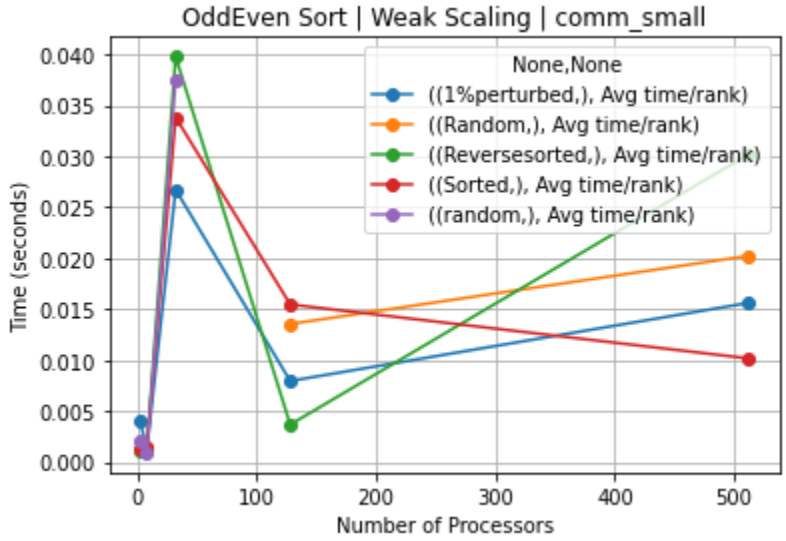

Speed Up | main
None,None
((16777216, 1%perturbed), Avg time/rank)
((16777216, Random), Avg time/rank)
((16777216, Reversesorted), Avg time/rank)
((16777216, Sorted), Avg time/rank)
((16777216, random), Avg time/rank)
Time (seconds)
Threads

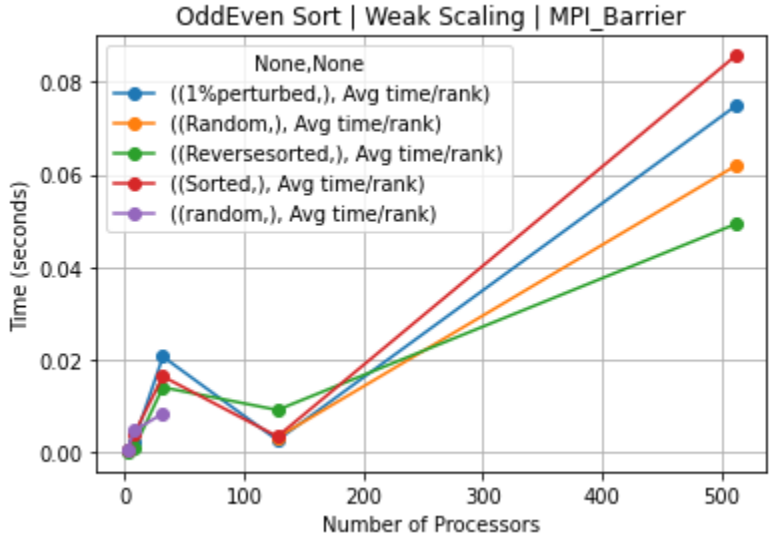
Weak Scaling

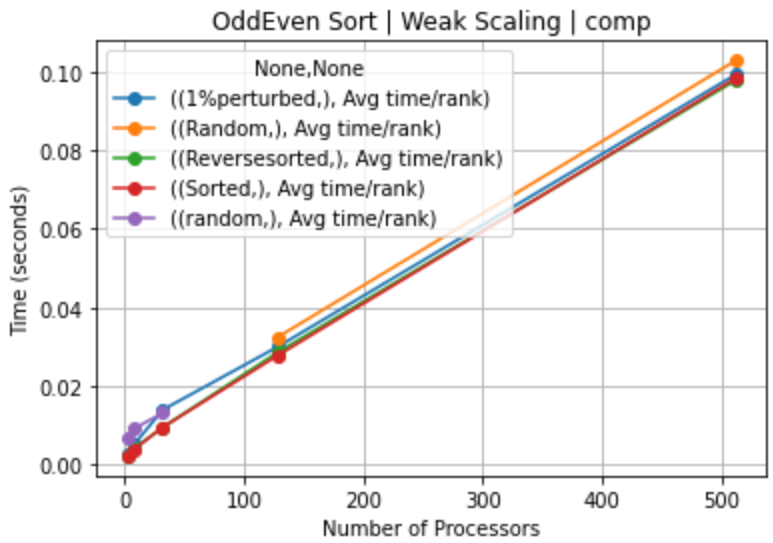

OddEven Sort | Weak Scaling | comm
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)
Time (seconds)
Number of Processors
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
100
200
300
400
500

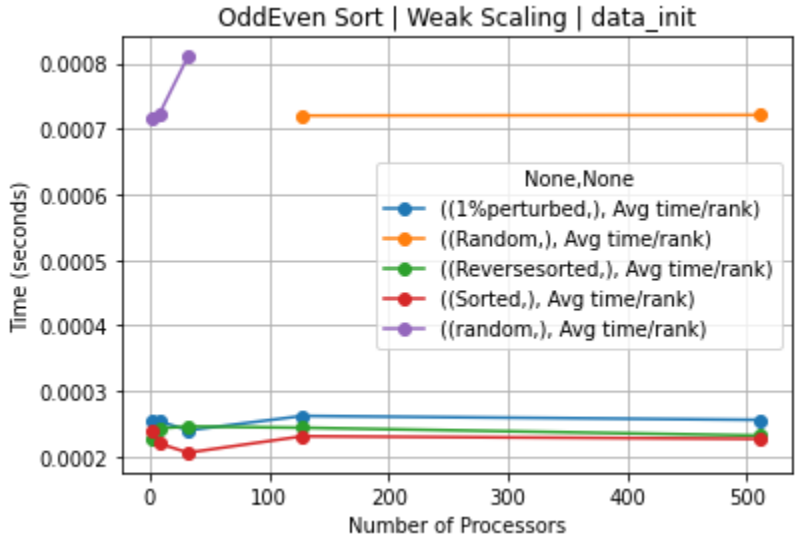

OddEven Sort | Weak Scaling | MPI_Gather
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)
Time (seconds)
Number of Processors
0.05
0.04
0.03
0.02
0.01
0.00
0
100
200
300
400
500

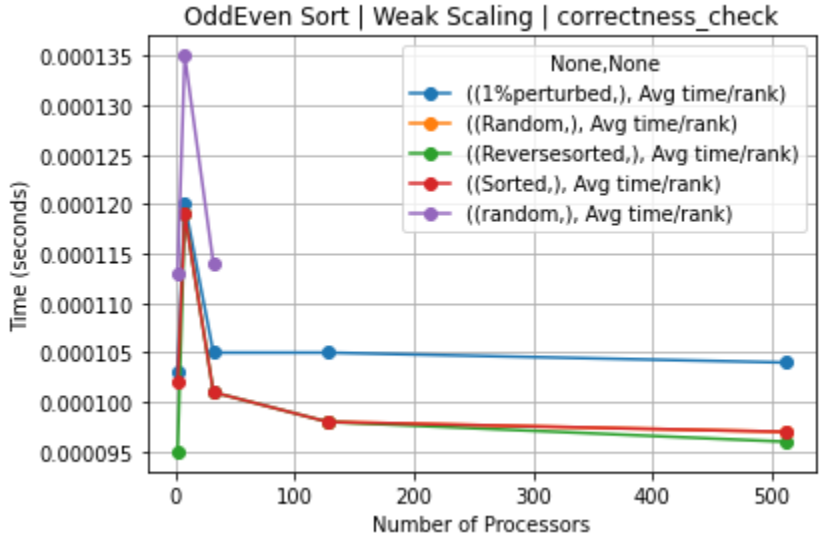

OddEven Sort | Weak Scaling | comm_large
Time (seconds)
Number of Processors
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)

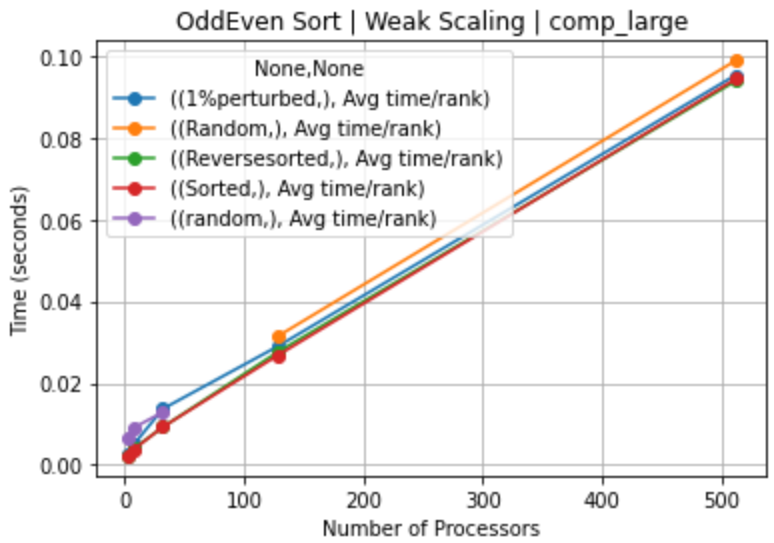

OddEven Sort | Weak Scaling | comm_small
Time (seconds)
Number of Processors
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)

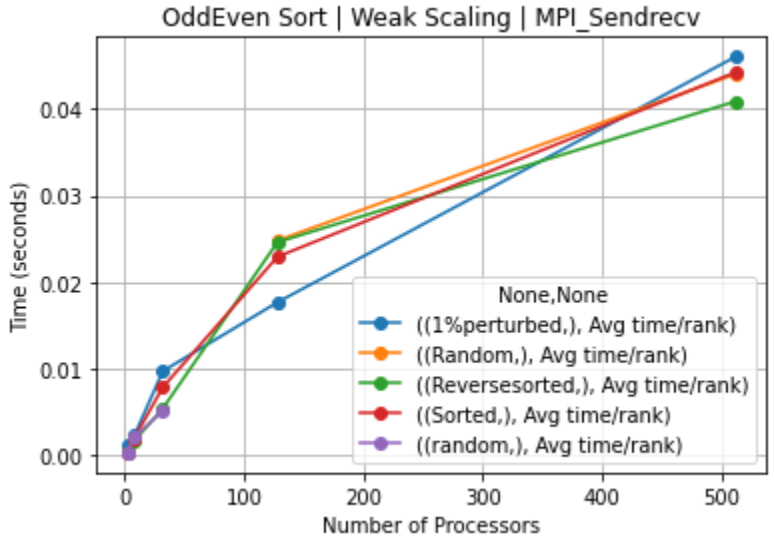

OddEven Sort | Weak Scaling | MPI_Barrier
Time (seconds)
Number of Processors
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)

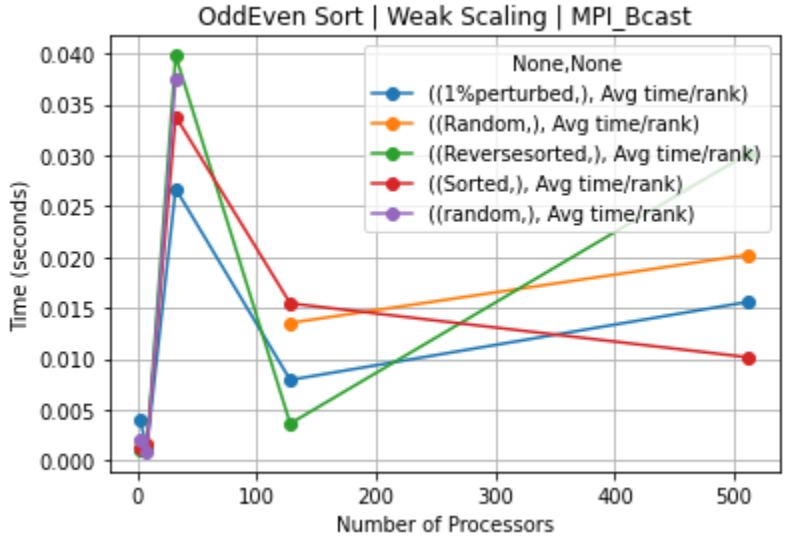

OddEven Sort | Weak Scaling | comp
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)
Time (seconds)
Number of Processors

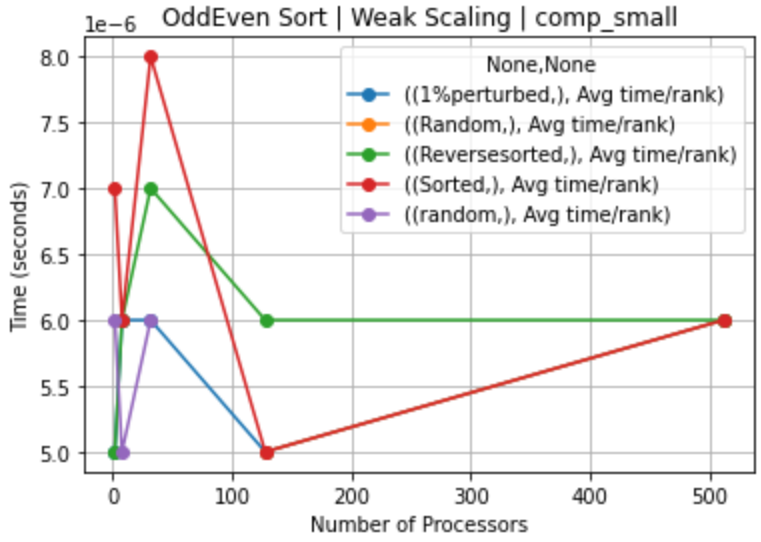

OddEven Sort | Weak Scaling | data_init
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)
Time (seconds)
Number of Processors


OddEven Sort | Weak Scaling | correctness_check
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)
Time (seconds)
Number of Processors

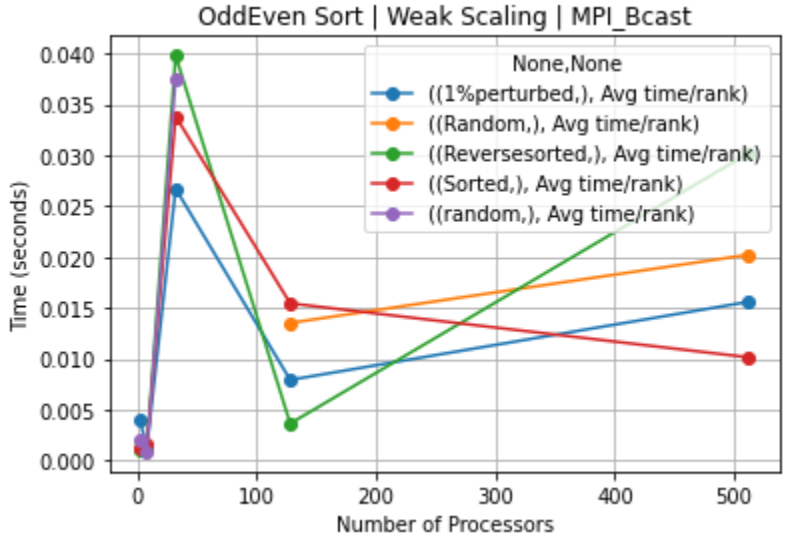

OddEven Sort | Weak Scaling | comp_large
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)
Time (seconds)
Number of Processors
0.00
0.02
0.04
0.06
0.08
0.10
0
100
200
300
400
500


OddEven Sort | Weak Scaling | MPI_Sendrecv
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)
Time (seconds)
Number of Processors
0.00
0.01
0.02
0.03
0.04
0
100
200
300
400
500


OddEven Sort | Weak Scaling | MPI_Bcast
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)
Time (seconds)
Number of Processors
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
0
100
200
300
400
500


OddEven Sort | Weak Scaling | comp_small
1e−6
8.0
7.5
7.0
6.5
6.0
5.5
5.0
Time (seconds)
0
100
200
300
400
500
Number of Processors
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)

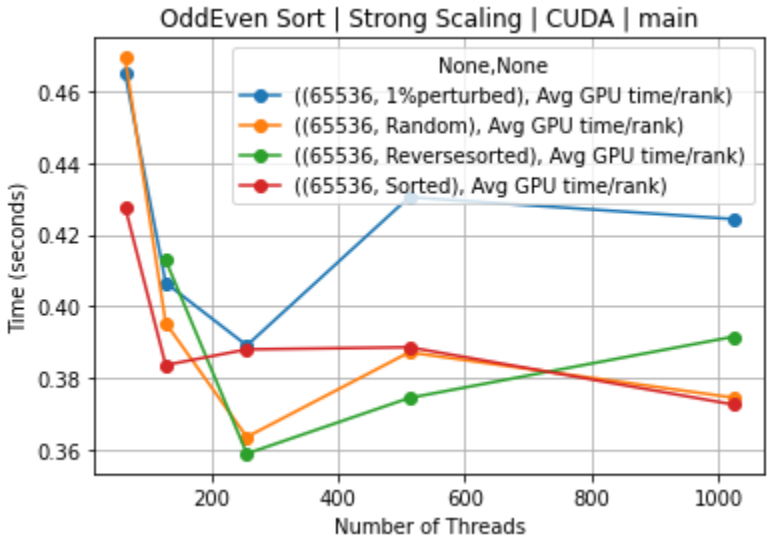

OddEven Sort | Weak Scaling | main
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
Time (seconds)
0
100
200
300
400
500
Number of Processors
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)

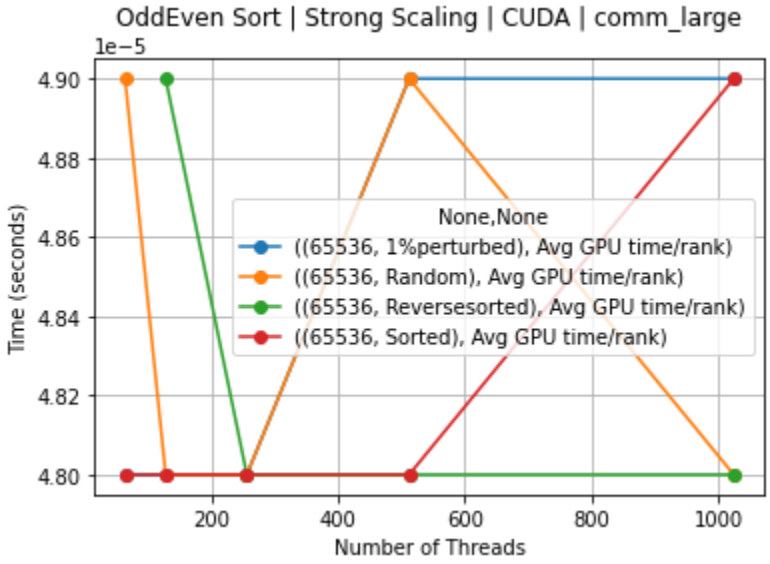

OddEven Sort | Weak Scaling | MPI_Bcast
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Time (seconds)
0
100
200
300
400
500
Number of Processors
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
((random,), Avg time/rank)

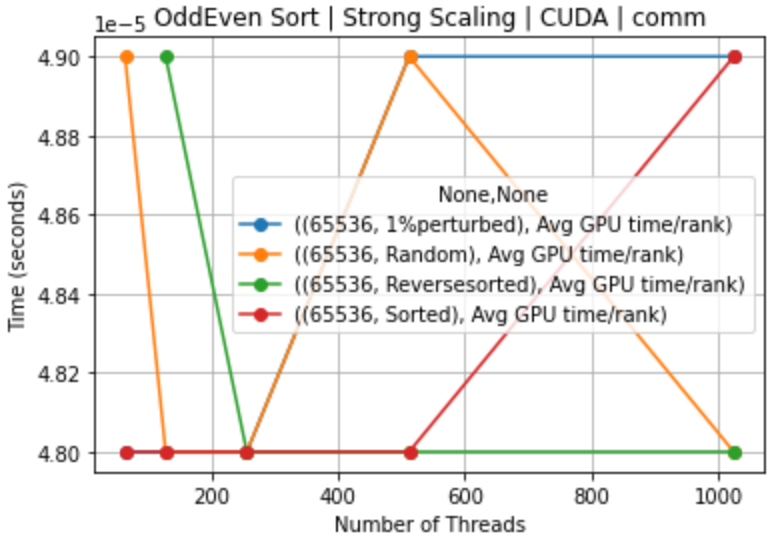
For weak scaling, we see a few trends. For comp_large, comm and main, we see a linear increase with nearly perfect straight lines. This scaling is unideal, and is most likely due to the inefficient nature of the algorithm. In fact, my implementation can have at most half of all data being transferred between processes in any given iteration of the algorithm. This shoots up the communication time, increasing the overhead of the algorithm.

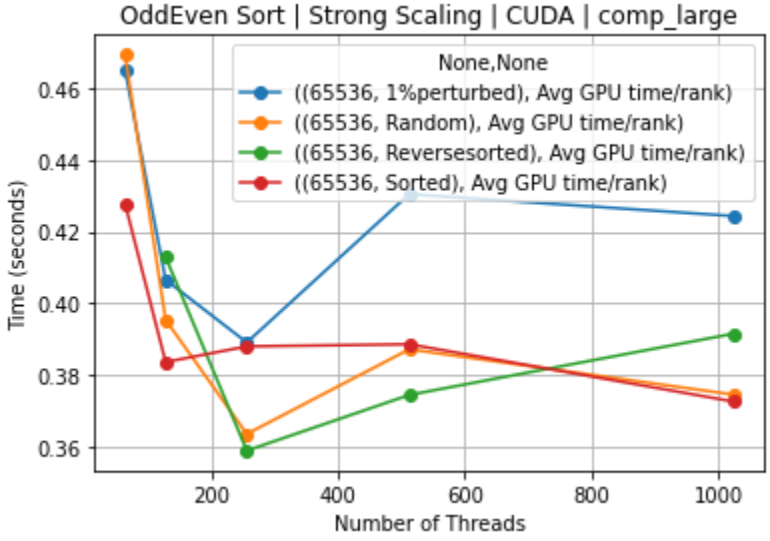
OddEven Sort CUDA:

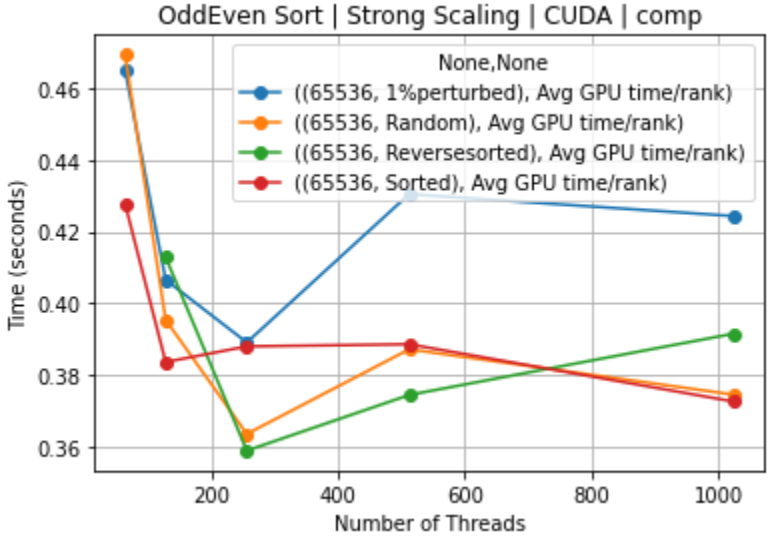
Strong Scaling 65536

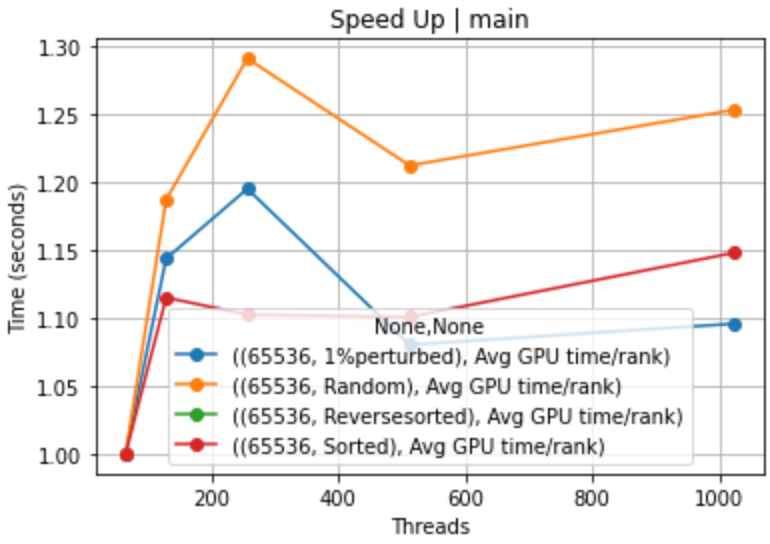


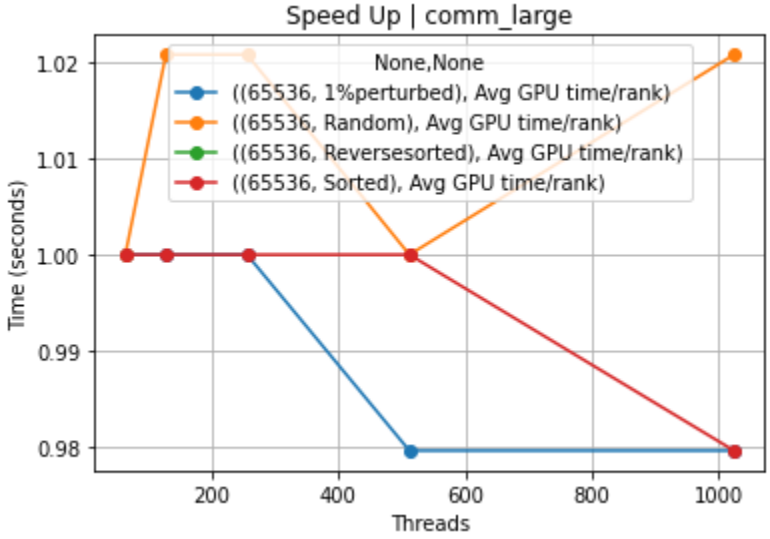


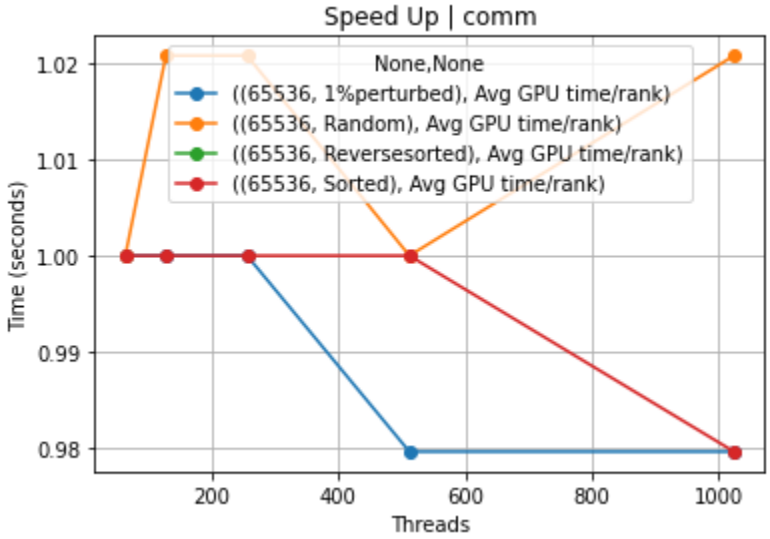

OddEven Sort | Strong Scaling | CUDA | comm
1e−5
4.90
4.88
4.86
4.84
4.82
4.80
Time (seconds)
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

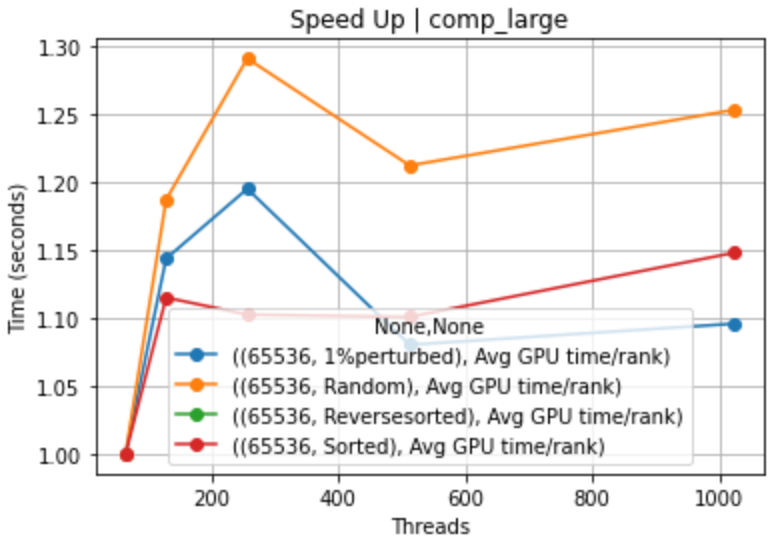

OddEven Sort | Strong Scaling | CUDA | comp_large
0.46
0.44
0.42
0.40
0.38
0.36
Time (seconds)
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

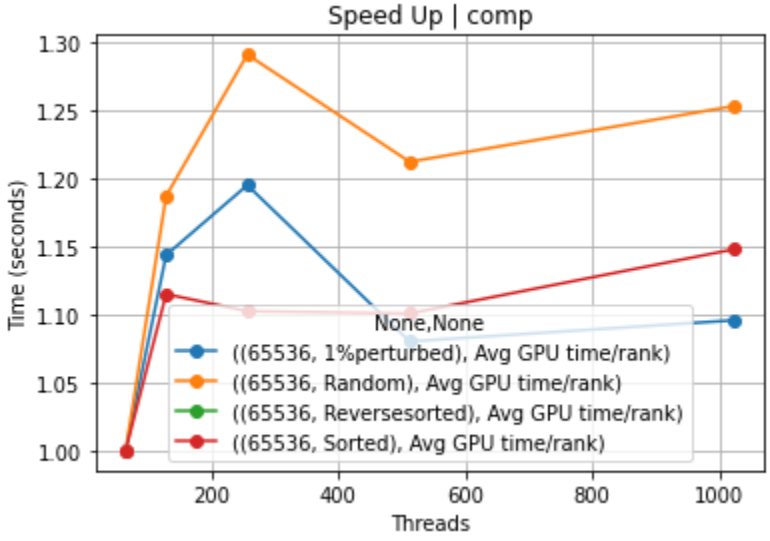

OddEven Sort | Strong Scaling | CUDA | comp
0.46
0.44
0.42
0.40
0.38
0.36
Time (seconds)
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)


Speed Up | main
Time (seconds)
Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

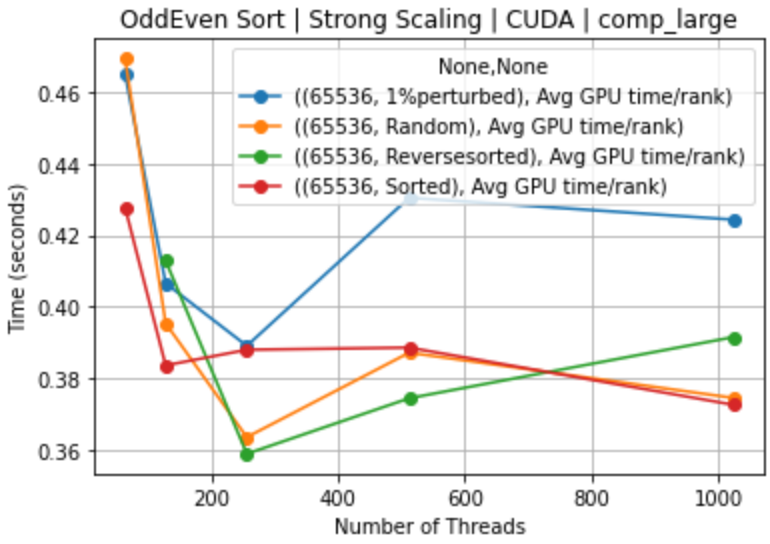

Speed Up | comm_large
Time (seconds)
Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

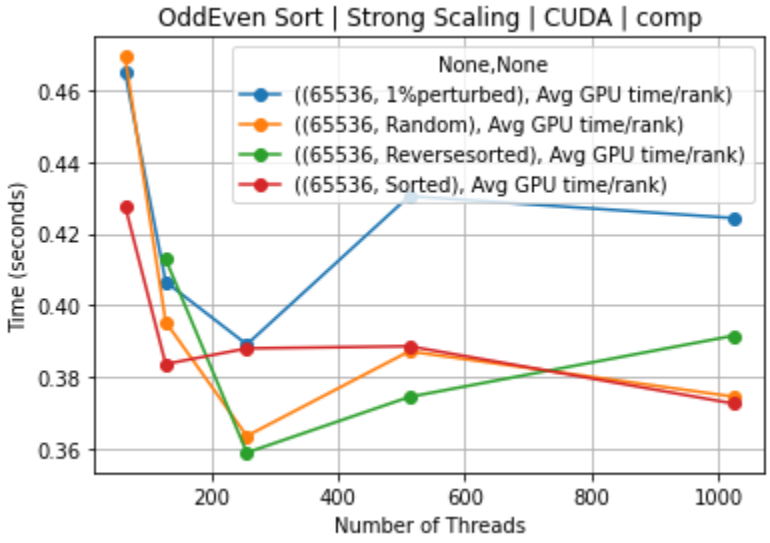

Speed Up | comm
Time (seconds)
Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

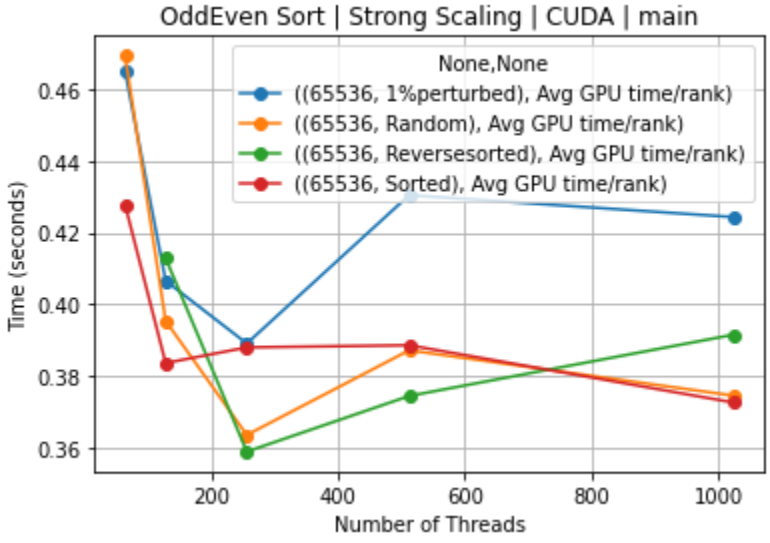


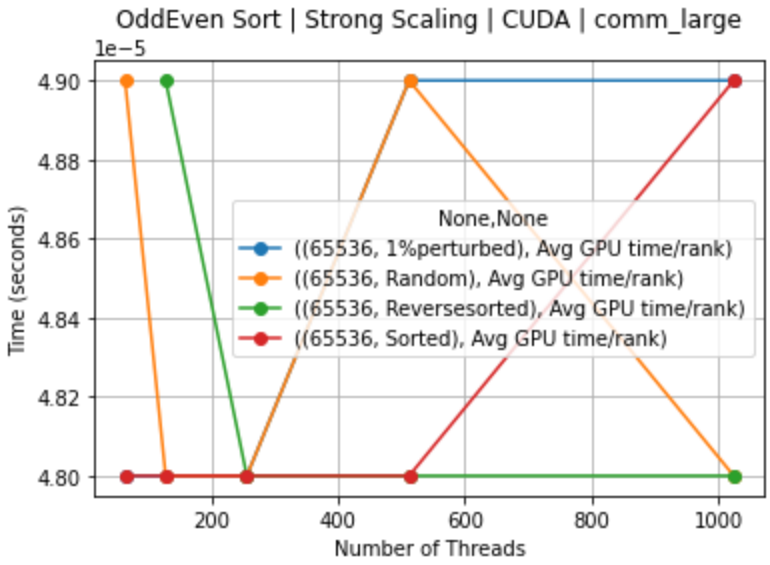


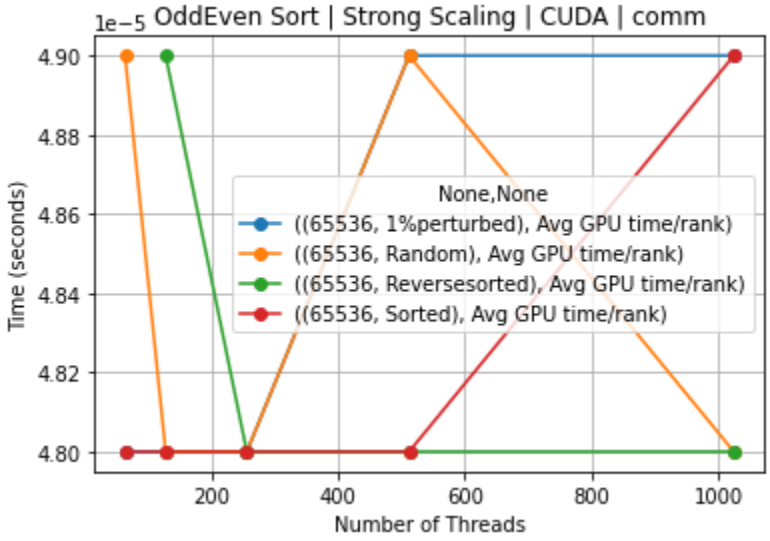
Strong Scaling 262144


OddEven Sort | Strong Scaling | CUDA | comp_large
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
0.46
0.44
0.42
0.40
0.38
0.36
200
400
600
800
1000
Number of Threads

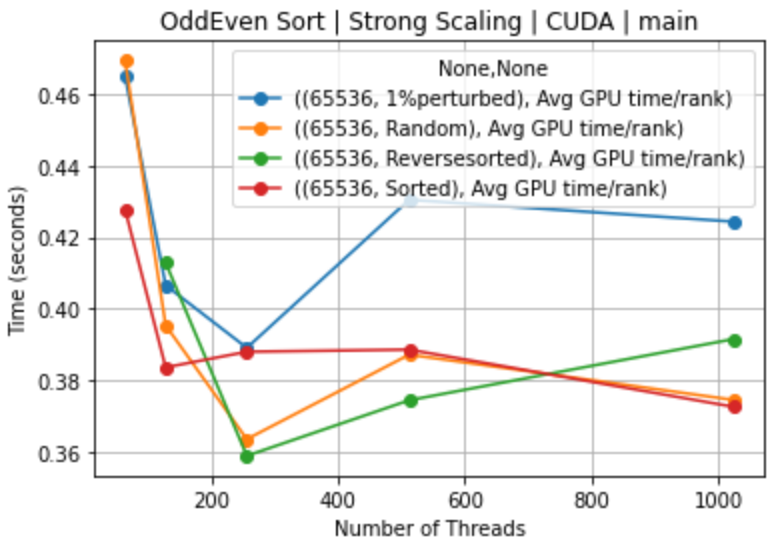

OddEven Sort | Strong Scaling | CUDA | comp
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
0.46
0.44
0.42
0.40
0.38
0.36
200
400
600
800
1000
Number of Threads

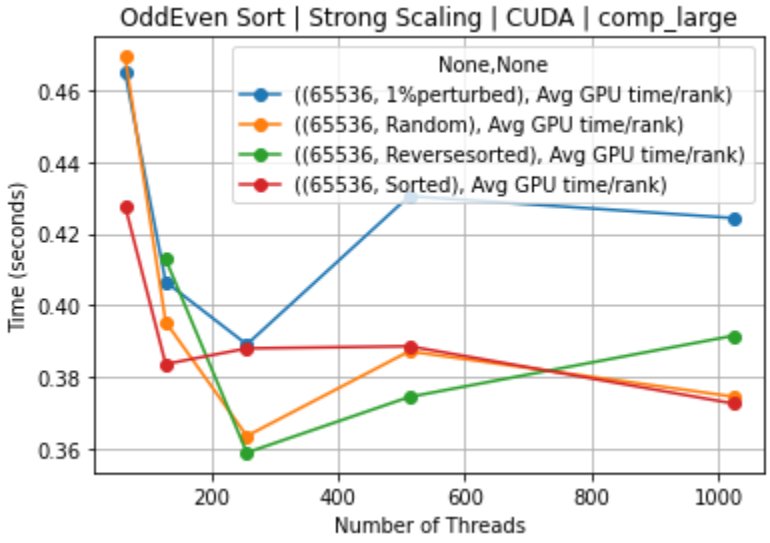

OddEven Sort | Strong Scaling | CUDA | main
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
0.46
0.44
0.42
0.40
0.38
0.36
200
400
600
800
1000
Number of Threads

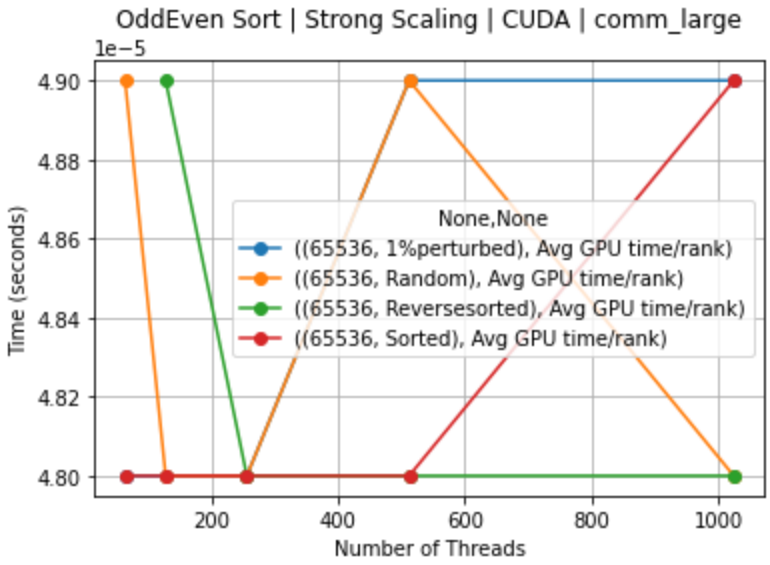


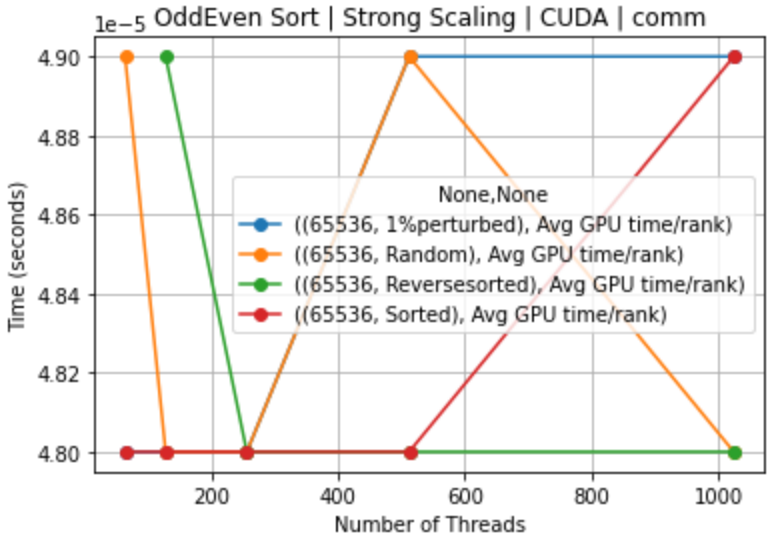


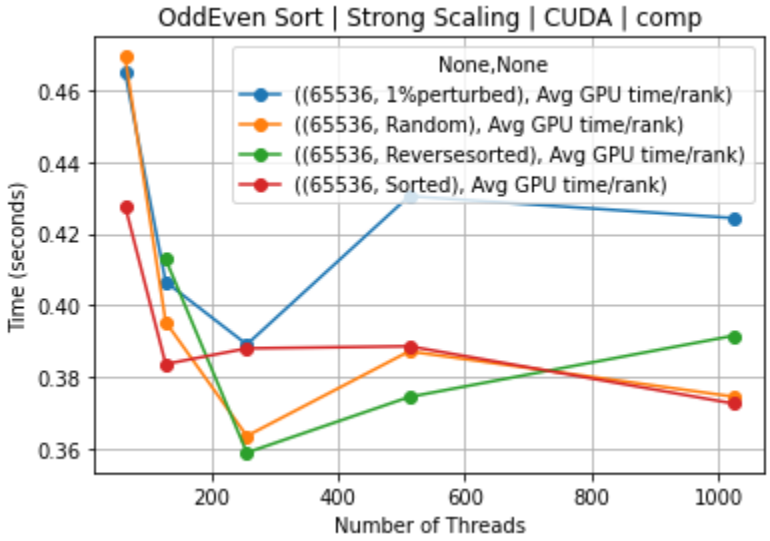
Strong Scaling 1048576


OddEven Sort | Strong Scaling | CUDA | main
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
0.46
0.44
0.42
0.40
0.38
0.36
200
400
600
800
1000
Number of Threads

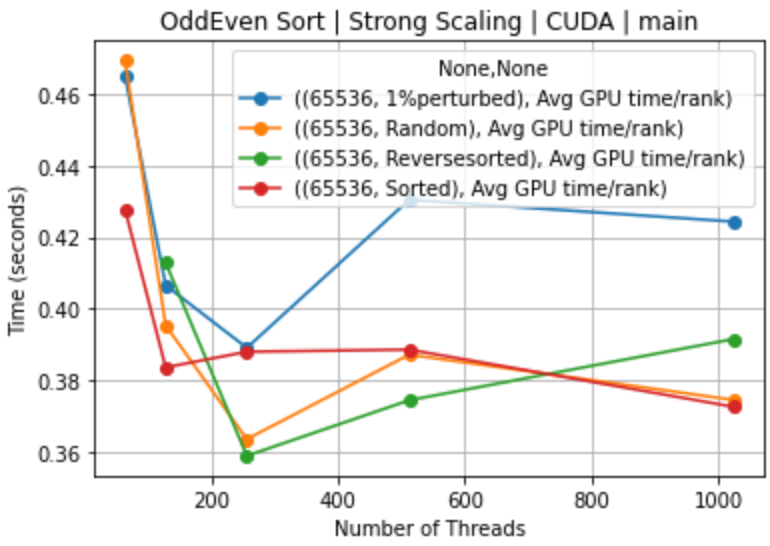

OddEven Sort | Strong Scaling | CUDA | comp_large
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
0.46
0.44
0.42
0.40
0.38
0.36
200
400
600
800
1000
Number of Threads

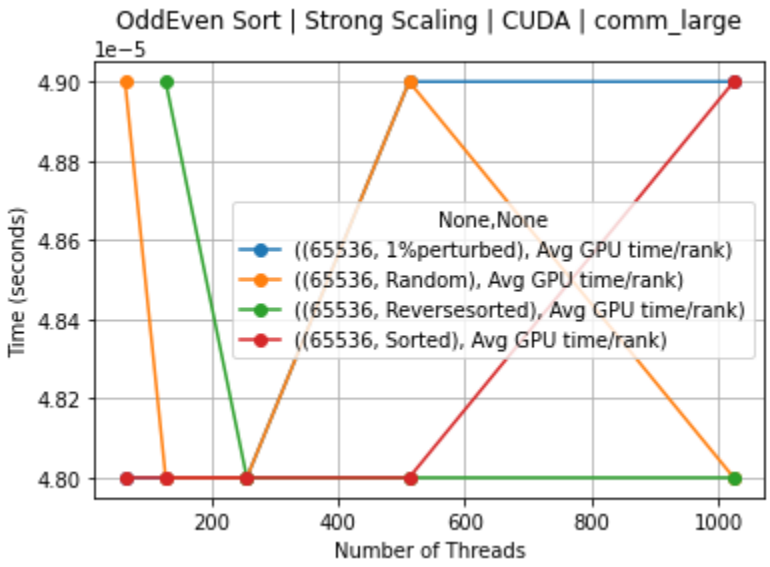

OddEven Sort | Strong Scaling | CUDA | comm_large
1e−5
4.90
4.88
4.86
4.84
4.82
4.80
Time (seconds)
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

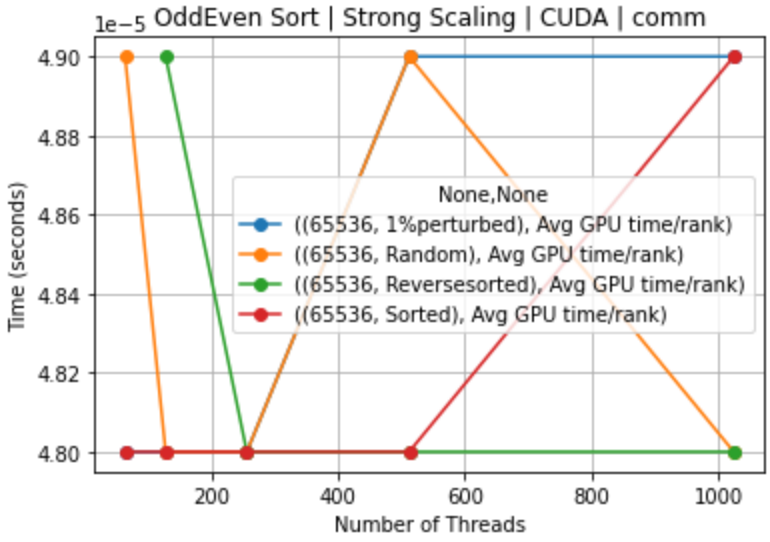


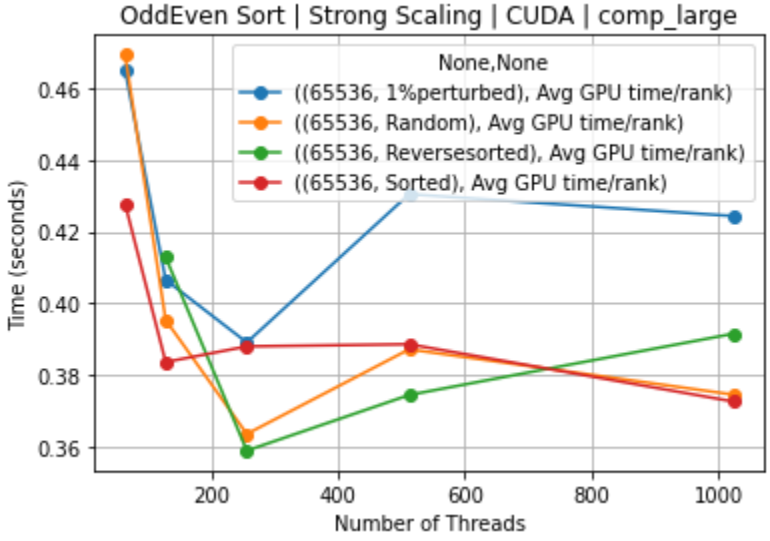


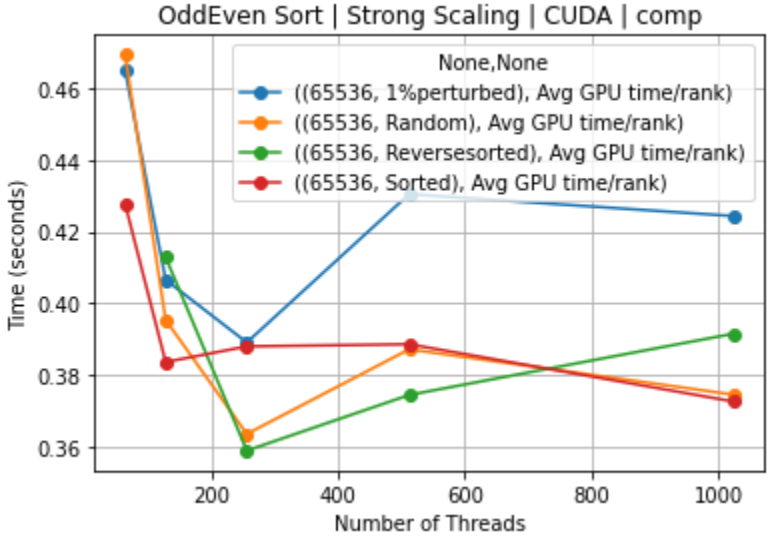
Strong Scaling 4194304


OddEven Sort | Strong Scaling | CUDA | main
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
0.46
0.44
0.42
0.40
0.38
0.36
200
400
600
800
1000
Number of Threads

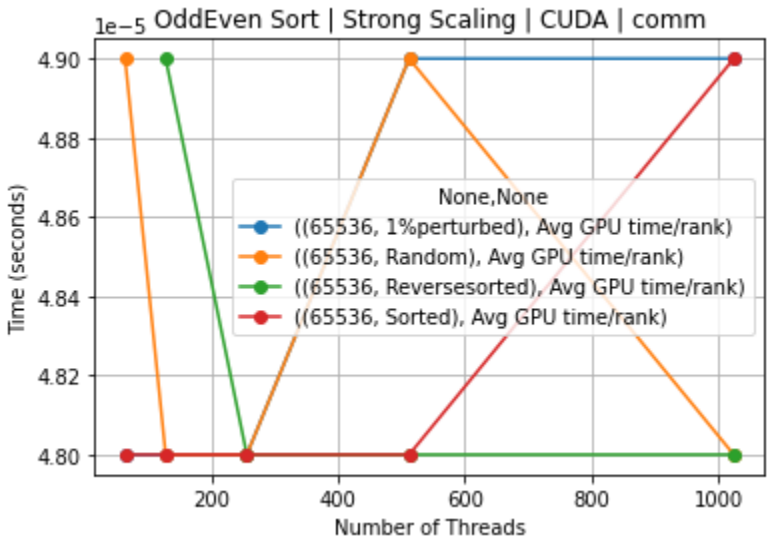

OddEven Sort | Strong Scaling | CUDA | comm_large
1e−5
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
4.90
4.88
4.86
4.84
4.82
4.80
200
400
600
800
1000
Number of Threads

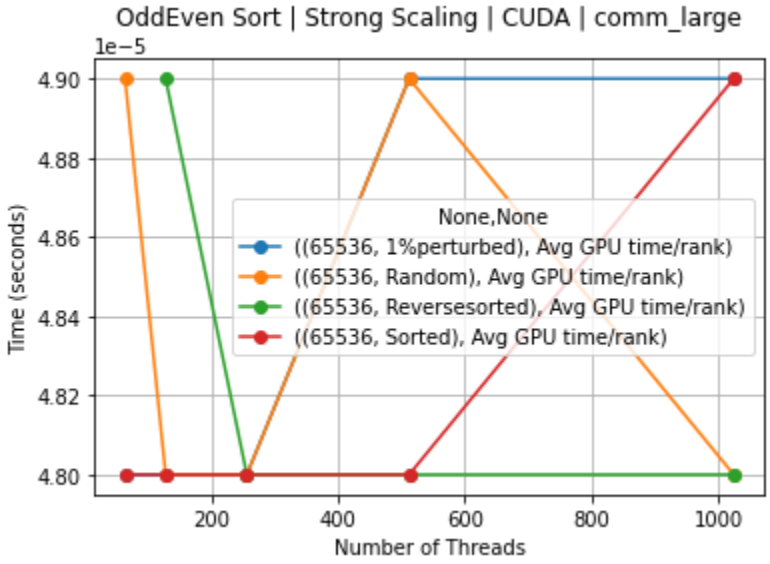

OddEven Sort | Strong Scaling | CUDA | comm
1e−5
4.90
4.88
4.86
4.84
4.82
4.80
Time (seconds)
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

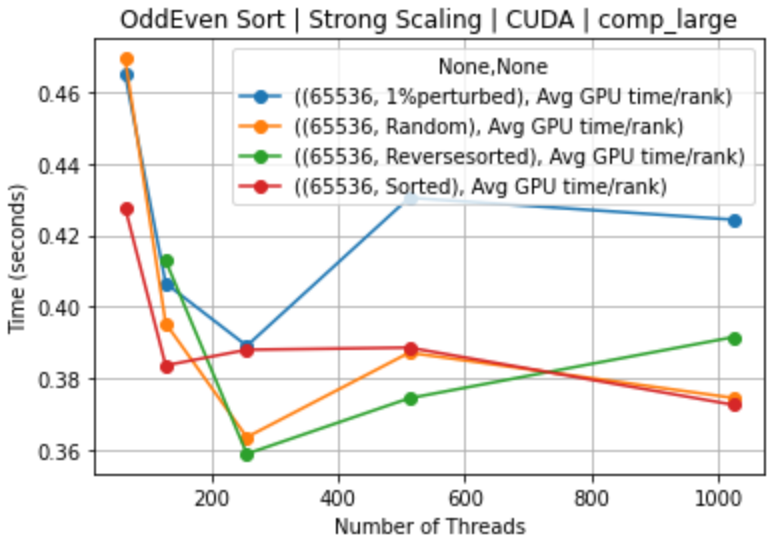

OddEven Sort | Strong Scaling | CUDA | comp_large
0.46
0.44
0.42
0.40
0.38
0.36
Time (seconds)
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

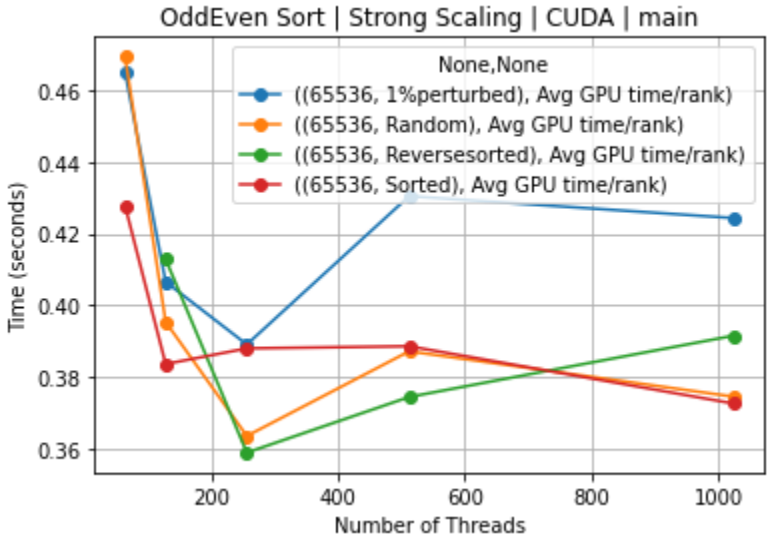

OddEven Sort | Strong Scaling | CUDA | comp
0.46
0.44
0.42
0.40
0.38
0.36
Time (seconds)
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

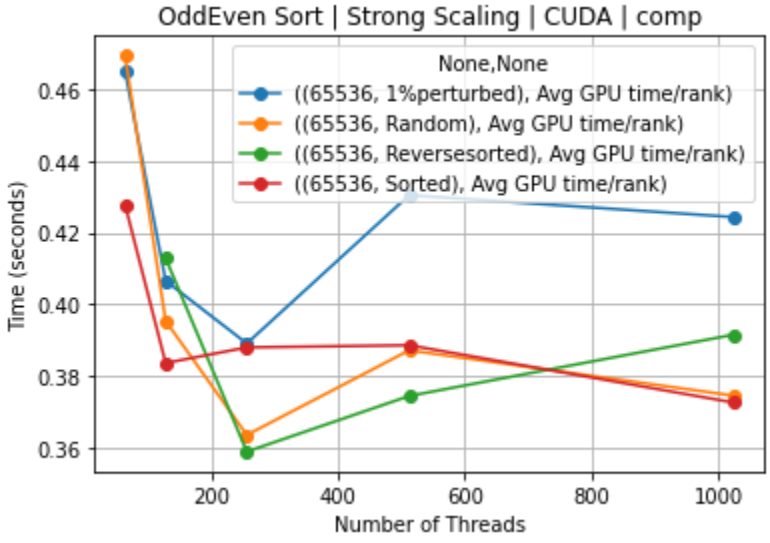
Strong Scaling 1048576



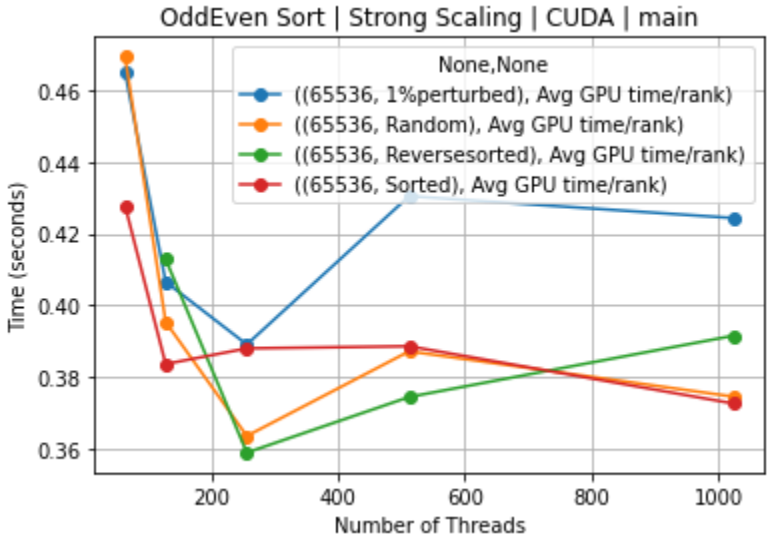


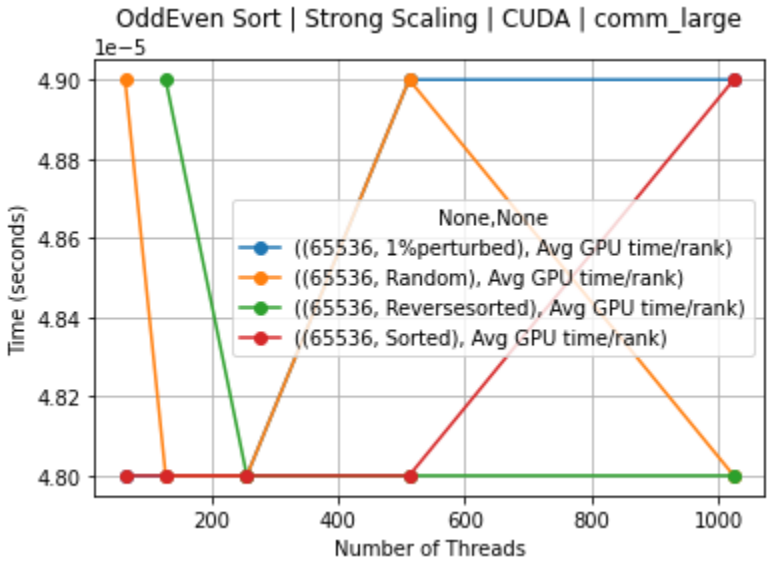

OddEven Sort | Strong Scaling | CUDA | comp_large
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
0.46
0.44
0.42
0.40
0.38
0.36
200
400
600
800
1000
Number of Threads

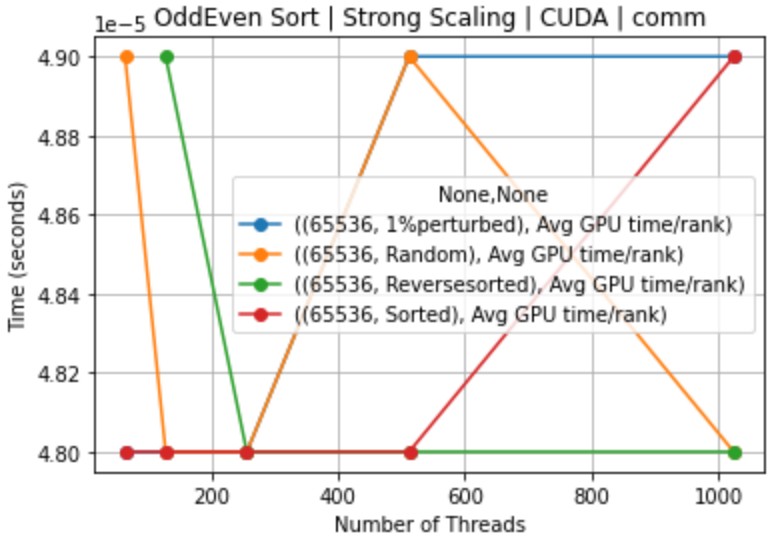

OddEven Sort | Strong Scaling | CUDA | main
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
0.46
0.44
0.42
0.40
0.38
0.36
200
400
600
800
1000
Number of Threads

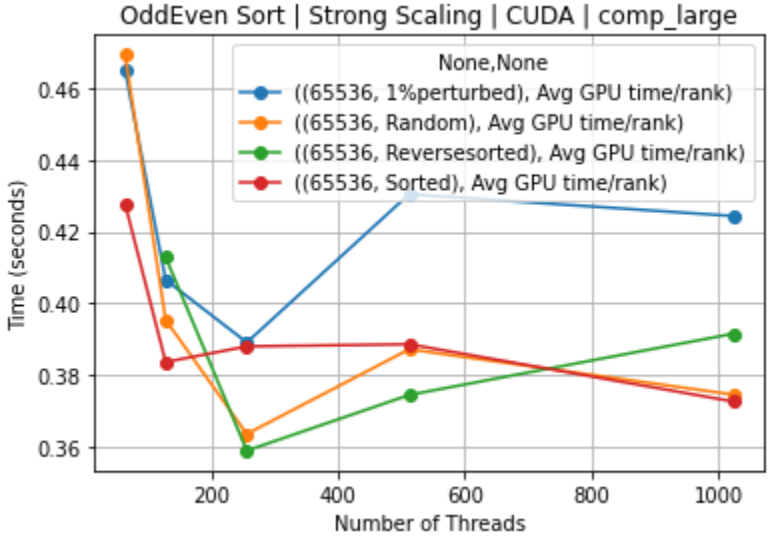

OddEven Sort | Strong Scaling | CUDA | comp
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)
Time (seconds)
0.46
0.44
0.42
0.40
0.38
0.36
200
400
600
800
1000
Number of Threads

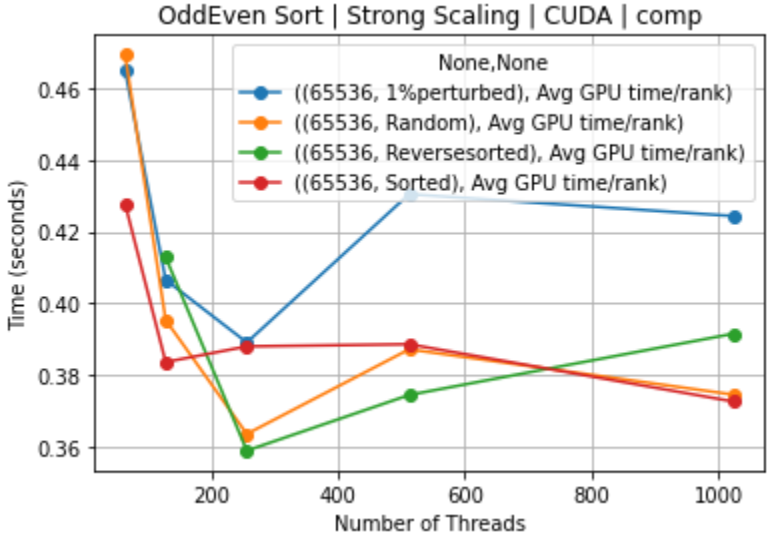
Strong Scaling 16777216



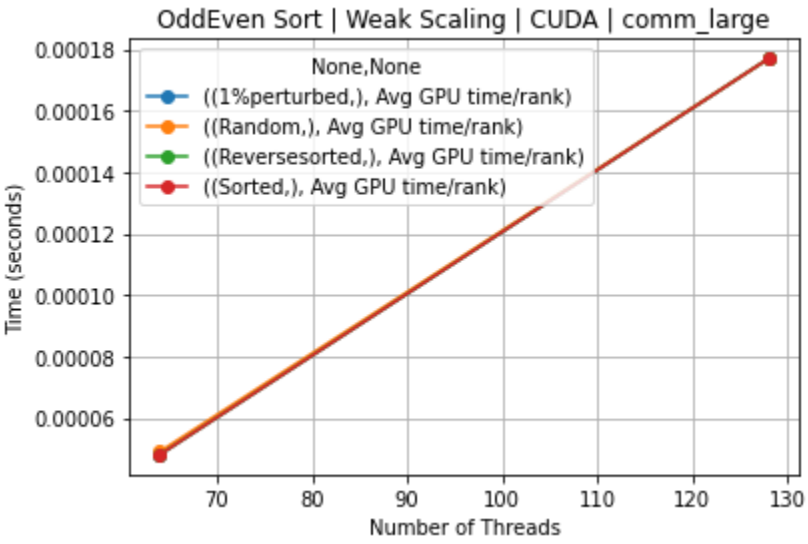


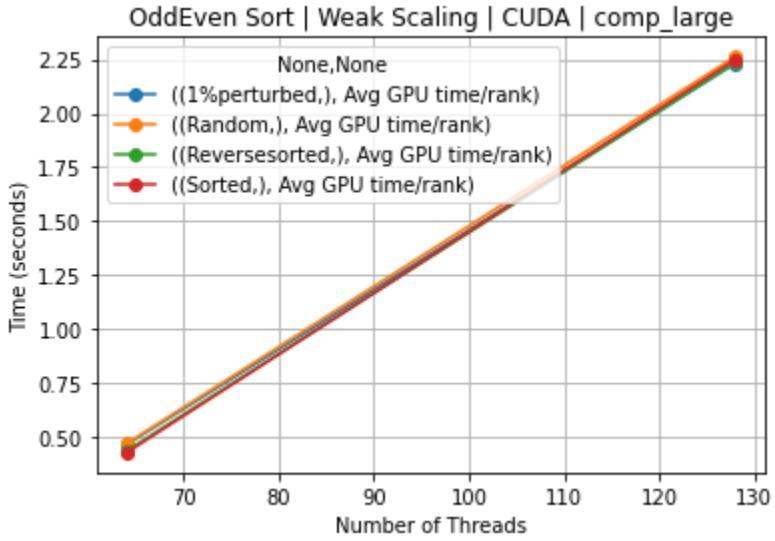

OddEven Sort | Strong Scaling | CUDA | comm
1e−5
4.90
4.88
4.86
4.84
4.82
4.80
Time (seconds)
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

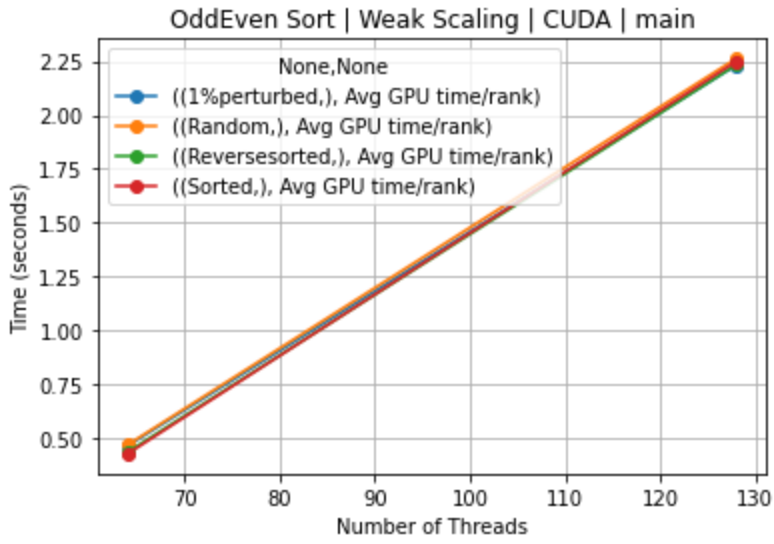

OddEven Sort | Strong Scaling | CUDA | comp_large
0.46
0.44
0.42
0.40
0.38
0.36
Time (seconds)
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

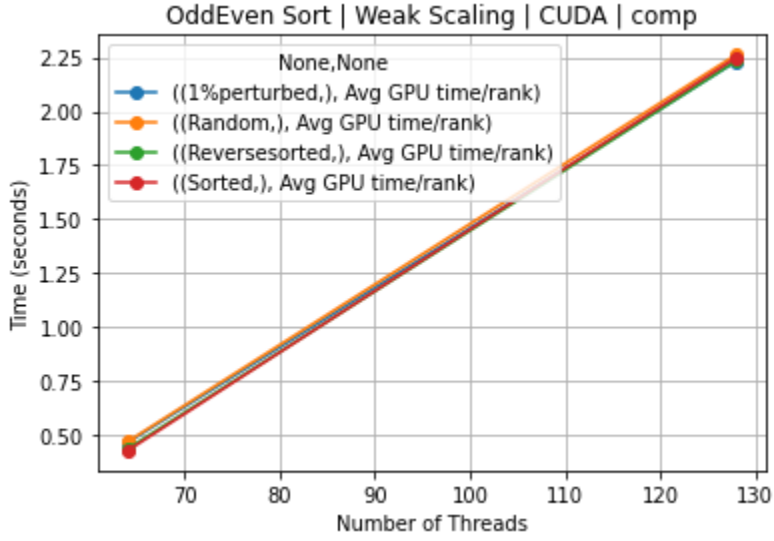

OddEven Sort | Strong Scaling | CUDA | comp
0.46
0.44
0.42
0.40
0.38
0.36
Time (seconds)
200
400
600
800
1000
Number of Threads
None,None
((65536, 1%perturbed), Avg GPU time/rank)
((65536, Random), Avg GPU time/rank)
((65536, Reversesorted), Avg GPU time/rank)
((65536, Sorted), Avg GPU time/rank)

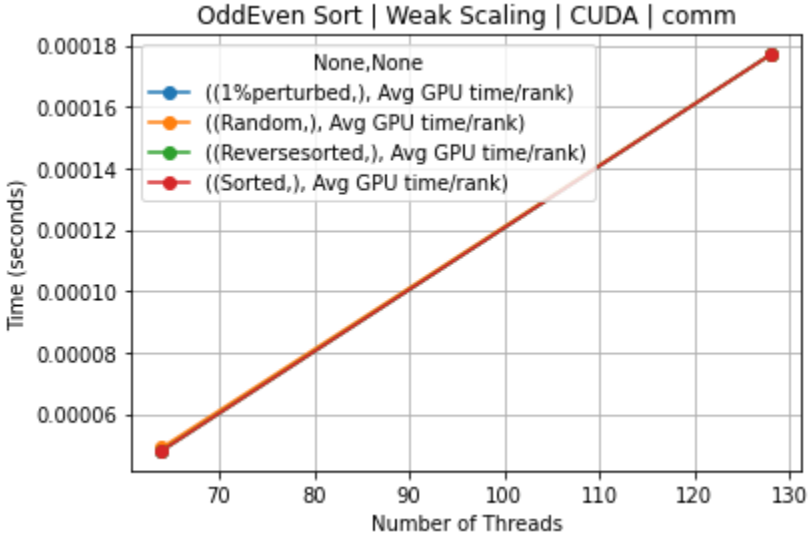
Weak Scaling






OddEven Sort | Weak Scaling | CUDA | main
None,None
((1%perturbed,), Avg GPU time/rank)
((Random,), Avg GPU time/rank)
((Reversesorted,), Avg GPU time/rank)
((Sorted,), Avg GPU time/rank)
Time (seconds)
Number of Threads

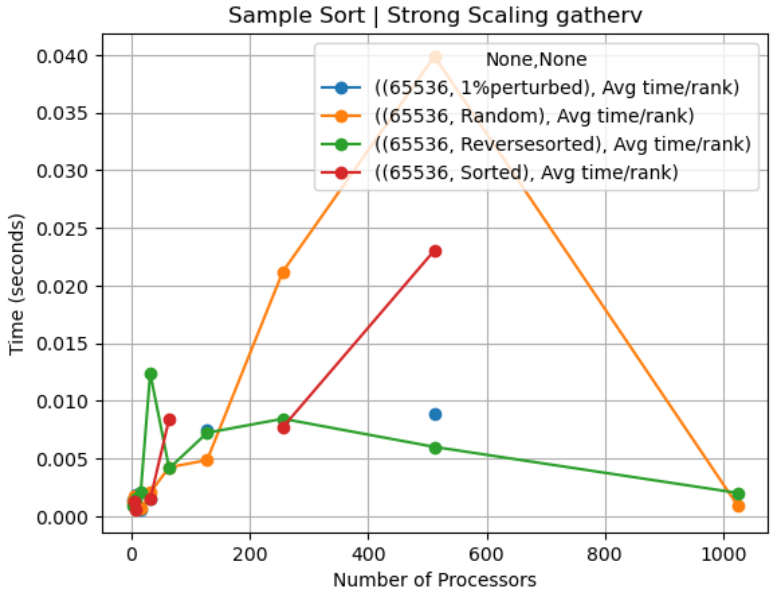

OddEven Sort | Weak Scaling | CUDA | comp
None,None
((1%perturbed,), Avg GPU time/rank)
((Random,), Avg GPU time/rank)
((Reversesorted,), Avg GPU time/rank)
((Sorted,), Avg GPU time/rank)
Time (seconds)
Number of Threads

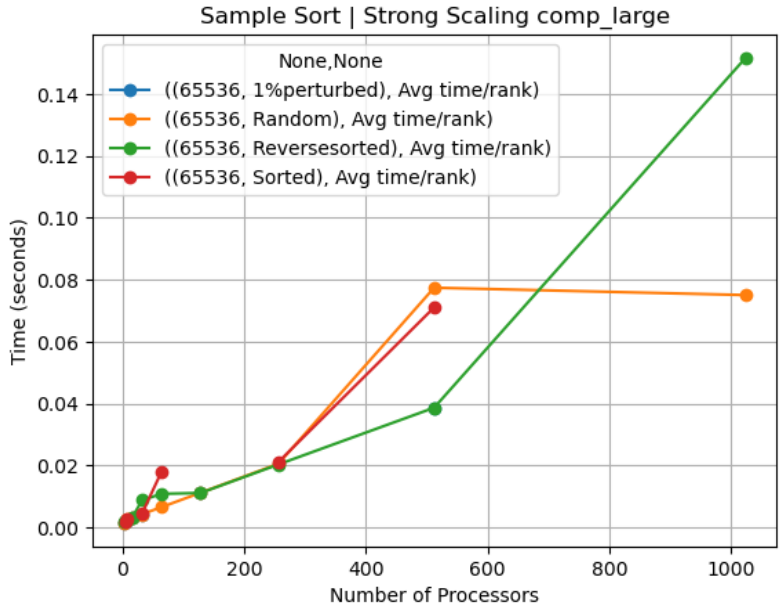

OddEven Sort | Weak Scaling | CUDA | comm
None,None
((1%perturbed,), Avg GPU time/rank)
((Random,), Avg GPU time/rank)
((Reversesorted,), Avg GPU time/rank)
((Sorted,), Avg GPU time/rank)
Time (seconds)
Number of Threads

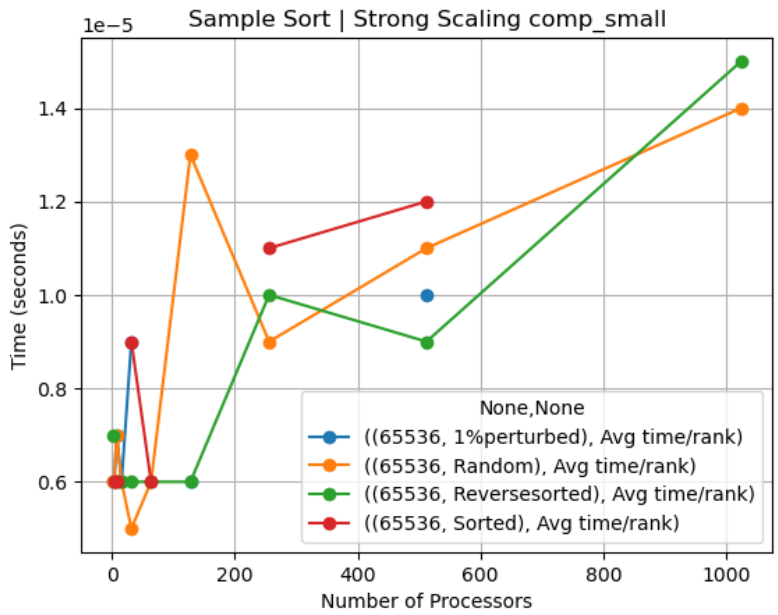
We see a linear decrease time for number of threads, until a point where things increase up.

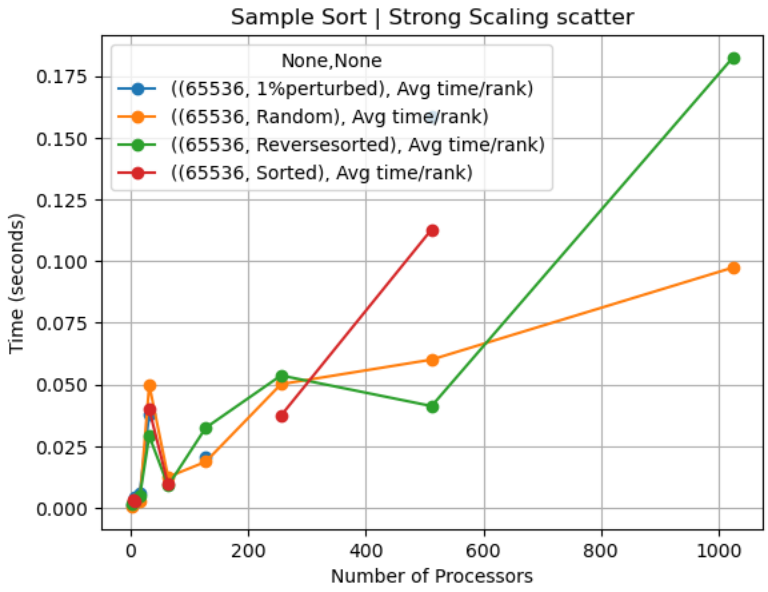
Sample Sort MPI:

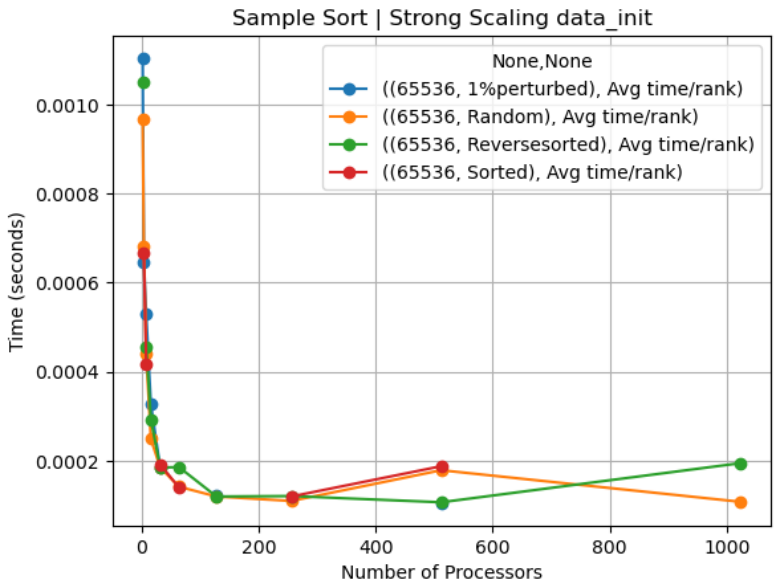
Strong Scaling:

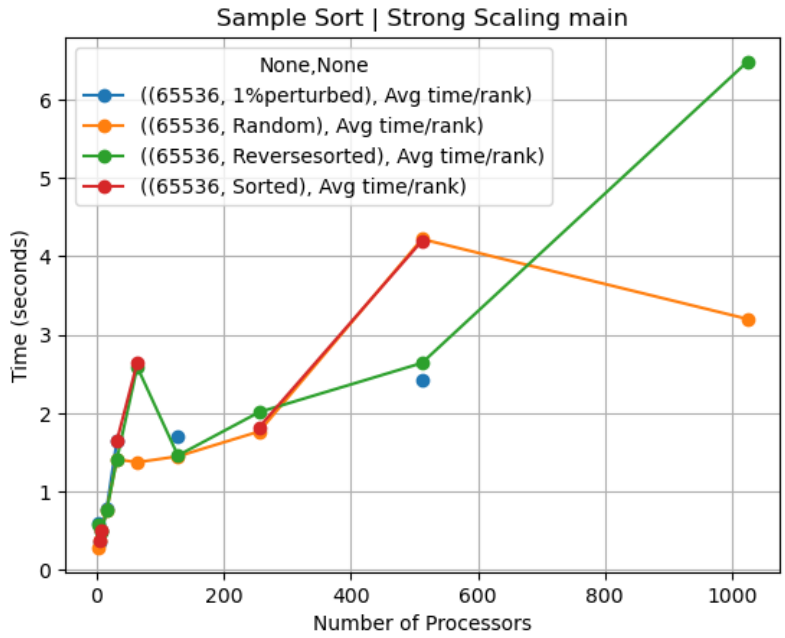


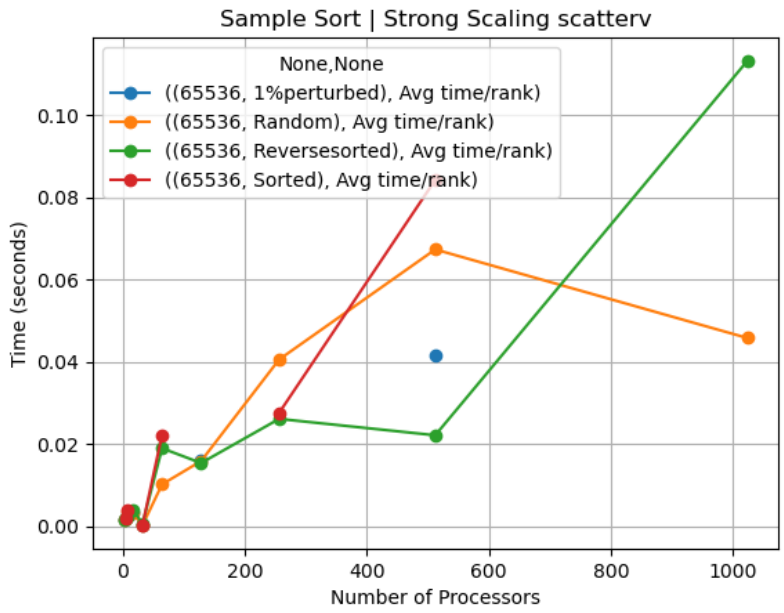



Sample Sort | Strong Scaling comp_small
1e−5
Time (seconds)
Number of Processors
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)


Sample Sort | Strong Scaling scatter
Time (seconds)
Number of Processors
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

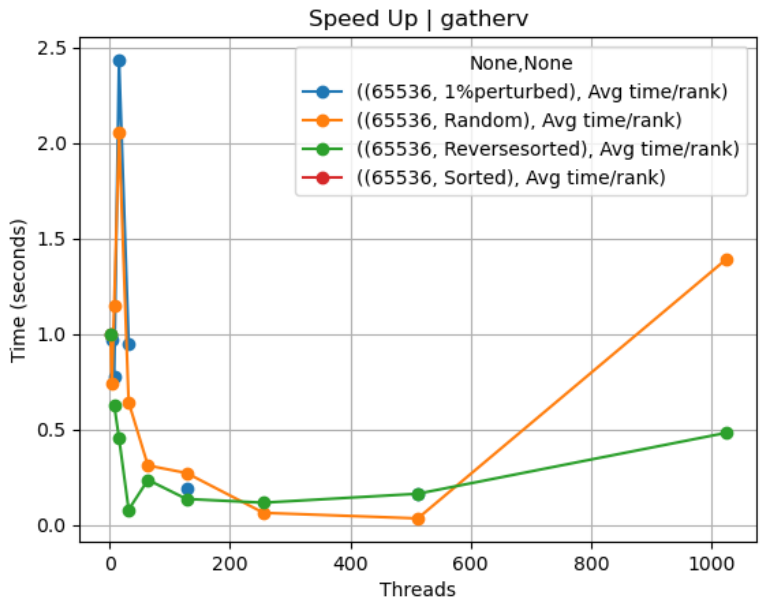

Sample Sort | Strong Scaling data_init
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
0.0010
0.0008
0.0006
0.0004
0.0002
0
200
400
600
800
1000
Number of Processors

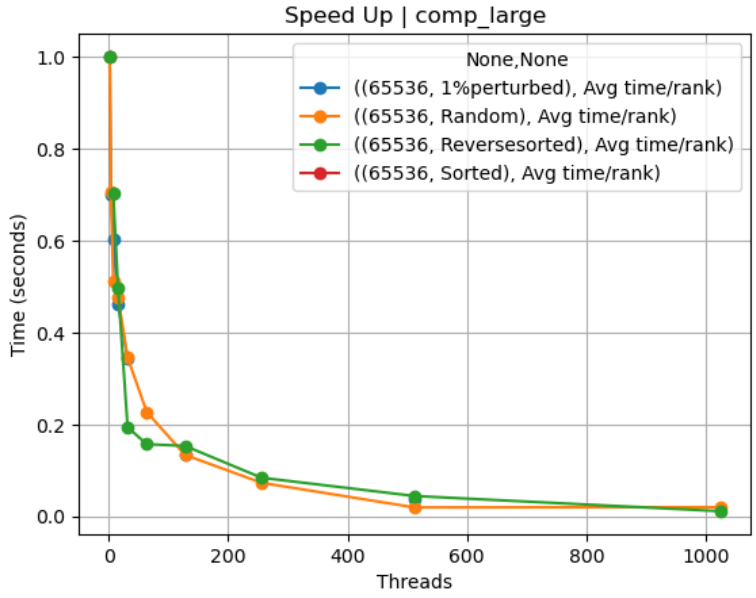

Sample Sort | Strong Scaling main
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
6
5
4
3
2
1
0
0
200
400
600
800
1000
Number of Processors

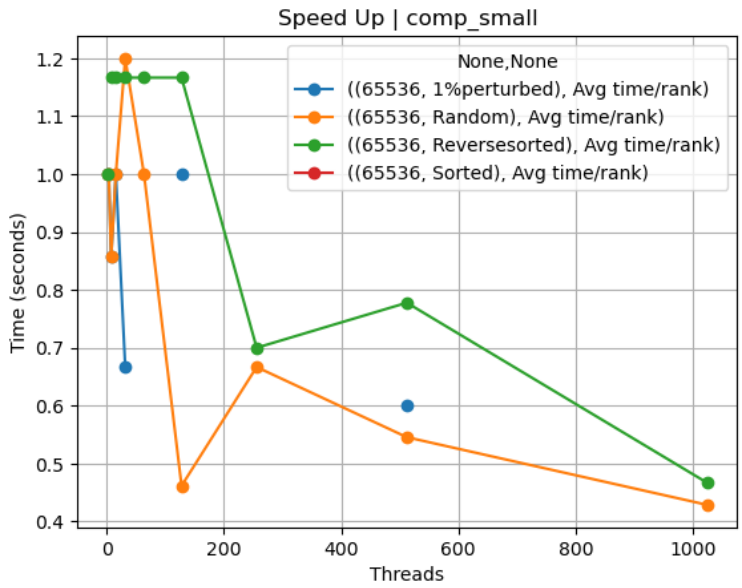


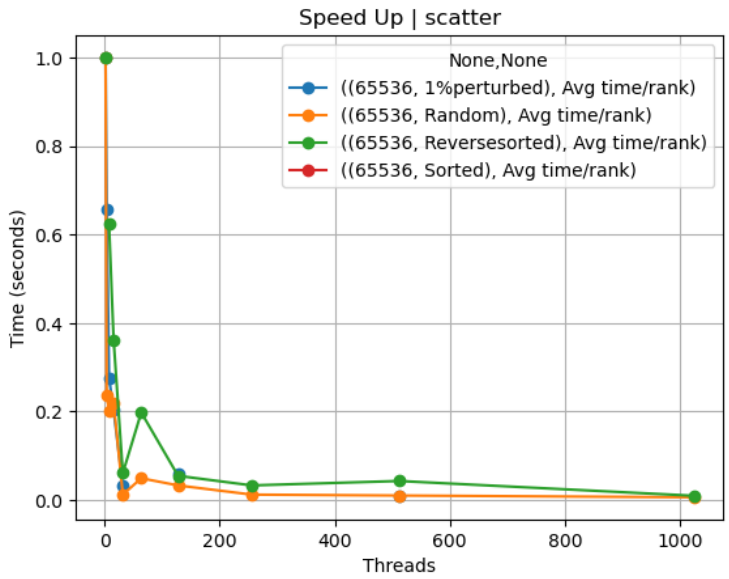
The time complexity for these processors are very linear. There may be an inefficiency in the algorithm such that the extra processors do not improve the final time. I assume that when I increase the amount of processors that my program is using the inefficiencies in the communication between processors are exacerbated. This result in the performance to linearly decrease as I increase the number of processors that my program is using.

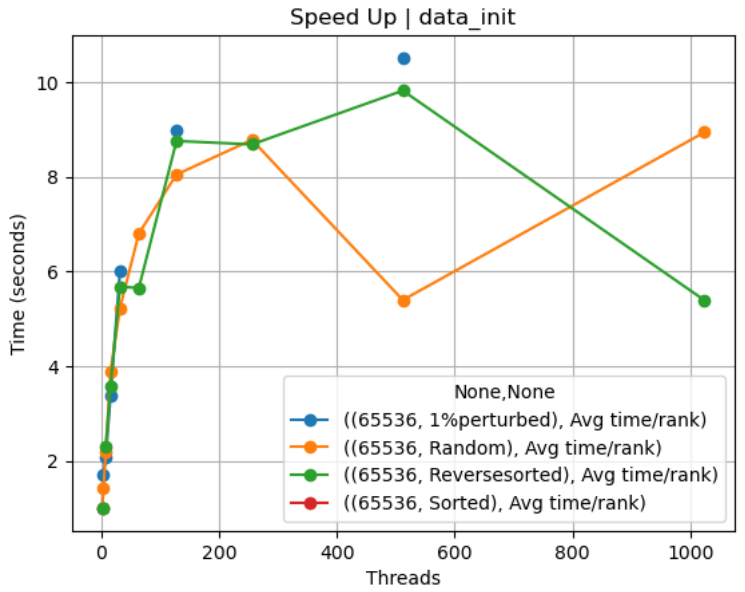
Speedup:

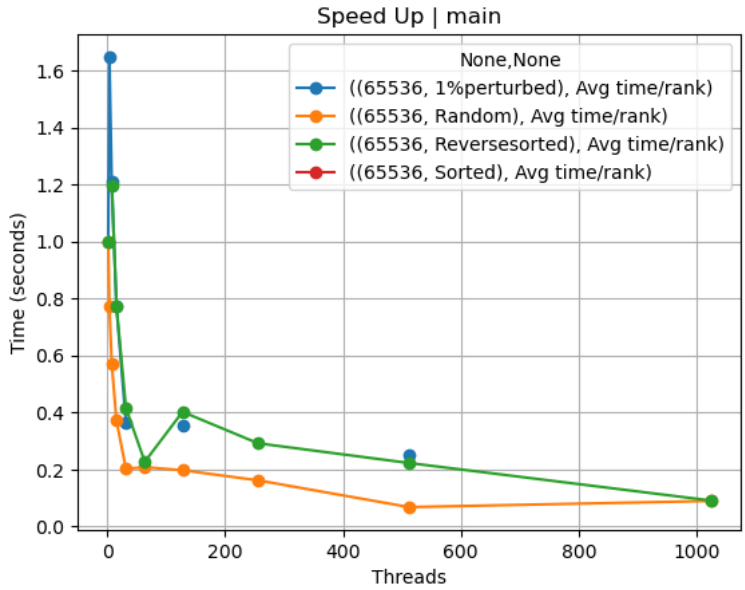

Speed Up | gatherv
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
0.0
0.5
1.0
1.5
2.0
2.5
0
200
400
600
800
1000

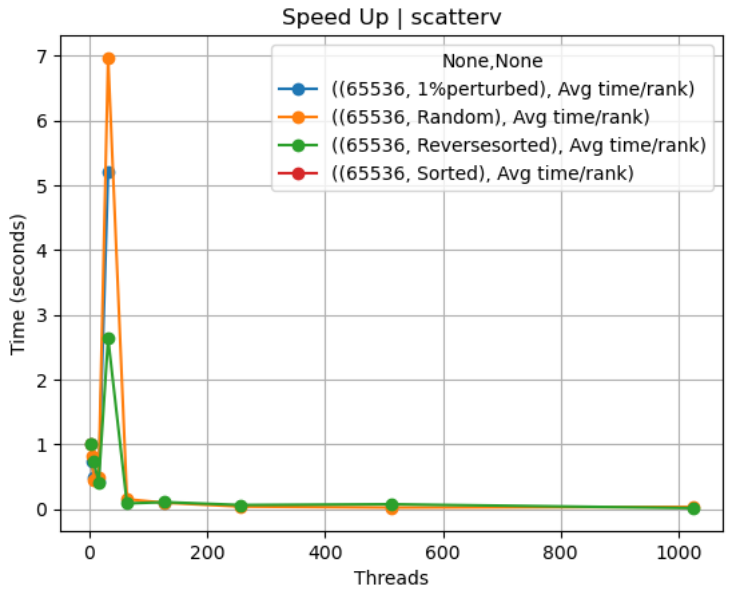

Speed Up | comp_large
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000


Speed Up | comp_small
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
200
400
600
800
1000


Speed Up | scatter
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)
Time (seconds)
Threads
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

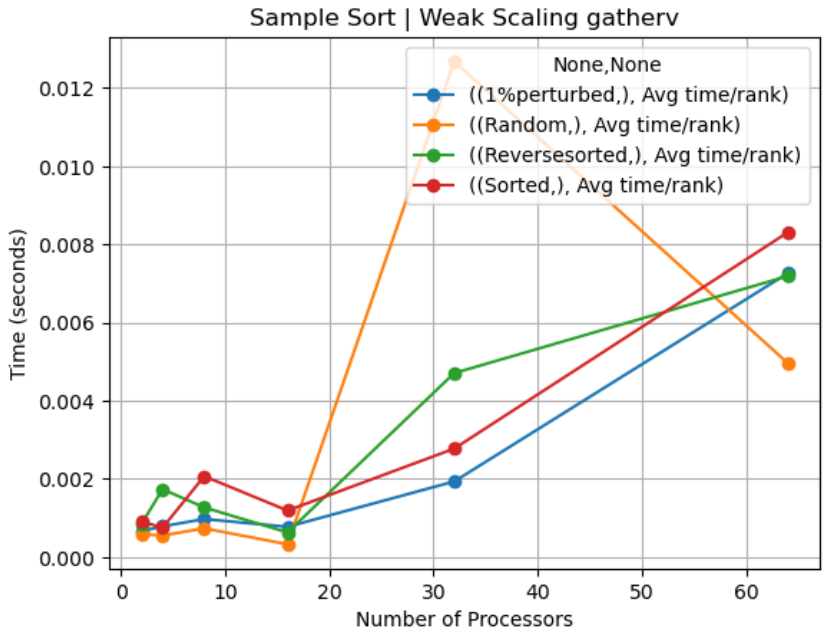

Speed Up | data_init
Time (seconds)
Threads
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

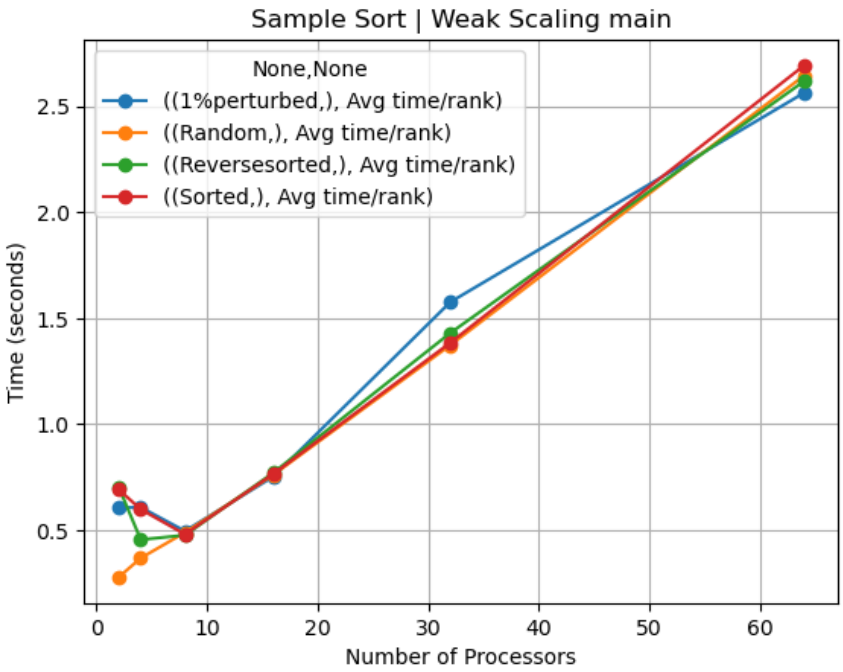

Speed Up | main
Time (seconds)
Threads
None,None
((65536, 1%perturbed), Avg time/rank)
((65536, Random), Avg time/rank)
((65536, Reversesorted), Avg time/rank)
((65536, Sorted), Avg time/rank)

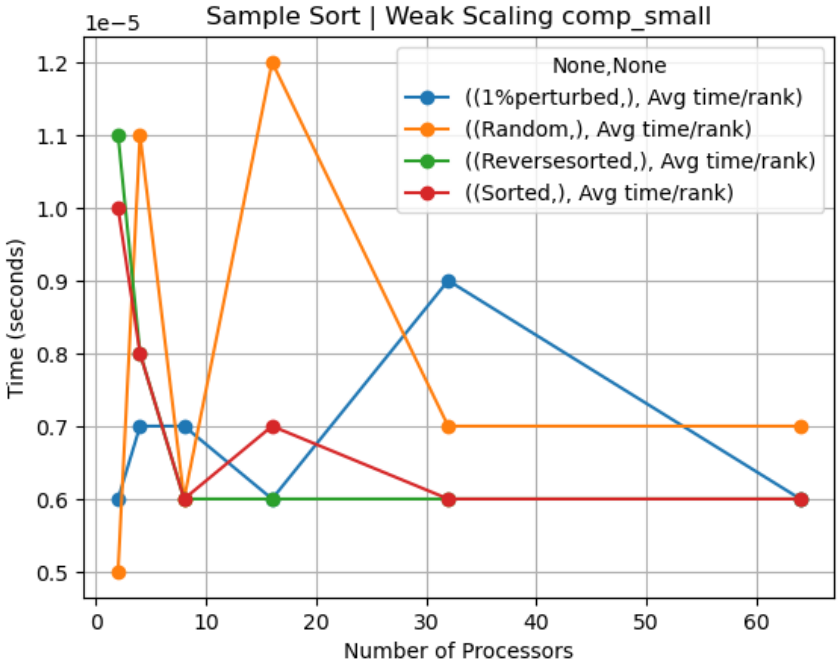


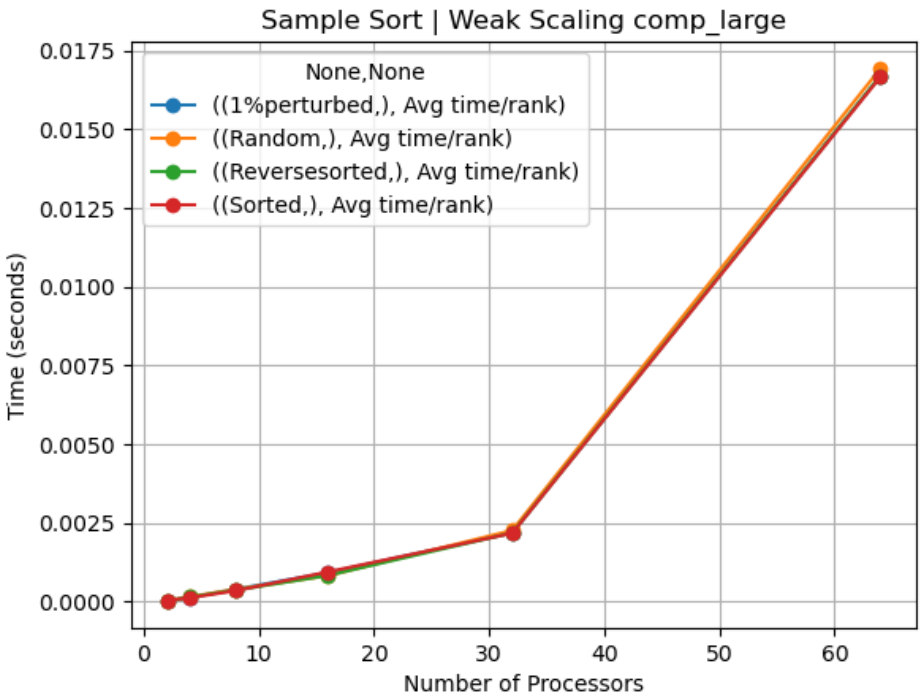
The performance graphs of the speedup further show the inefficiency of communications between nodes. One interesting observation is that for 16 nodes the communication performance is vastly improved. With that being said, I believe that this shows that the communication functions could be improved to have that same communication speedup for all numbers of processors to better show the benefits of parallelism.

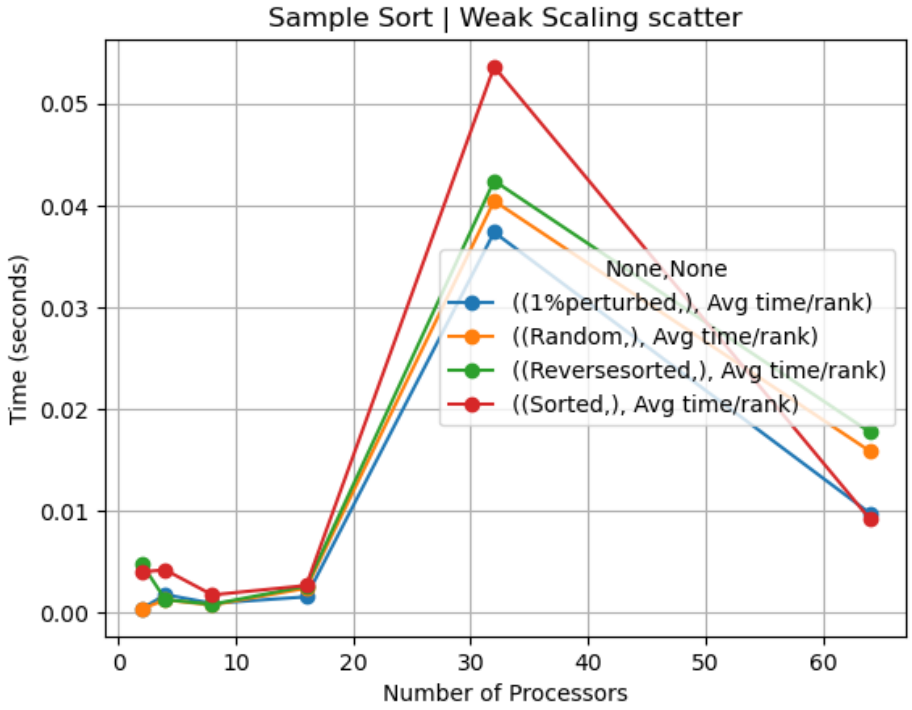
Weak Scaling:

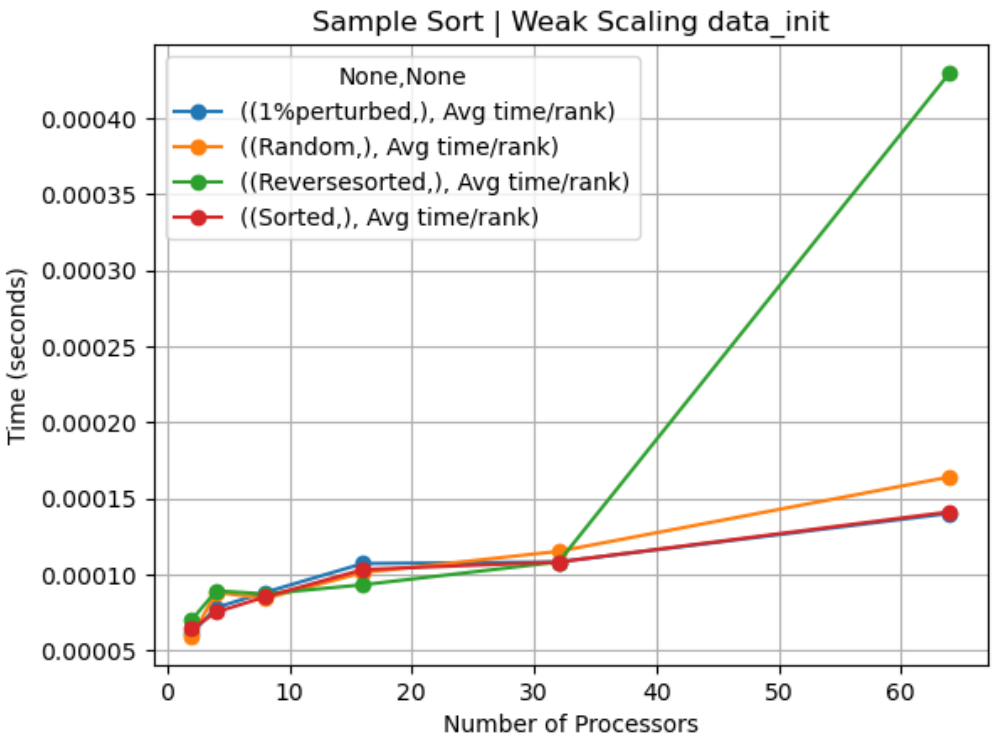


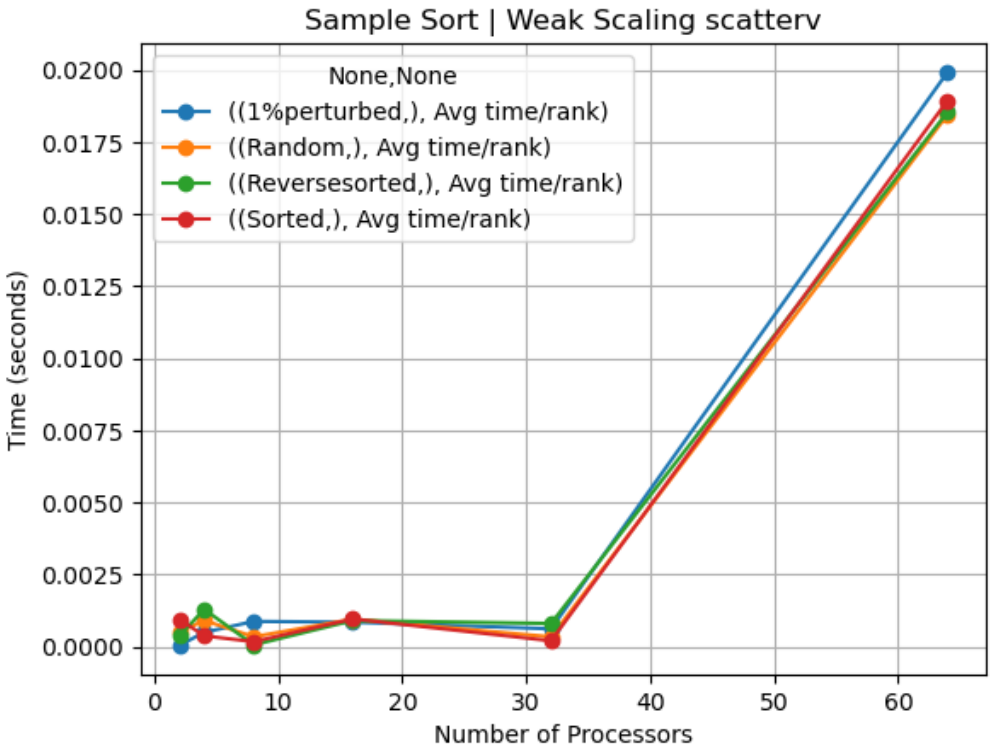

Sample Sort | Weak Scaling main
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
Time (seconds)
Number of Processors
0
10
20
30
40
50
60
0.5
1.0
1.5
2.0
2.5


Sample Sort | Weak Scaling comp_small
1e−5
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
Time (seconds)
Number of Processors
0
10
20
30
40
50
60
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2


Sample Sort | Weak Scaling comp_large
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
Time (seconds)
Number of Processors
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
10
20
30
40
50
60


Sample Sort | Weak Scaling scatter
None,None
((1%perturbed,), Avg time/rank)
((Random,), Avg time/rank)
((Reversesorted,), Avg time/rank)
((Sorted,), Avg time/rank)
Time (seconds)
Number of Processors
0.05
0.04
0.03
0.02
0.01
0.00
0
10
20
30
40
50
60



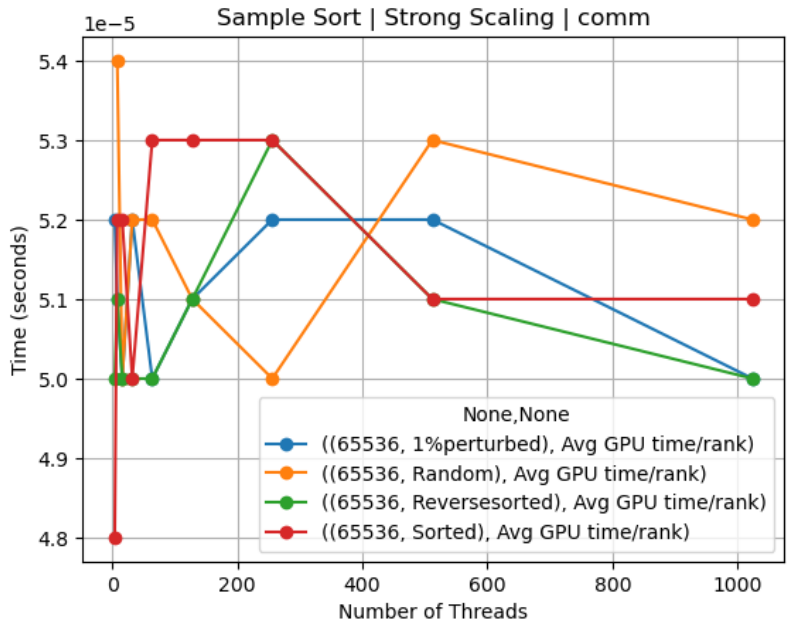


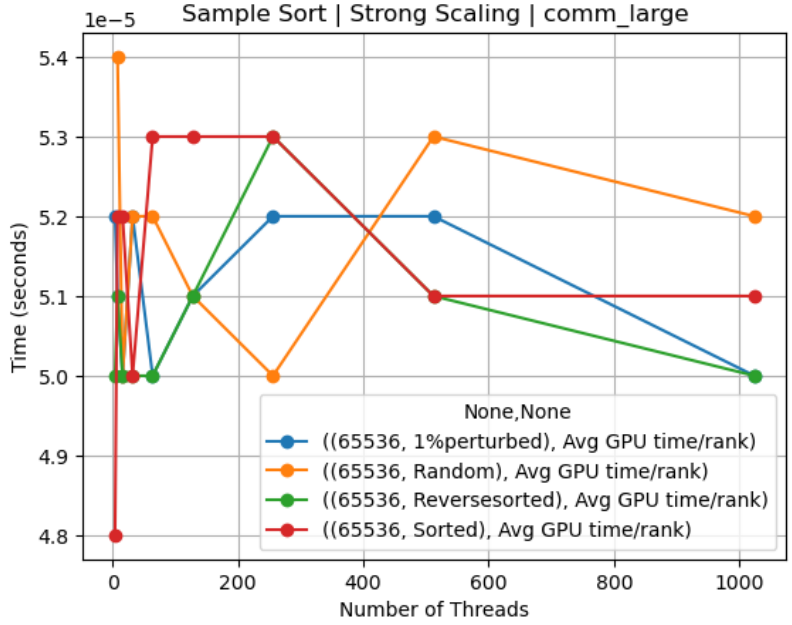
The weak scaling graph shows that as soon as the amount of node being using goes above 32 the communication performance suffers greatly. This may be due to the aforementioned issue in communication. It is important to node that the main graph does not reflect this because it

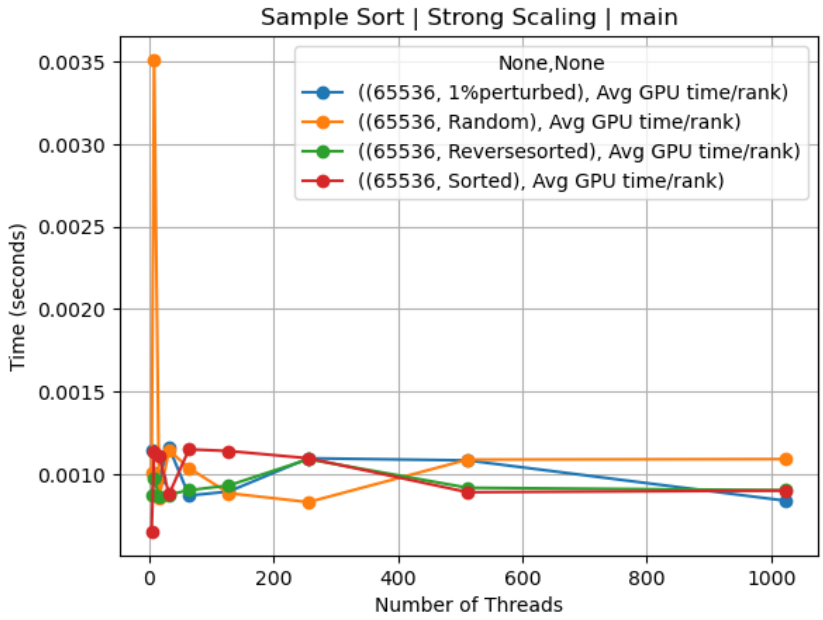
aggregates all of the performance within the program rather than just the communication performance and include some linear computation in that section.

Sample Sort CUDA:

Strong Scaling:

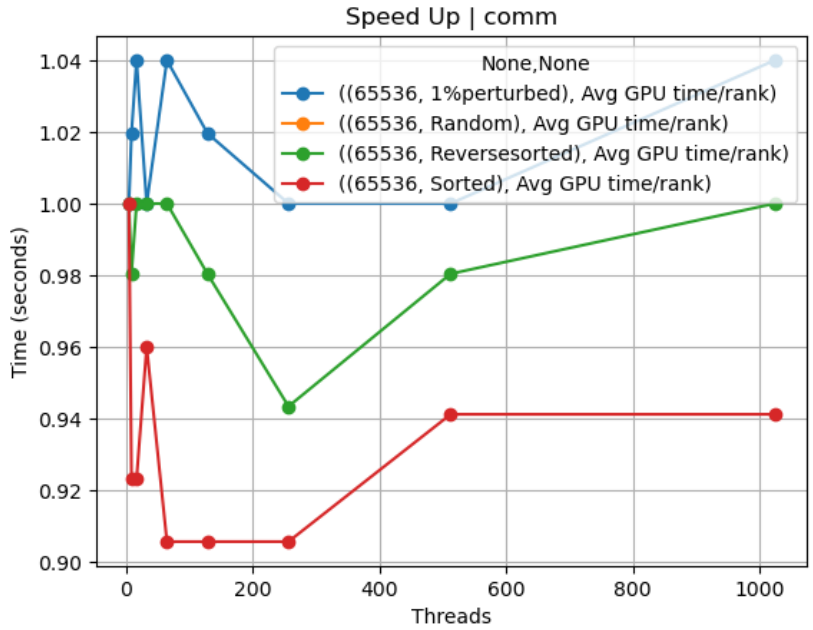


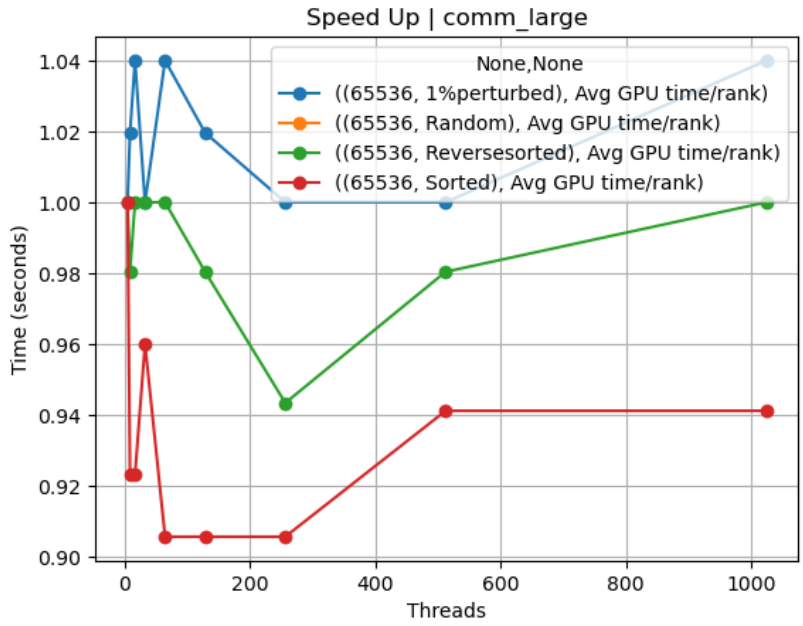


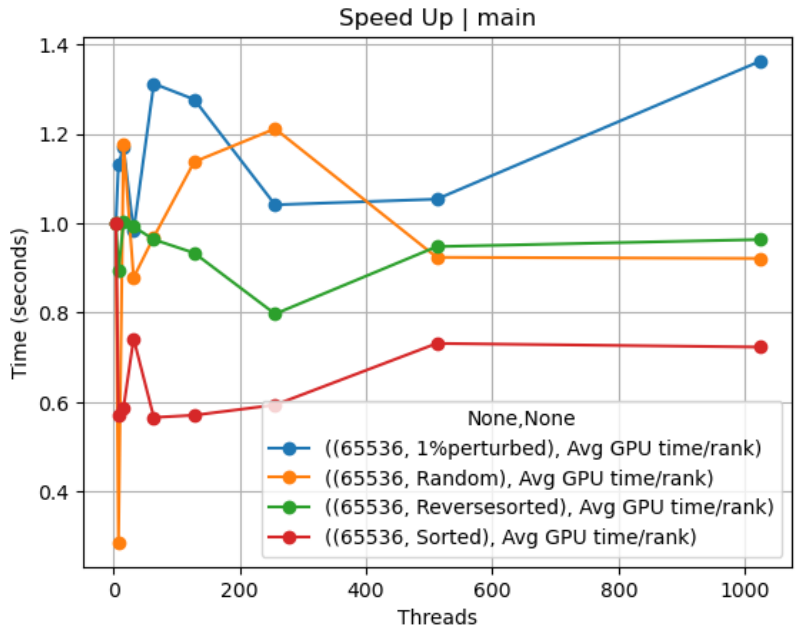


The cuda implementation shows a lot better performance than the mpi implementation. As the number of threads increases the comm and main performance shows slight improvement. Comm shows the largest improvement out of all of the different metrics.

Speedup:

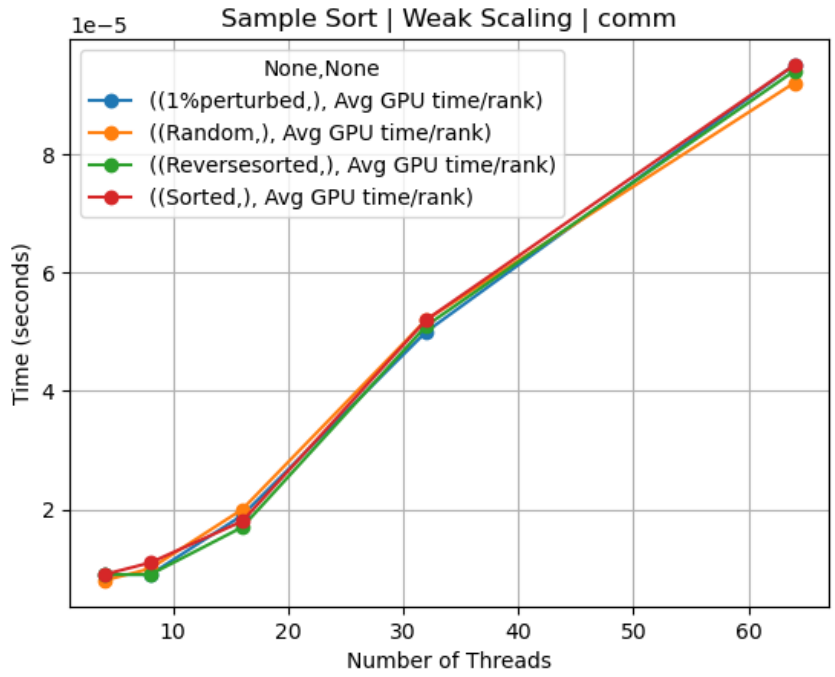


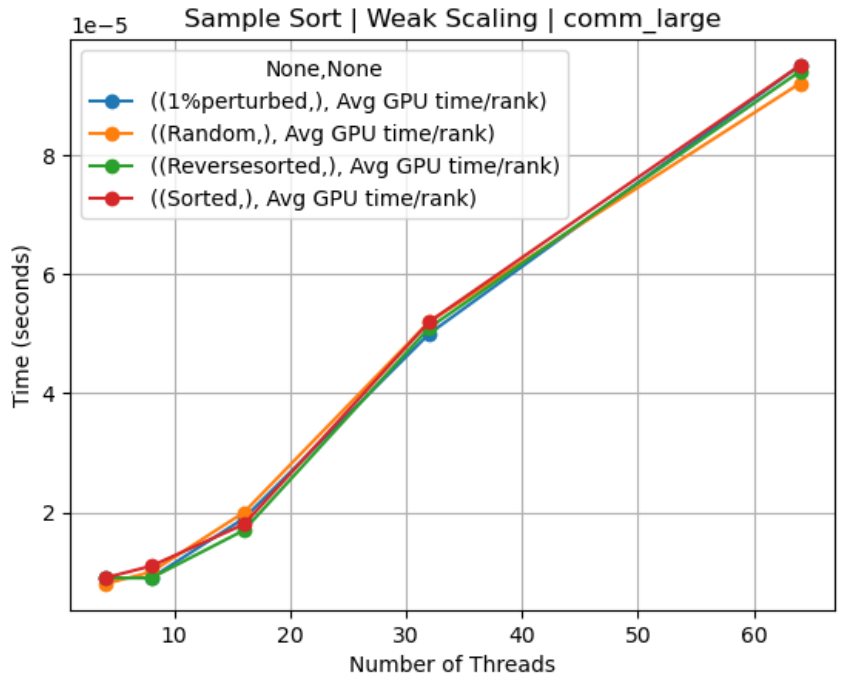


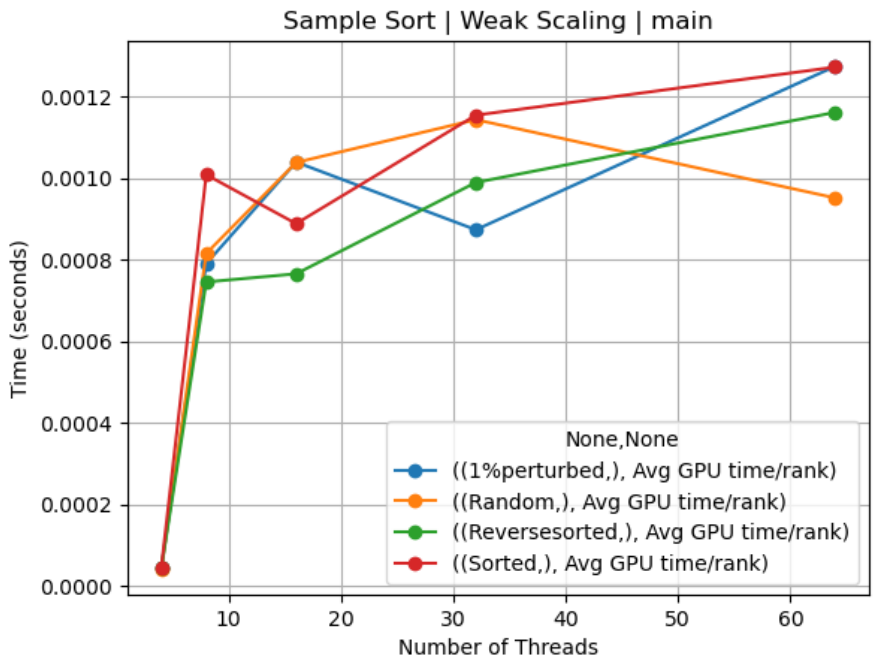


The speedup shows the performance improvement gained from increasing the number of processors. I noticed an outlier with a dip in performance in the random sort. I do not believe that this is representative of an average run, rather it is just an instance of varying performance.

Weak Scaling:



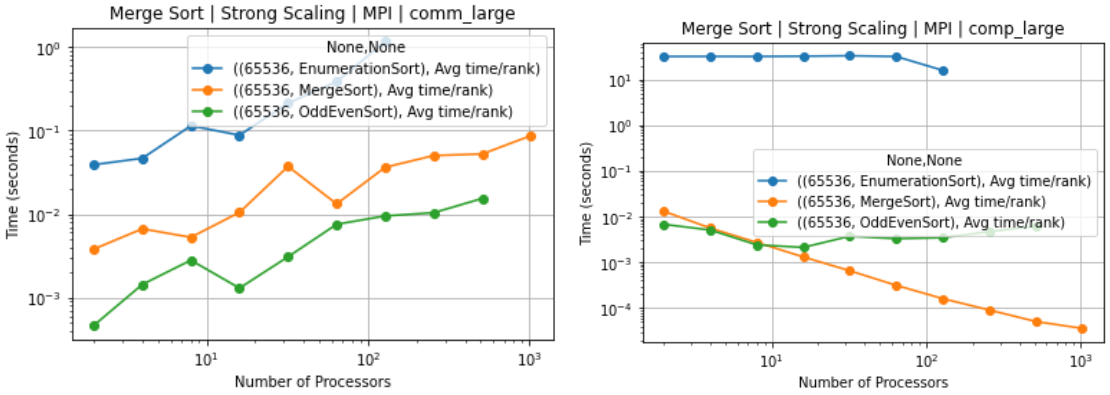


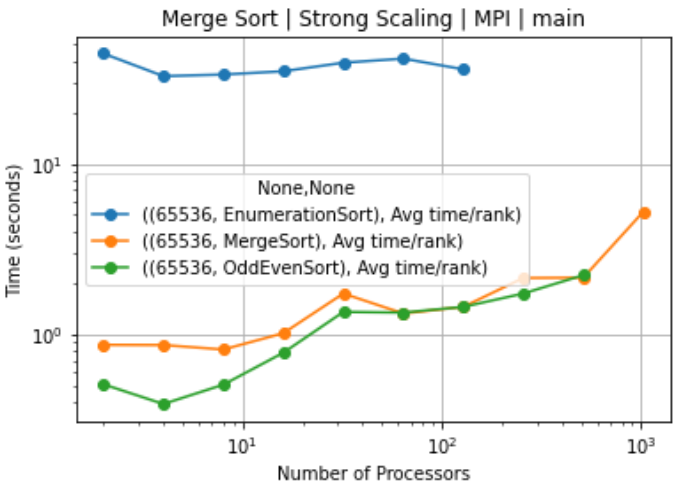


The result from the weak scaling are roughly what I would expect. The comm performance is mainly linear. I notice that the main scaling is less linear. I assume that this is because data initialization is linear and therefore as we increase the number of N, it takes a bit longer to generate those values.

MPI Comparison:

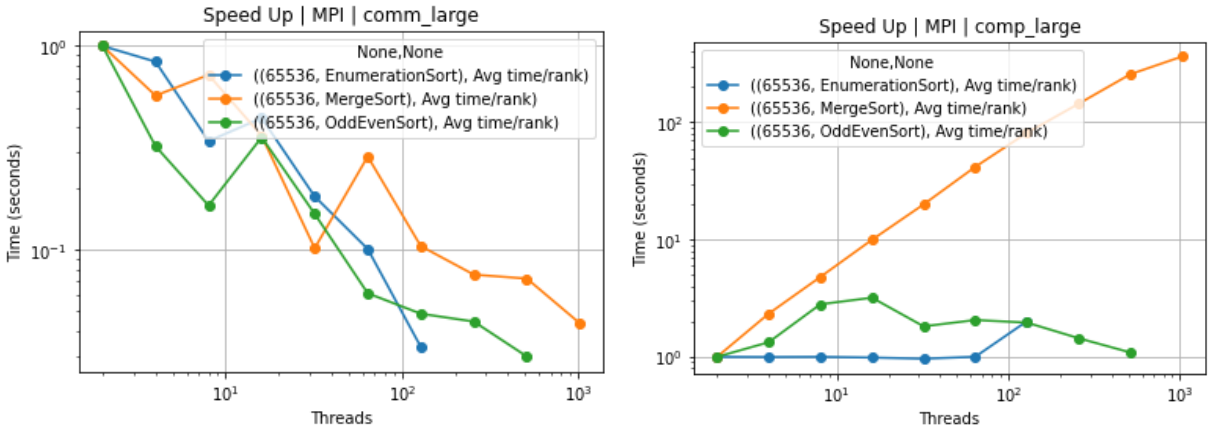
Strong Scaling
(No SampleSort due to issues in metadata)

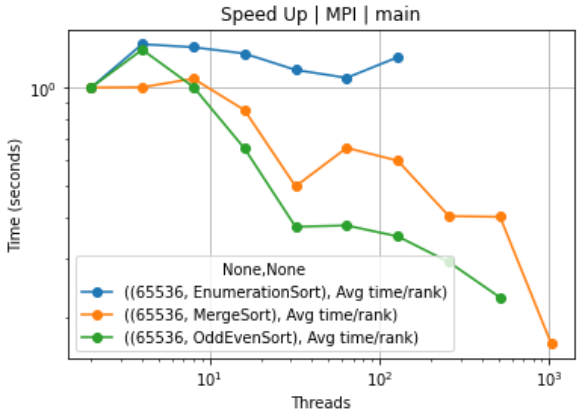




For the Strong Scaling between all the algorithms there seems to be a clear increase in the time it takes for the comm_large to run as the number of processors increase, where as, the time it takes to run comp_large decreases over time, and while Enumeration Sort doesn't have as much data it shows that it begins to decrease and may go even further down.

Strong Scaling Speed Up

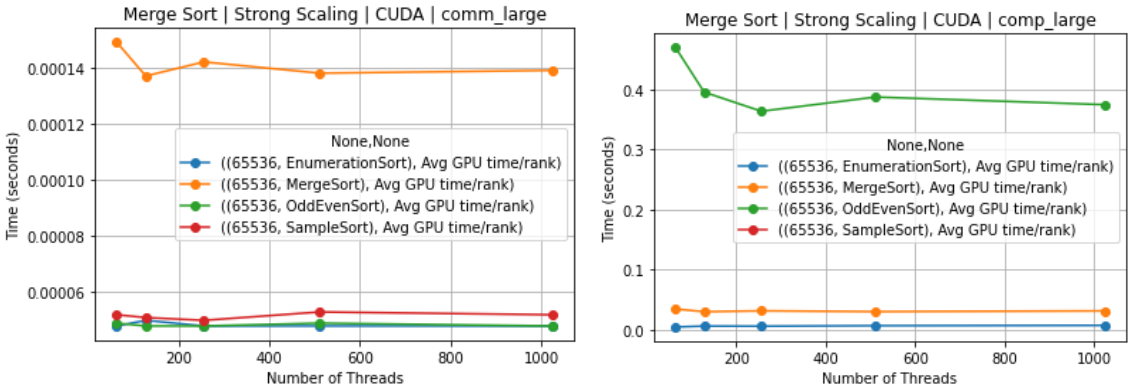


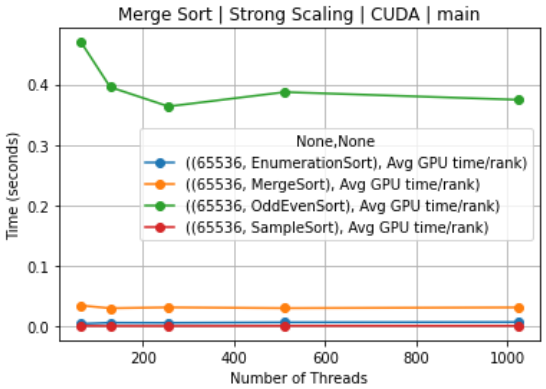


The speed up has a similar showing in that there is little to no speed up for the Algorithms in the comm_large, but comp_large does show speed up for each of the present Algorithms.

CUDA Comparison:

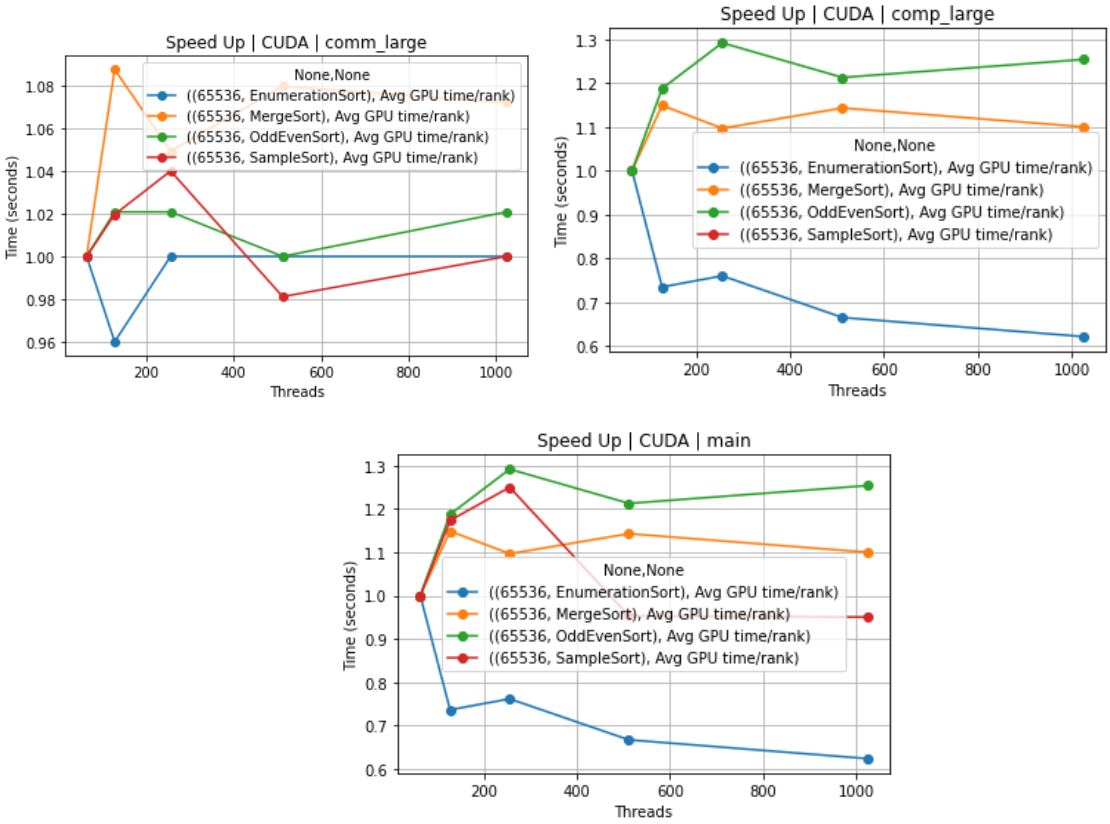
Strong Scaling





Here it is a bit difficult to see due to the differences in the Algorithms, but it is clear that for comm_large, the MergeSort takes quite a bit of time, whereas for comp_large, OddEven Sort seems to take the most time. And further analysis is a little difficult given the scale of the algorithms when put together.

Strong Scaling Speed Up



Finally the speed up here shows a bit of a mixture for both comm_large and comp_large, though while never substantial, there are slight increases over time as the processors increase. Seemingly more so in OddEven and MergeSort.