In [96]:
```
import sys
sys.path.append("/scratch/group/csce435-f23/python-3.8.17/lib/python3.8/site-packages"
sys.path.append("/scratch/group/csce435-f23/thicket")
from glob import glob

import matplotlib.pyplot as plt
import pandas as pd

import thicket as th

pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)
```

Weak Scaling

(increase problem size, increase number of processors)

Random

In [97]:
```
tkrand = th.Thicket.from_caliperreader(glob("cali_data_missingLast2ArraySizes/*-0.cali
tkrand.dataframe = tkrand.dataframe.drop(["nid", "spot.channel", "Total time", "Min ti
```

In [98]:
```
gbrand = tkrand.groupby("InputSize")
```

```
5  thickets created...
{65536: <thicket.thicket.Thicket object at 0x2b16baadde20>, 262144: <thicket.thicket.
Thicket object at 0x2b16b9e9cd60>, 1048576: <thicket.thicket.Thicket object at 0x2b16
bae542e0>, 4194304: <thicket.thicket.Thicket object at 0x2b16bad35310>, 16777216: <th
icket.thicket.Thicket object at 0x2b16bb6c0a90>}
```

In [99]:
```
ctkrand = th.Thicket.concat_thickets(
    thickets=list(gbrand.values()),
    headers=list(gbrand.keys()),
    axis="columns",
    metadata_key="num_procs"
)
```

In [100…
```
ctkrand.dataframe
```

Out[100]:

		65536	262144	1048576	4194304	16777216	
		Avg time/rank	Avg time/rank	Avg time/rank	Avg time/rank	Avg time/rank	
node	**num_procs**						
{'name': 'main', 'type': 'function'}	**2**	2.052147	8.117737	33.027831	121.367928	527.901550	
	4	0.961521	3.848648	15.711205	62.496550	251.470677	
	8	0.479038	1.922948	7.700841	30.851393	124.325947	
	16	0.243121	0.978323	3.893086	15.437189	61.818237	
	32	0.122418	0.486324	1.952422	7.799405	31.193084	
{'name': 'comm', 'type': 'function'}	**2**	0.001134	0.002199	0.006304	0.009816	0.071478	c
	4	0.002053	0.003697	0.165433	0.654567	3.178008	c
	8	0.001994	0.001907	0.005190	0.011485	0.785124	c
	16	0.001593	0.004612	0.008911	0.046832	0.207075	c
	32	0.001429	0.001712	0.004800	0.015024	0.065646	c
{'name': 'comm_large', 'type': 'function'}	**2**	0.001113	0.002177	0.006280	0.009790	0.071445	comm_
	4	0.002032	0.003676	0.165410	0.654540	3.177978	comm_
	8	0.001974	0.001887	0.005167	0.011459	0.785097	comm_
	16	0.001574	0.004592	0.008889	0.046807	0.207050	comm_
	32	0.001409	0.001692	0.004778	0.014999	0.065620	comm_
{'name': 'MPI_Gather', 'type': 'function'}	**2**	0.000292	0.001470	0.000837	0.003495	0.037223	MPI_G
	4	0.000779	0.002867	0.163102	0.646554	3.150132	MPI_G
	8	0.000474	0.001004	0.002448	0.003614	0.762056	MPI_G
	16	0.001095	0.003157	0.007311	0.037477	0.187604	MPI_G
	32	0.000226	0.001032	0.003128	0.009979	0.047520	MPI_G
{'name': 'MPI_Scatter', 'type': 'function'}	**2**	0.000136	0.000445	0.005209	0.005819	0.031566	MPI_S
	4	0.000141	0.000436	0.001855	0.005950	0.021849	MPI_S
	8	0.000186	0.000319	0.001860	0.005984	0.016691	MPI_S
	16	0.000172	0.000857	0.000829	0.007002	0.012731	MPI_S
	32	0.000350	0.000245	0.000927	0.003133	0.011305	MPI_S
{'name': 'comp', 'type': 'function'}	**2**	1.569347	6.220516	25.267192	93.085064	404.613614	
	4	0.734743	2.943333	11.902455	47.299319	189.826631	
	8	0.364472	1.465817	5.883794	23.579997	94.406360	
	16	0.183989	0.741725	2.956223	11.726656	46.955130	
	32	0.091445	0.366721	1.474866	5.883786	23.547130	
{'name': 'comp_large',	**2**	0.278110	1.099838	4.483552	16.654032	71.522268	comp_

		65536	262144	1048576	4194304	16777216	ı
		Avg time/rank	Avg time/rank	Avg time/rank	Avg time/rank	Avg time/rank	
node	num_procs						
'type': 'function'}	4	0.130919	0.522903	2.121184	8.394996	33.901237	comp_
	8	0.064885	0.261078	1.047296	4.204308	16.893292	comp_
	16	0.032673	0.131796	0.525570	2.097528	8.389140	comp_
	32	0.016197	0.065267	0.262972	1.050130	4.210633	comp_
{'name': 'comp_small', 'type': 'function'}	2	0.271780	1.068288	4.353527	16.092318	69.102072	comp_
	4	0.127916	0.507440	2.055241	8.116996	32.662728	comp_
	8	0.063125	0.253266	1.014882	4.068188	16.319229	comp_
	16	0.031799	0.128362	0.509073	2.026077	8.094242	comp_
	32	0.015793	0.063554	0.254951	1.014948	4.060336	comp_
{'name': 'correctness_check', 'type': 'function'}	2	0.000194	0.000737	0.002903	0.011595	0.046430	correctness_
	4	0.000190	0.000735	0.002886	0.011633	0.046784	correctness_
	8	0.000189	0.000734	0.002893	0.011753	0.046945	correctness_
	16	0.000194	0.000736	0.002912	0.011638	0.046937	correctness_
	32	0.000210	0.000737	0.002896	0.011621	0.046933	correctness_
{'name': 'data_init', 'type': 'function'}	2	0.001585	0.006276	0.024685	0.097345	0.392283	dat
	4	0.001600	0.006286	0.024658	0.097483	0.388345	dat
	8	0.001590	0.006283	0.024684	0.097551	0.388656	dat
	16	0.001609	0.006294	0.024794	0.098468	0.389607	dat
	32	0.001590	0.006334	0.024836	0.098047	0.391677	dat

In [101…

```
ctkrand.dataframe = ctkrand.dataframe.reset_index().drop(("node"), axis=1)
ctkrand.dataframe = ctkrand.dataframe.rename({("name", ""): "name", ("num_procs", ""):
```

```
<ipython-input-101-3c3f415ea473>:1: PerformanceWarning: dropping on a non-lexsorted m
ulti-index without a level parameter may impact performance.
  ctkrand.dataframe = ctkrand.dataframe.reset_index().drop(("node"), axis=1)
```

In [102…

```
ctkrand.dataframe
```

Out[102]:

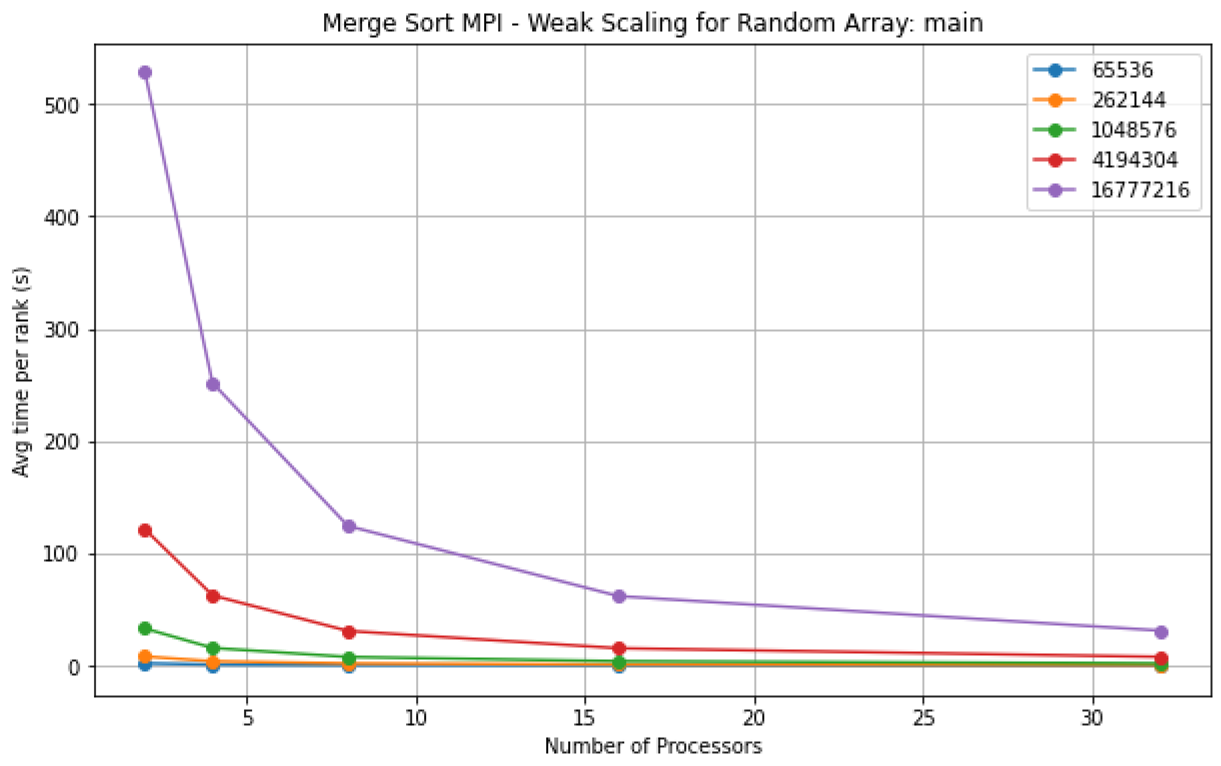
		65536	262144	1048576	4194304	16777216
		Avg time/rank	Avg time/rank	Avg time/rank	Avg time/rank	Avg time/rank
name	**num_procs**					
main	**2**	2.052147	8.117737	33.027831	121.367928	527.901550
	4	0.961521	3.848648	15.711205	62.496550	251.470677
	8	0.479038	1.922948	7.700841	30.851393	124.325947
	16	0.243121	0.978323	3.893086	15.437189	61.818237
	32	0.122418	0.486324	1.952422	7.799405	31.193084
comm	**2**	0.001134	0.002199	0.006304	0.009816	0.071478
	4	0.002053	0.003697	0.165433	0.654567	3.178008
	8	0.001994	0.001907	0.005190	0.011485	0.785124
	16	0.001593	0.004612	0.008911	0.046832	0.207075
	32	0.001429	0.001712	0.004800	0.015024	0.065646
comm_large	**2**	0.001113	0.002177	0.006280	0.009790	0.071445
	4	0.002032	0.003676	0.165410	0.654540	3.177978
	8	0.001974	0.001887	0.005167	0.011459	0.785097
	16	0.001574	0.004592	0.008889	0.046807	0.207050
	32	0.001409	0.001692	0.004778	0.014999	0.065620
MPI_Gather	**2**	0.000292	0.001470	0.000837	0.003495	0.037223
	4	0.000779	0.002867	0.163102	0.646554	3.150132
	8	0.000474	0.001004	0.002448	0.003614	0.762056
	16	0.001095	0.003157	0.007311	0.037477	0.187604
	32	0.000226	0.001032	0.003128	0.009979	0.047520
MPI_Scatter	**2**	0.000136	0.000445	0.005209	0.005819	0.031566
	4	0.000141	0.000436	0.001855	0.005950	0.021849
	8	0.000186	0.000319	0.001860	0.005984	0.016691
	16	0.000172	0.000857	0.000829	0.007002	0.012731
	32	0.000350	0.000245	0.000927	0.003133	0.011305
comp	**2**	1.569347	6.220516	25.267192	93.085064	404.613614
	4	0.734743	2.943333	11.902455	47.299319	189.826631
	8	0.364472	1.465817	5.883794	23.579997	94.406360
	16	0.183989	0.741725	2.956223	11.726656	46.955130
	32	0.091445	0.366721	1.474866	5.883786	23.547130
comp_large	**2**	0.278110	1.099838	4.483552	16.654032	71.522268

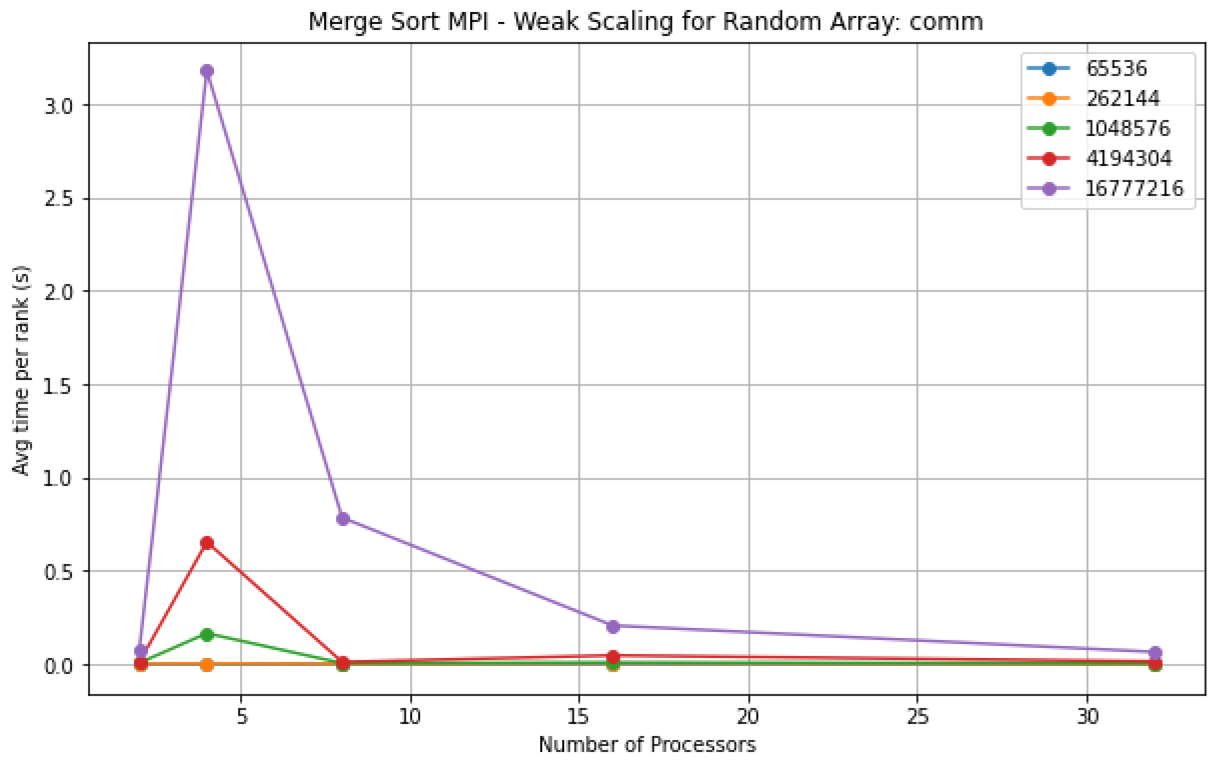
		65536	262144	1048576	4194304	16777216
		Avg time/rank	Avg time/rank	Avg time/rank	Avg time/rank	Avg time/rank
name	**num_procs**					
	4	0.130919	0.522903	2.121184	8.394996	33.901237
	8	0.064885	0.261078	1.047296	4.204308	16.893292
	16	0.032673	0.131796	0.525570	2.097528	8.389140
	32	0.016197	0.065267	0.262972	1.050130	4.210633
comp_small	**2**	0.271780	1.068288	4.353527	16.092318	69.102072
	4	0.127916	0.507440	2.055241	8.116996	32.662728
	8	0.063125	0.253266	1.014882	4.068188	16.319229
	16	0.031799	0.128362	0.509073	2.026077	8.094242
	32	0.015793	0.063554	0.254951	1.014948	4.060336
correctness_check	**2**	0.000194	0.000737	0.002903	0.011595	0.046430
	4	0.000190	0.000735	0.002886	0.011633	0.046784
	8	0.000189	0.000734	0.002893	0.011753	0.046945
	16	0.000194	0.000736	0.002912	0.011638	0.046937
	32	0.000210	0.000737	0.002896	0.011621	0.046933
data_init	**2**	0.001585	0.006276	0.024685	0.097345	0.392283
	4	0.001600	0.006286	0.024658	0.097483	0.388345
	8	0.001590	0.006283	0.024684	0.097551	0.388656
	16	0.001609	0.006294	0.024794	0.098468	0.389607
	32	0.001590	0.006334	0.024836	0.098047	0.391677

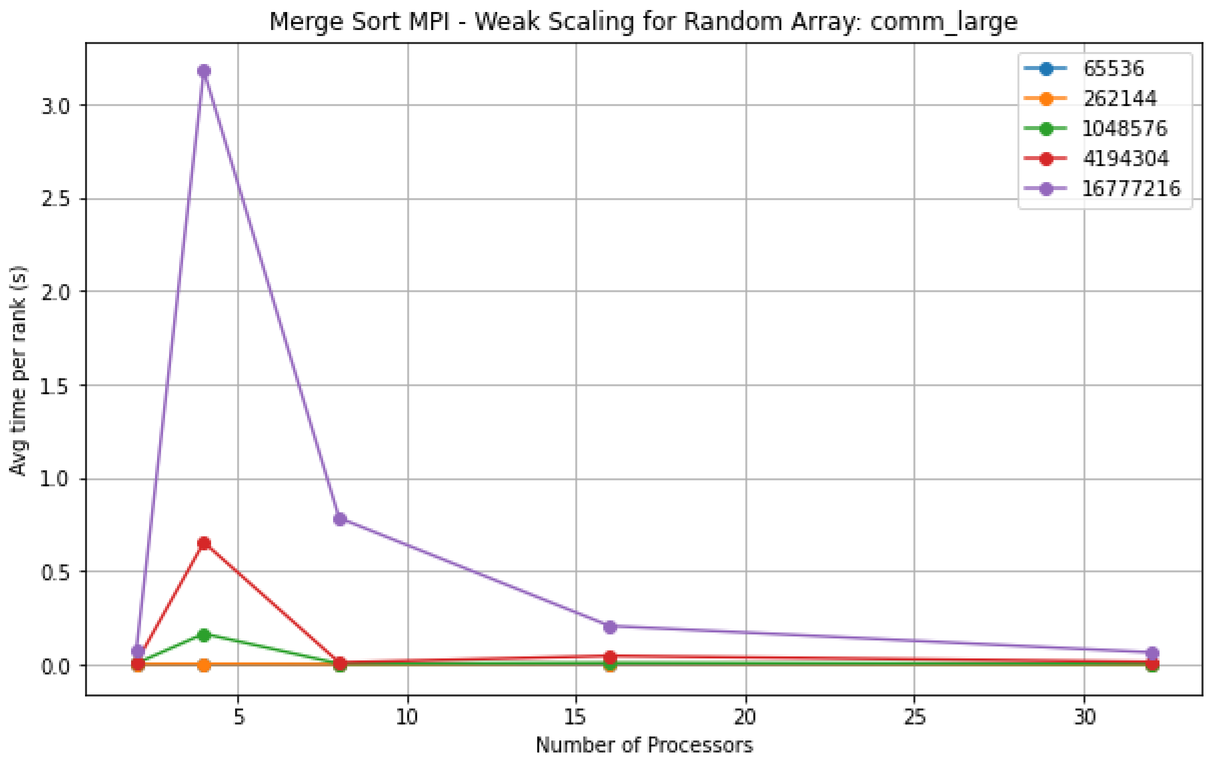
In [103…
```
main = ctkrand.dataframe.loc["main"]
comm = ctkrand.dataframe.loc["comm"]
comm_large = ctkrand.dataframe.loc["comm_large"]
MPI_Gather = ctkrand.dataframe.loc["MPI_Gather"]
MPI_Scatter = ctkrand.dataframe.loc["MPI_Scatter"]
comp = ctkrand.dataframe.loc["comp"]
comp_large = ctkrand.dataframe.loc["comp_large"]
comp_small = ctkrand.dataframe.loc["comp_small"]
correctness_check = ctkrand.dataframe.loc["correctness_check"]
data_init = ctkrand.dataframe.loc["data_init"]
```

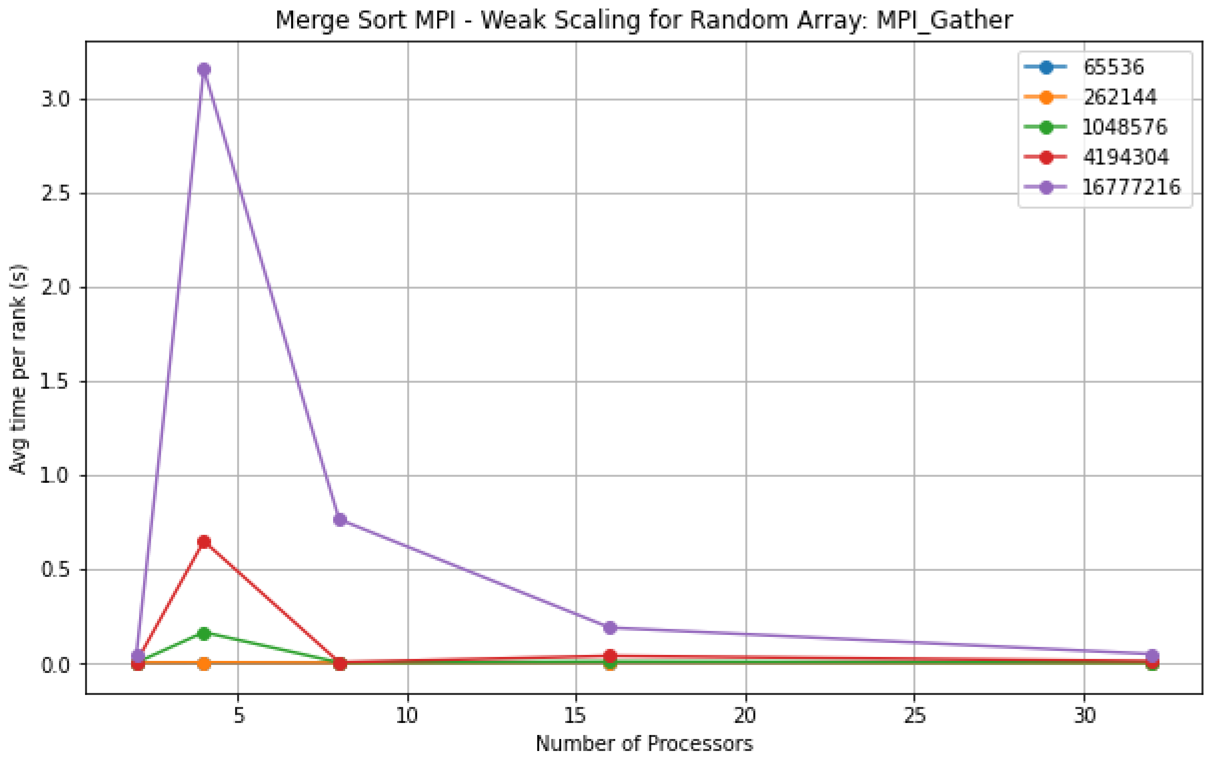
In [104…
```
regions = [main, comm, comm_large, MPI_Gather, MPI_Scatter, comp, comp_large, comp_sma
names = ["main", "comm", "comm_large", "MPI_Gather", "MPI_Scatter", "comp", "comp_larg
```

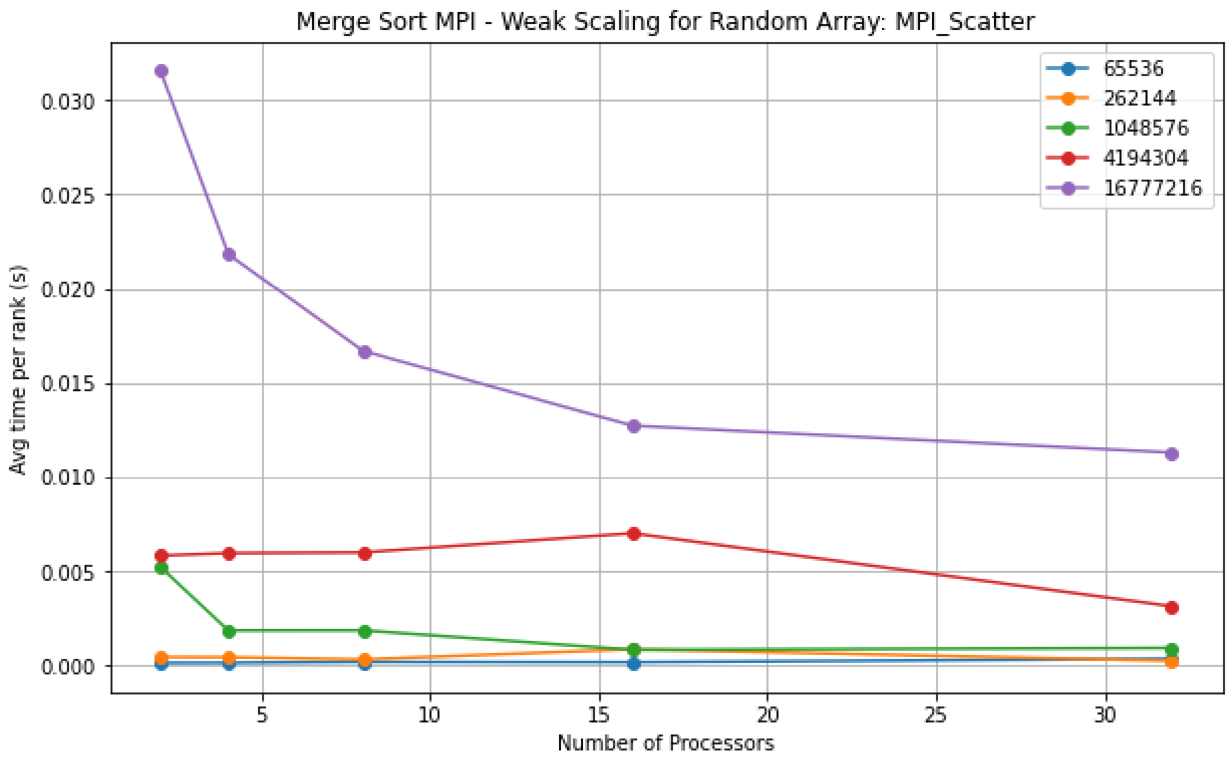
In [105…
```
for region, name in zip(regions, names):
    plt.figure(figsize=(10, 6))  # Adjust the figure size if needed
    legend_labels = []
    for column in region.columns:
        first_index = column[0]  # Extract the first index
        legend_labels.append(first_index)
        plt.plot(region.index, region.xs(column, axis=1), marker='o', label=column)

    plt.xlabel('Number of Processors')
    plt.ylabel('Avg time per rank (s)')
    plt.title(f'Merge Sort MPI - Weak Scaling for Random Array: {name}')
    plt.legend(legend_labels)
    plt.grid(True)
    plt.show()
```

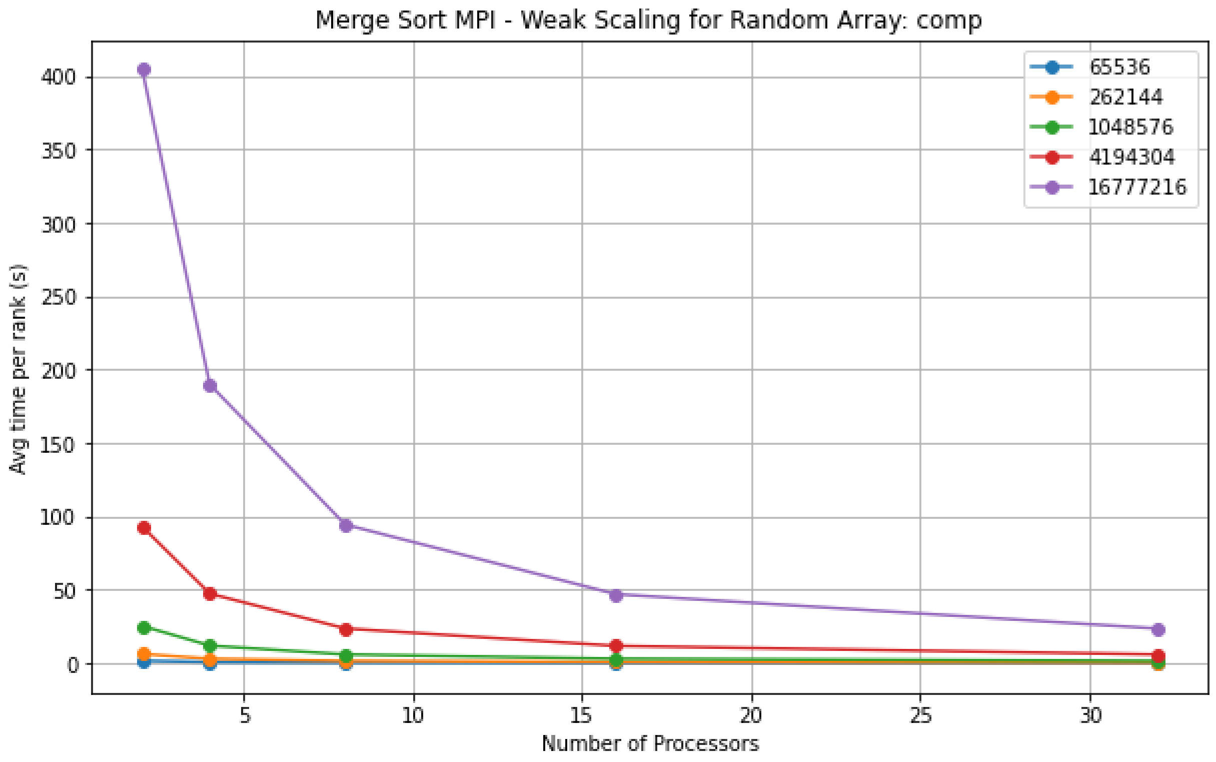


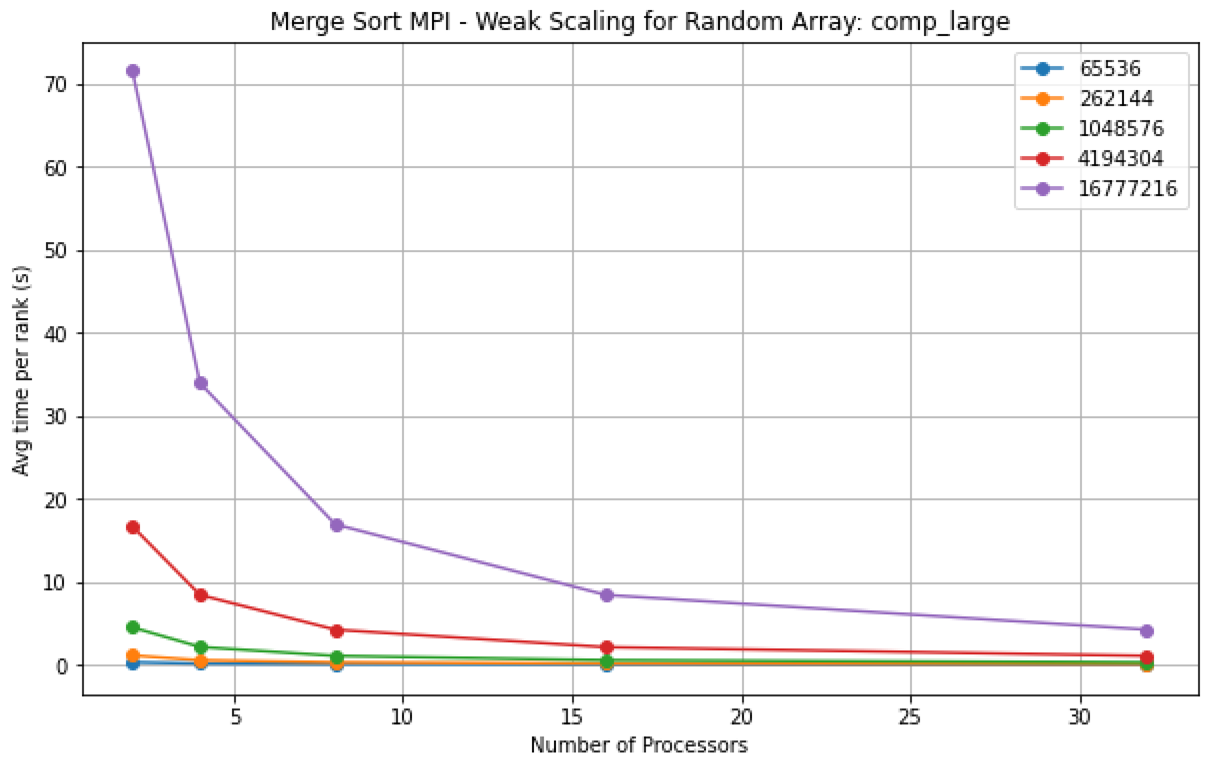


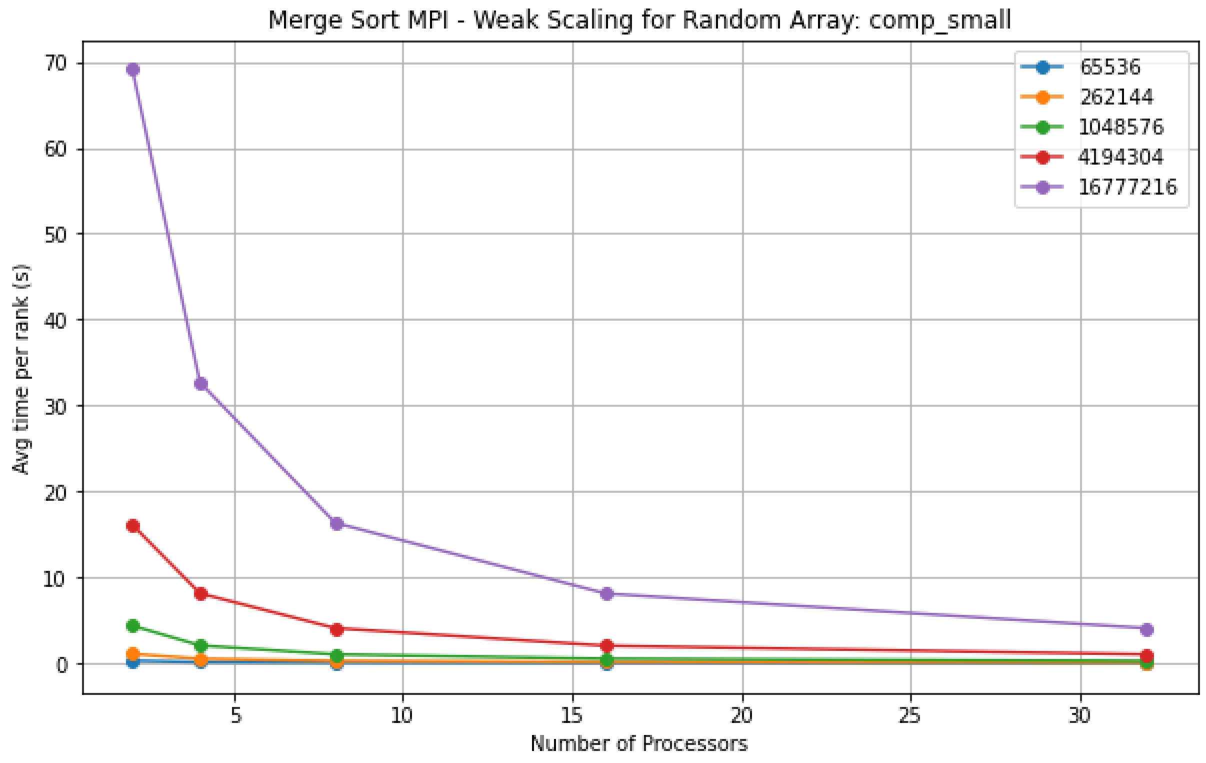

Merge Sort MPI - Weak Scaling for Random Array: comm_large
Avg time per rank (s)
Number of Processors
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216

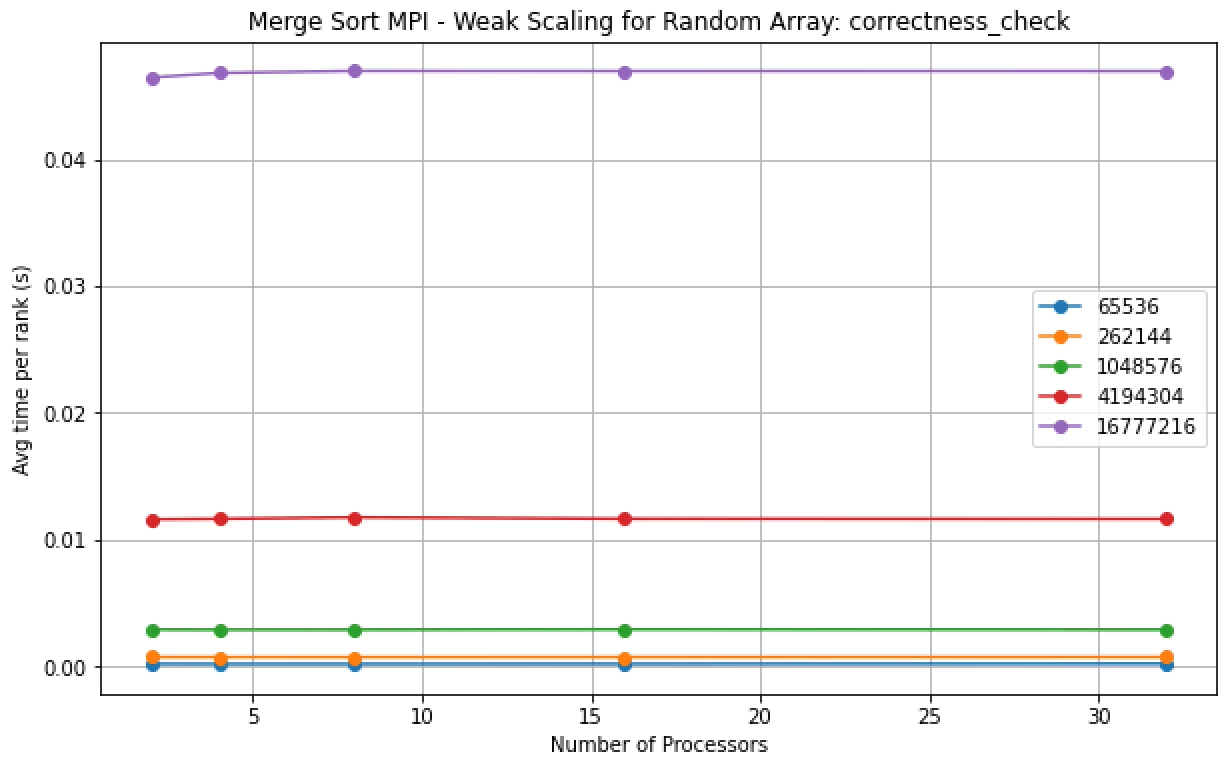

Merge Sort MPI - Weak Scaling for Random Array: MPI_Gather
Avg time per rank (s)
Number of Processors
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216

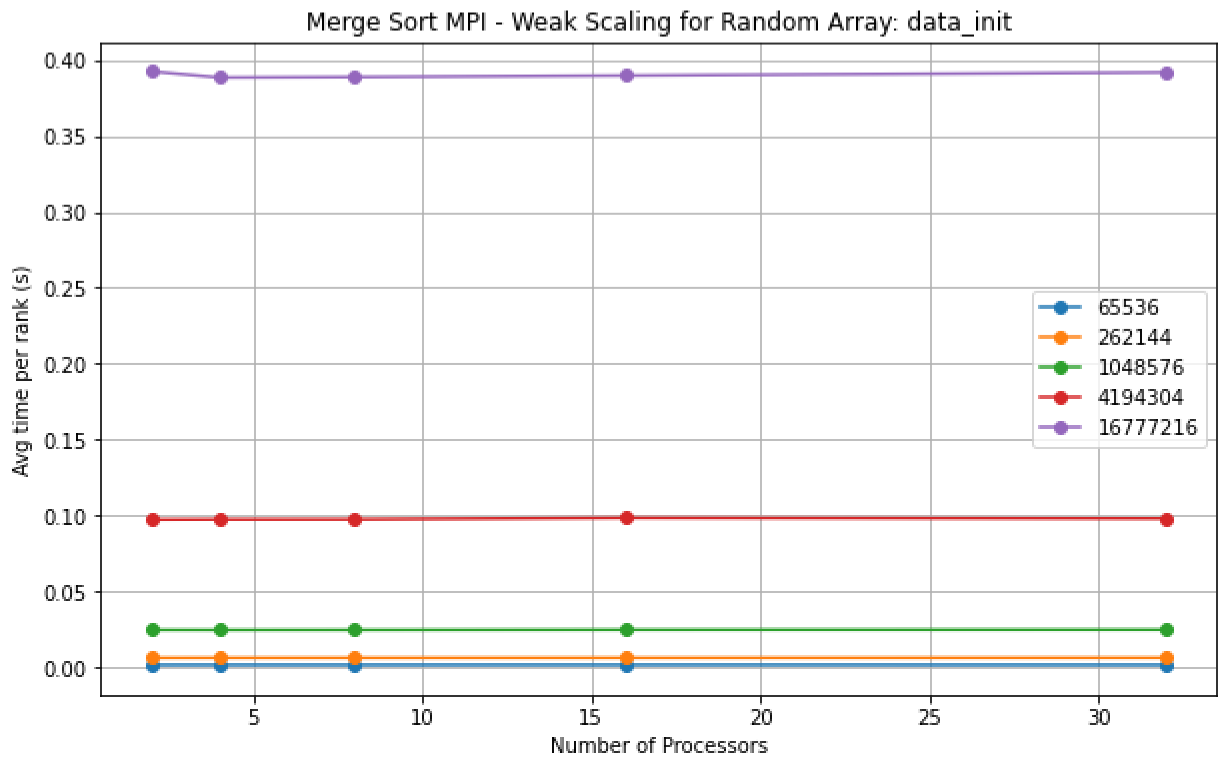

Merge Sort MPI - Weak Scaling for Random Array: MPI_Scatter
65536
262144
1048576
4194304
16777216
Avg time per rank (s)
Number of Processors
0.000
0.005
0.010
0.015
0.020
0.025
0.030
5
10
15
20
25
30


Merge Sort MPI - Weak Scaling for Random Array: comp
65536
262144
1048576
4194304
16777216
Avg time per rank (s)
Number of Processors
0
50
100
150
200
250
300
350
400
5
10
15
20
25
30


Merge Sort MPI - Weak Scaling for Random Array: comp_large
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0
10
20
30
40
50
60
70
5
10
15
20
25
30


Merge Sort MPI - Weak Scaling for Random Array: comp_small
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0
10
20
30
40
50
60
70
5
10
15
20
25
30





Sorted

In [106…

```
tksorted = th.Thicket.from_caliperreader(glob("cali_data_missingLast2ArraySizes/*-1.ca
tksorted.dataframe = tksorted.dataframe.drop(["nid", "spot.channel", "Total time", "Mi

gbsorted = tksorted.groupby("InputSize")

ctksorted = th.Thicket.concat_thickets(
    thickets=list(gbsorted.values()),
    headers=list(gbsorted.keys()),
    axis="columns",
    metadata_key="num_procs"
)
```

```
5  thickets created...
{65536: <thicket.thicket.Thicket object at 0x2b16bb235a00>, 262144: <thicket.thicket.
Thicket object at 0x2b16bb7743a0>, 1048576: <thicket.thicket.Thicket object at 0x2b16
bb796700>, 4194304: <thicket.thicket.Thicket object at 0x2b16b9f8ba00>, 16777216: <th
icket.thicket.Thicket object at 0x2b16b9fab670>}
```

In [107…

```
ctksorted.dataframe = ctksorted.dataframe.reset_index().drop(("node"), axis=1)
ctksorted.dataframe = ctksorted.dataframe.rename({("name", ""): "name", ("num_procs",

main = ctksorted.dataframe.loc["main"]
comm = ctksorted.dataframe.loc["comm"]
comm_large = ctksorted.dataframe.loc["comm_large"]
MPI_Gather = ctksorted.dataframe.loc["MPI_Gather"]
MPI_Scatter = ctksorted.dataframe.loc["MPI_Scatter"]
comp = ctksorted.dataframe.loc["comp"]
comp_large = ctksorted.dataframe.loc["comp_large"]
comp_small = ctksorted.dataframe.loc["comp_small"]
correctness_check = ctksorted.dataframe.loc["correctness_check"]
data_init = ctksorted.dataframe.loc["data_init"]
```

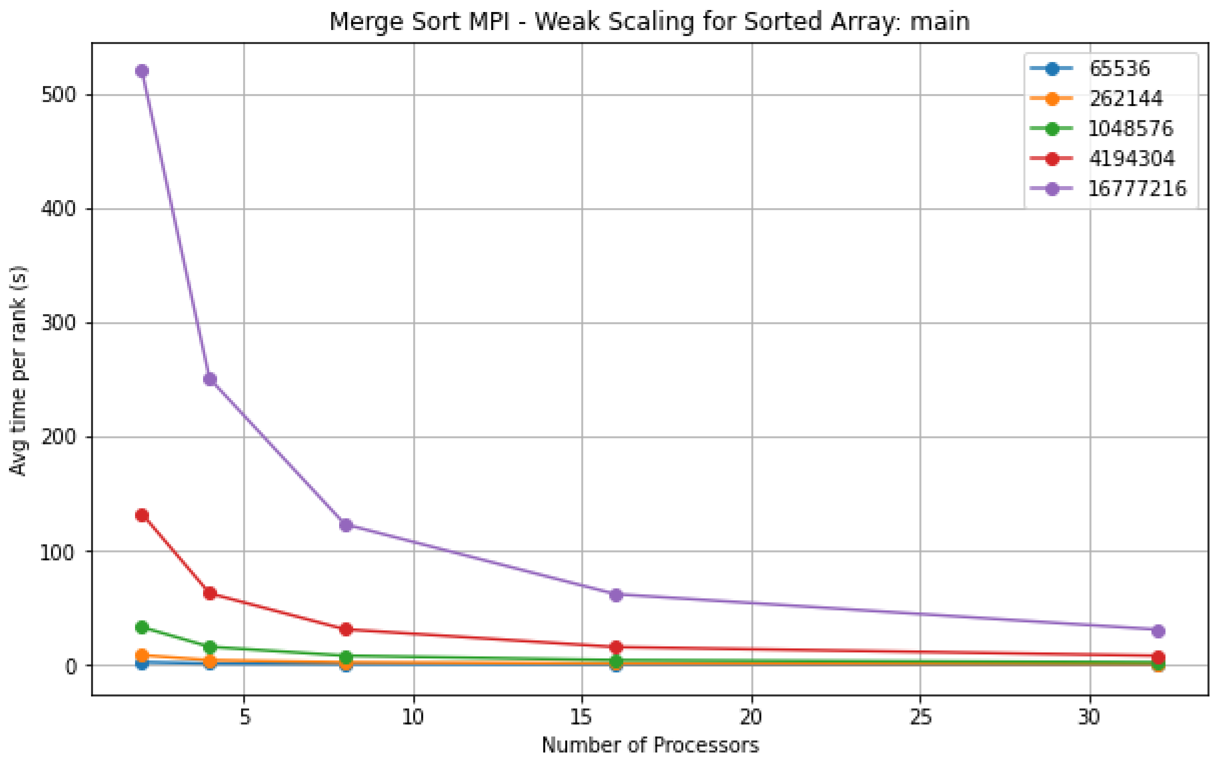
```
<ipython-input-107-0be137252a44>:1: PerformanceWarning: dropping on a non-lexsorted m
ulti-index without a level parameter may impact performance.
  ctksorted.dataframe = ctksorted.dataframe.reset_index().drop(("node"), axis=1)
```

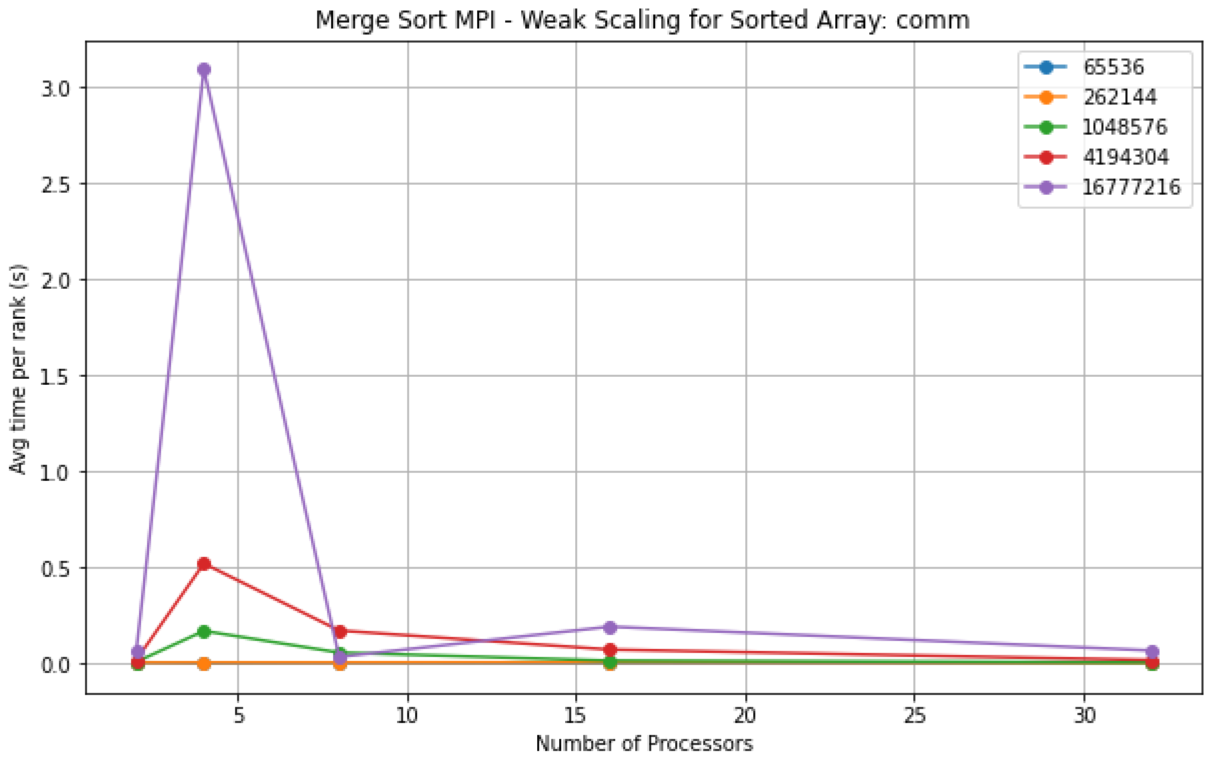
In [108…

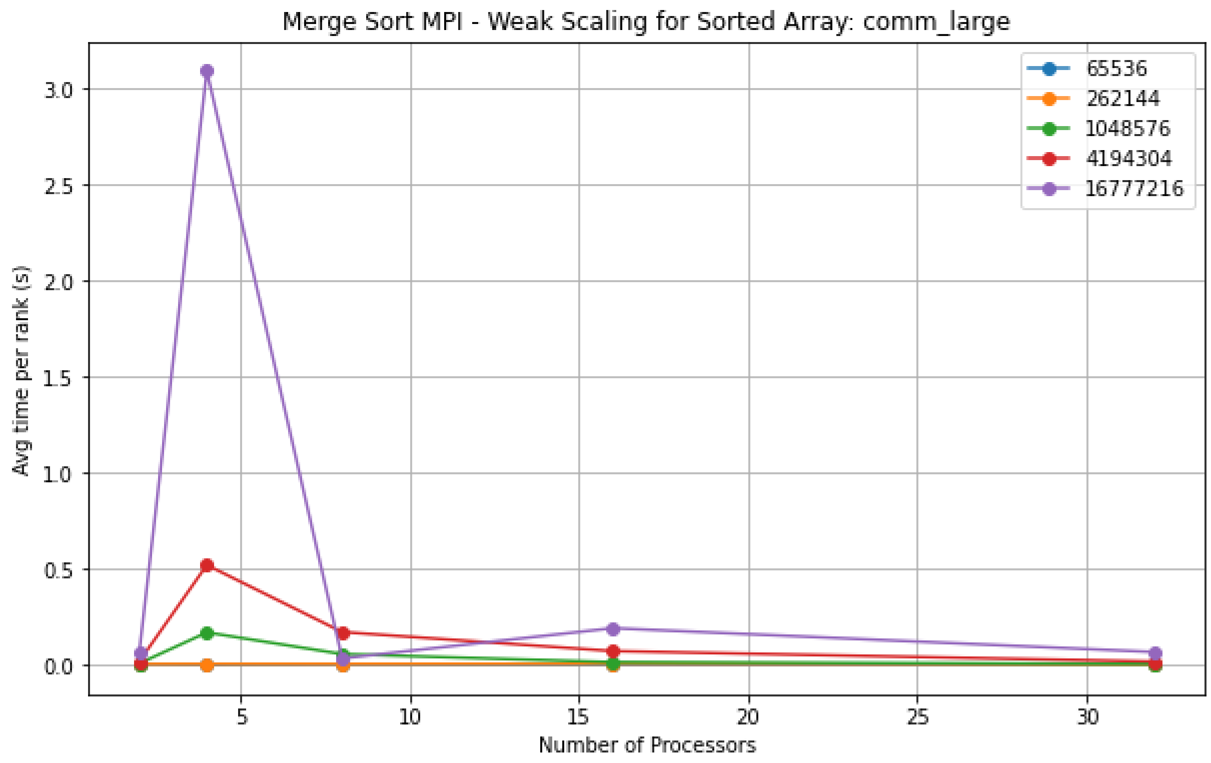
```
regions = [main, comm, comm_large, MPI_Gather, MPI_Scatter, comp, comp_large, comp_sma
names = ["main", "comm", "comm_large", "MPI_Gather", "MPI_Scatter", "comp", "comp_larg
```

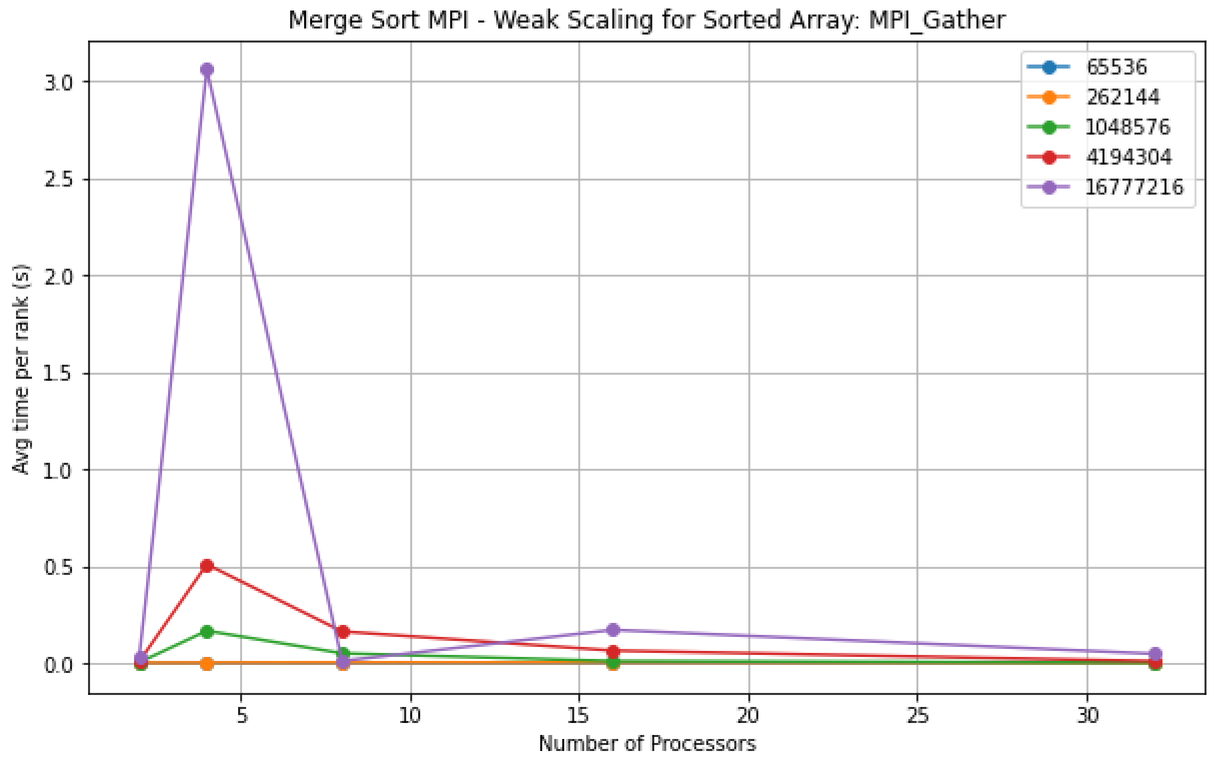
In [109…

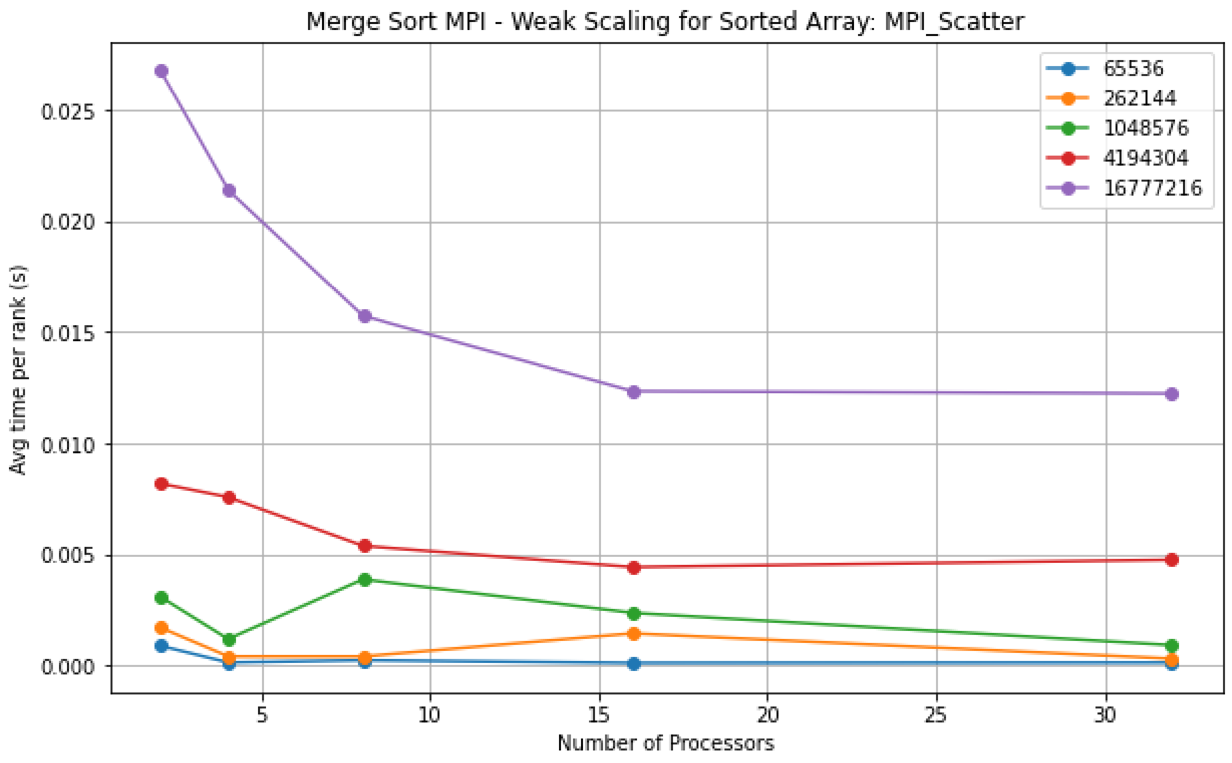
```
for region, name in zip(regions, names):
    plt.figure(figsize=(10, 6))  # Adjust the figure size if needed
    legend_labels = []
    for column in region.columns:
        first_index = column[0]  # Extract the first index
        legend_labels.append(first_index)
        plt.plot(region.index, region.xs(column, axis=1), marker='o', label=column)

    plt.xlabel('Number of Processors')
    plt.ylabel('Avg time per rank (s)')
    plt.title(f'Merge Sort MPI - Weak Scaling for Sorted Array: {name}')
    plt.legend(legend_labels)
    plt.grid(True)
    plt.show()
```

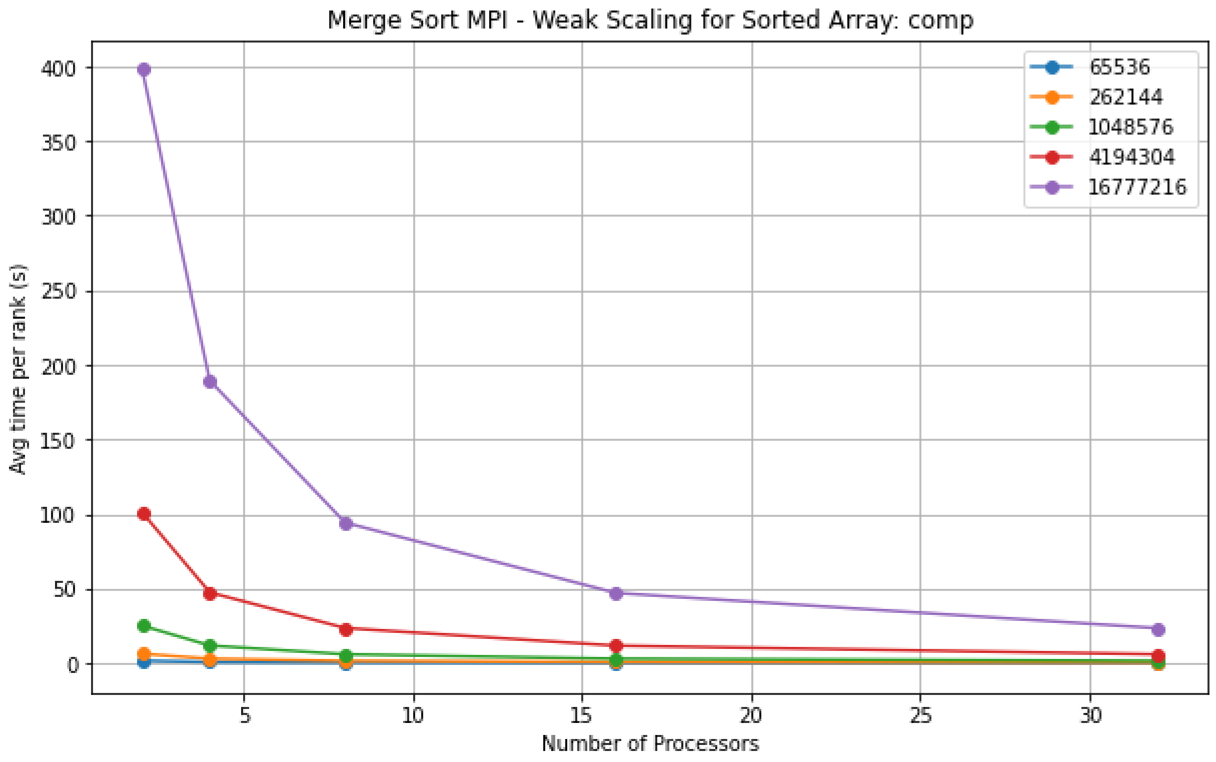

Merge Sort MPI - Weak Scaling for Sorted Array: main
Avg time per rank (s)
Number of Processors
0
100
200
300
400
500
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216

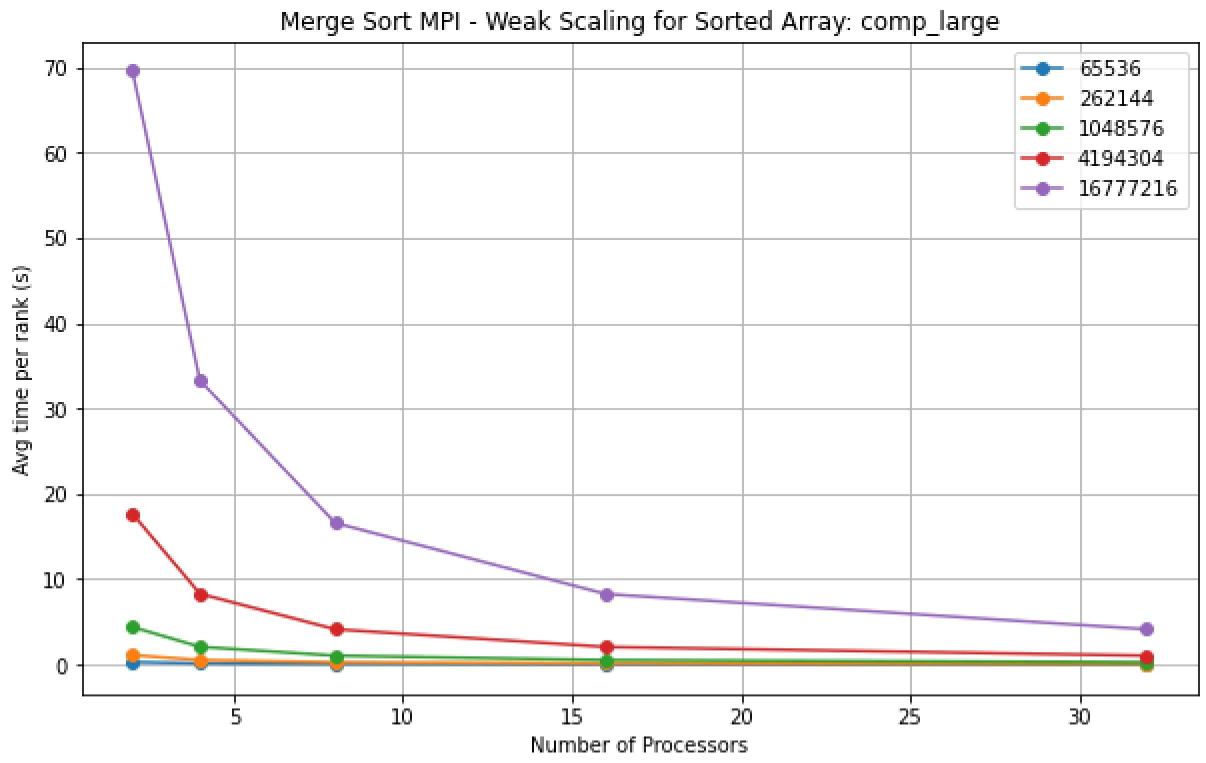

Merge Sort MPI - Weak Scaling for Sorted Array: comm
Avg time per rank (s)
Number of Processors
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216

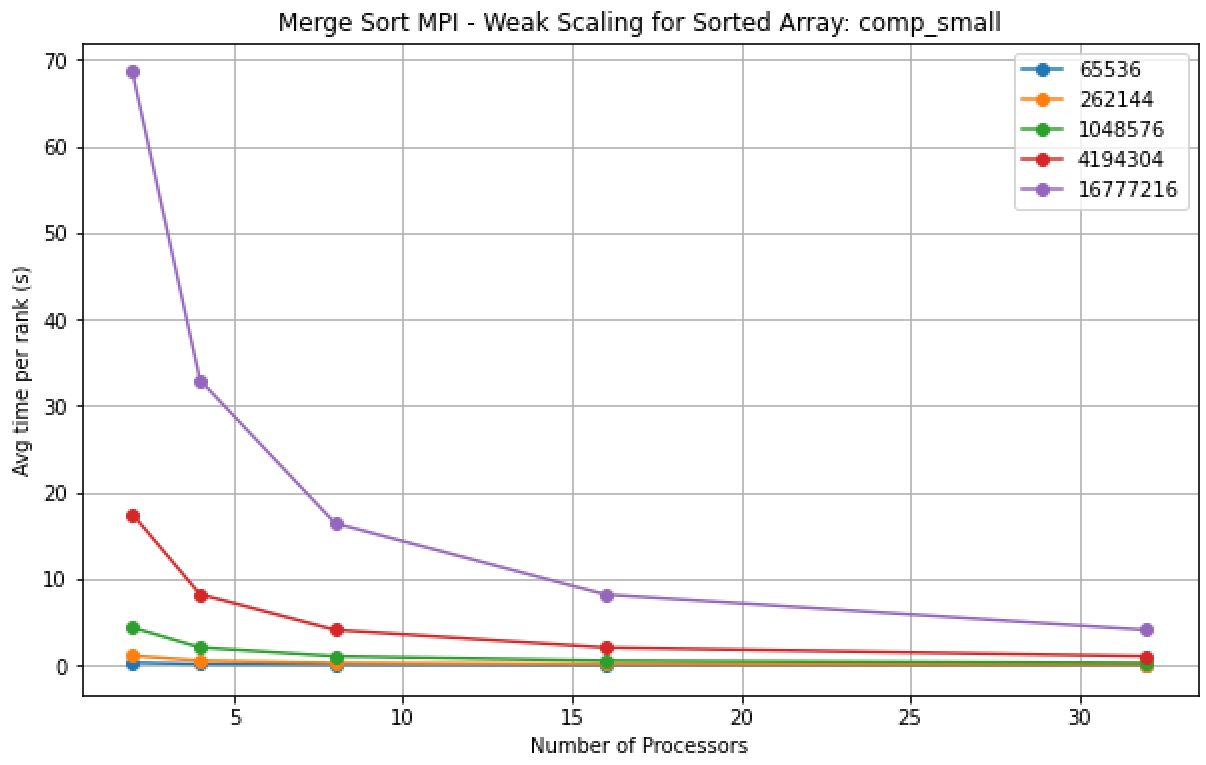

Merge Sort MPI - Weak Scaling for Sorted Array: comm_large
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30

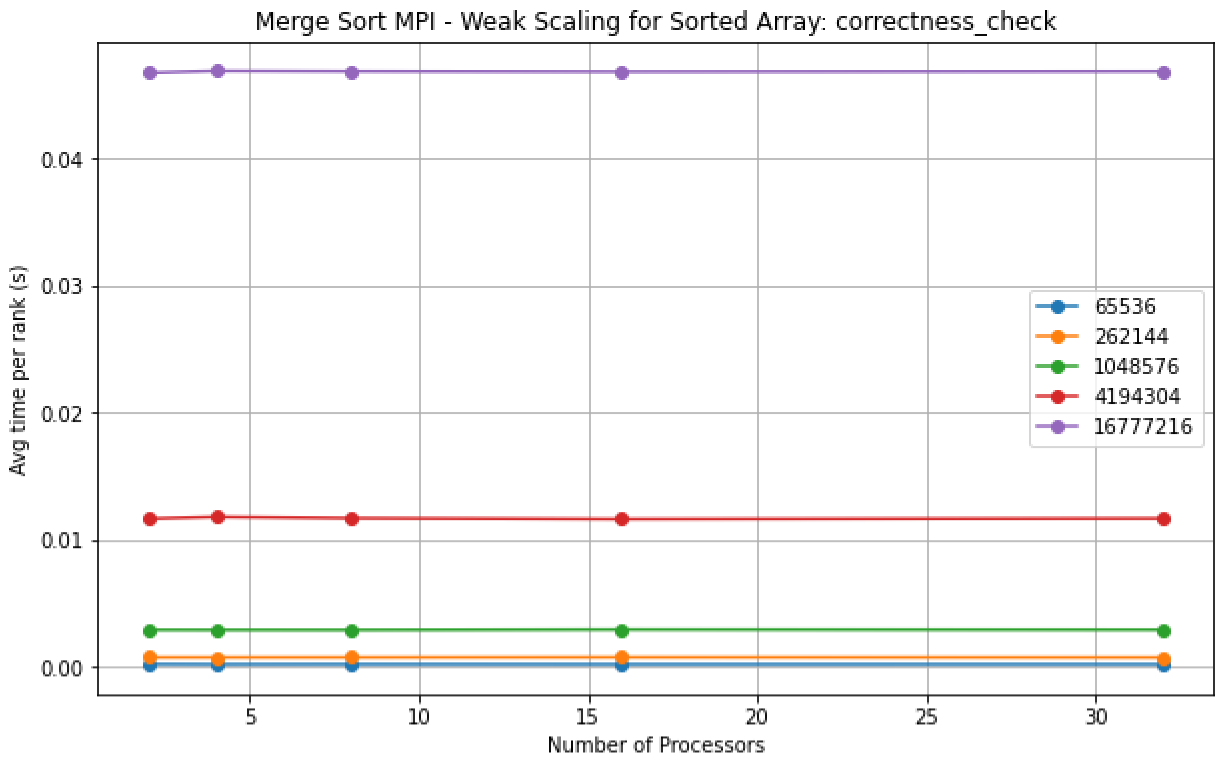

Merge Sort MPI - Weak Scaling for Sorted Array: MPI_Gather
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30

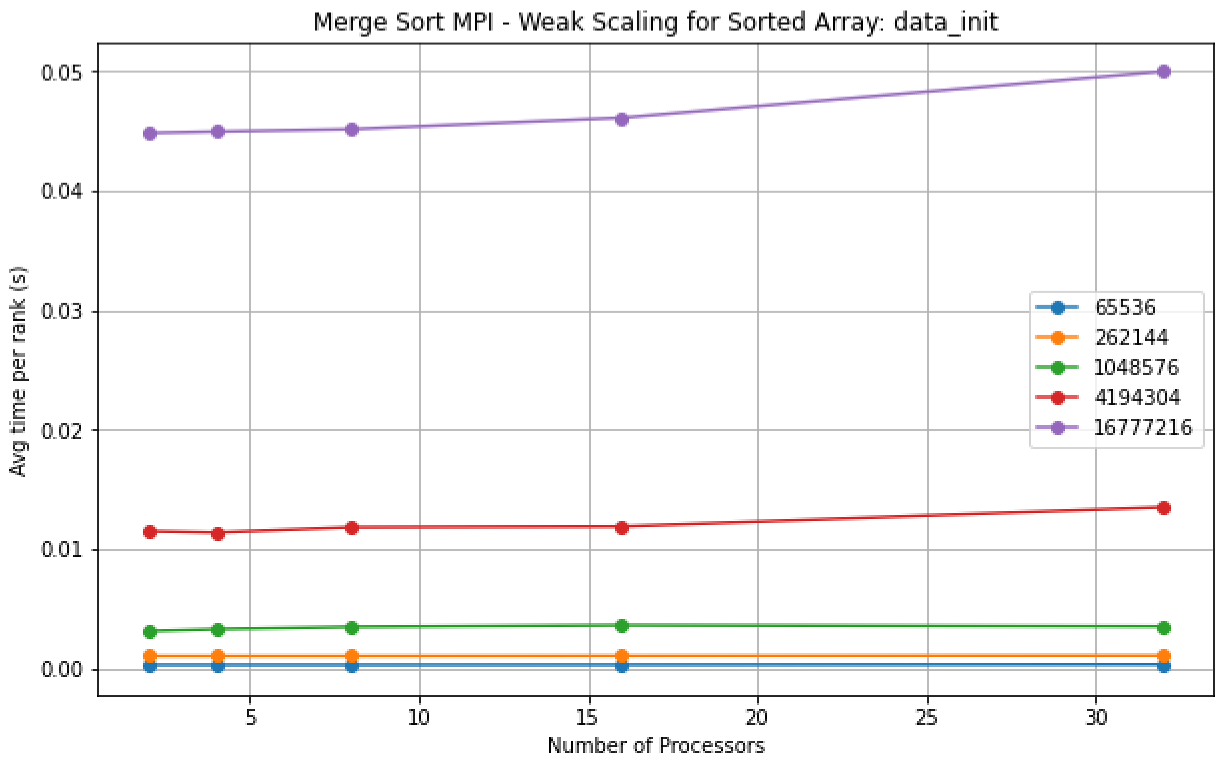

Merge Sort MPI - Weak Scaling for Sorted Array: MPI_Scatter
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0.000
0.005
0.010
0.015
0.020
0.025
5
10
15
20
25
30


Merge Sort MPI - Weak Scaling for Sorted Array: comp
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0
50
100
150
200
250
300
350
400
5
10
15
20
25
30


Merge Sort MPI - Weak Scaling for Sorted Array: comp_large
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0
10
20
30
40
50
60
70
5
10
15
20
25
30


Merge Sort MPI - Weak Scaling for Sorted Array: comp_small
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0
10
20
30
40
50
60
70
5
10
15
20
25
30





Reverse Sorted

In [110…
```
tkrev = th.Thicket.from_caliperreader(glob("cali_data_missingLast2ArraySizes/*-2.cali"
tkrev.dataframe = tkrev.dataframe.drop(["nid", "spot.channel", "Total time", "Min time

gbrev = tkrev.groupby("InputSize")

ctkrev = th.Thicket.concat_thickets(
    thickets=list(gbrev.values()),
    headers=list(gbrev.keys()),
    axis="columns",
    metadata_key="num_procs"
)
```

```
5  thickets created...
{65536: <thicket.thicket.Thicket object at 0x2b16bb2a5b20>, 262144: <thicket.thicket.
Thicket object at 0x2b16bb2bd490>, 1048576: <thicket.thicket.Thicket object at 0x2b16
bb15d3a0>, 4194304: <thicket.thicket.Thicket object at 0x2b16bb778f70>, 16777216: <th
icket.thicket.Thicket object at 0x2b16bb852040>}
```

In [111…

```
ctkrev.dataframe = ctkrev.dataframe.reset_index().drop(("node"), axis=1)
ctkrev.dataframe = ctkrev.dataframe.rename({("name", ""): "name", ("num_procs", ""): '

main = ctkrev.dataframe.loc["main"]
comm = ctkrev.dataframe.loc["comm"]
comm_large = ctkrev.dataframe.loc["comm_large"]
MPI_Gather = ctkrev.dataframe.loc["MPI_Gather"]
MPI_Scatter = ctkrev.dataframe.loc["MPI_Scatter"]
comp = ctkrev.dataframe.loc["comp"]
comp_large = ctkrev.dataframe.loc["comp_large"]
comp_small = ctkrev.dataframe.loc["comp_small"]
correctness_check = ctkrev.dataframe.loc["correctness_check"]
data_init = ctkrev.dataframe.loc["data_init"]
```

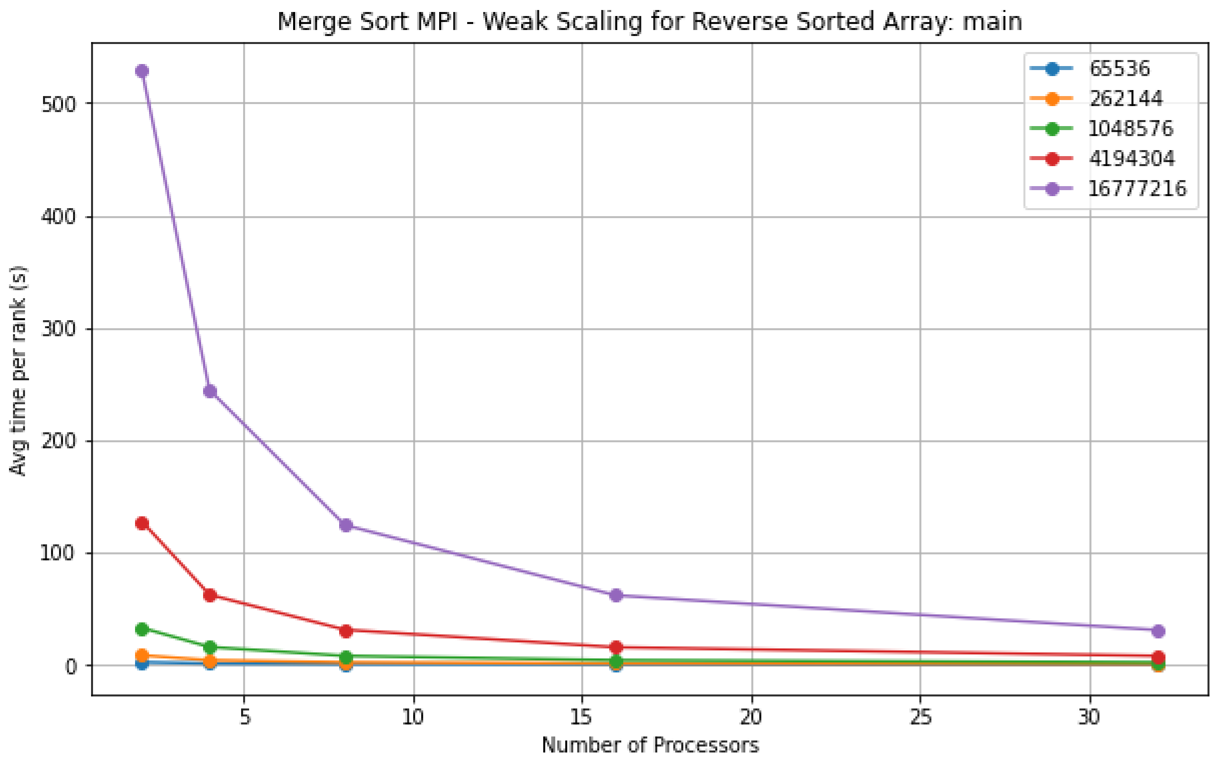
```
<ipython-input-111-51629b6c07fa>:1: PerformanceWarning: dropping on a non-lexsorted m
ulti-index without a level parameter may impact performance.
  ctkrev.dataframe = ctkrev.dataframe.reset_index().drop(("node"), axis=1)
```

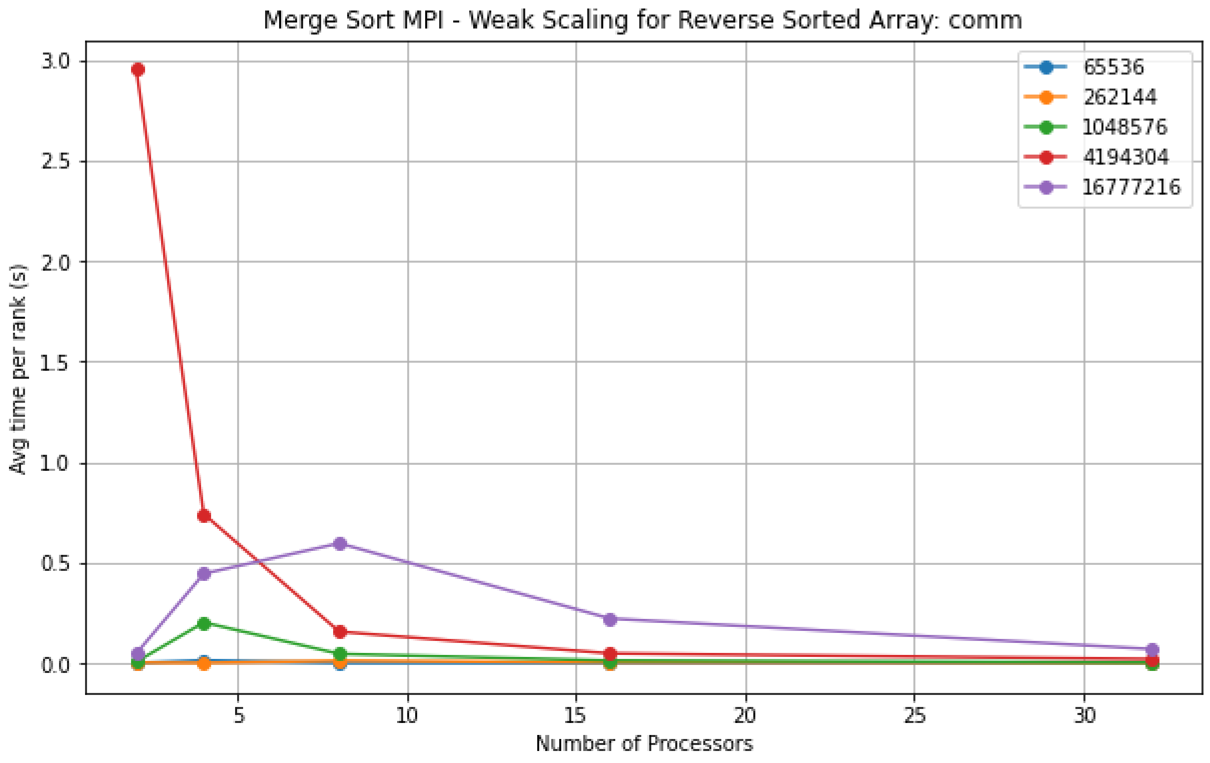
In [112…

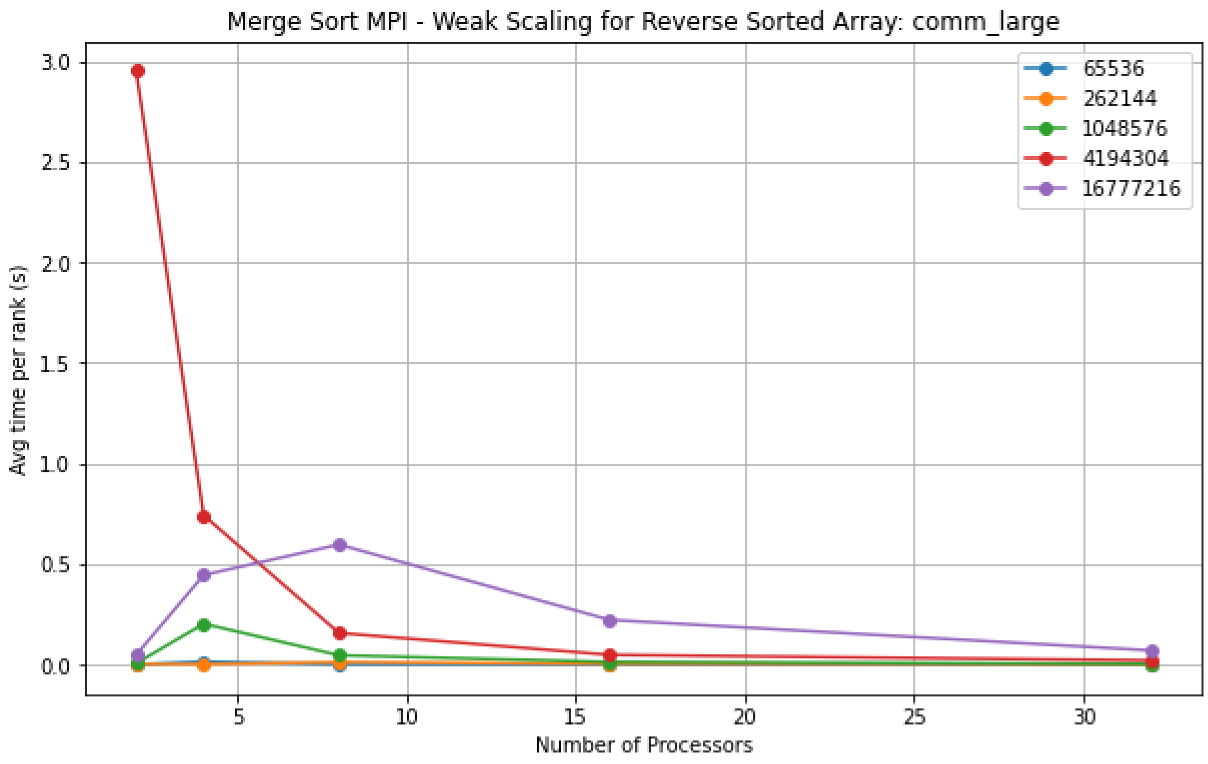
```
regions = [main, comm, comm_large, MPI_Gather, MPI_Scatter, comp, comp_large, comp_sma
names = ["main", "comm", "comm_large", "MPI_Gather", "MPI_Scatter", "comp", "comp_larg
```

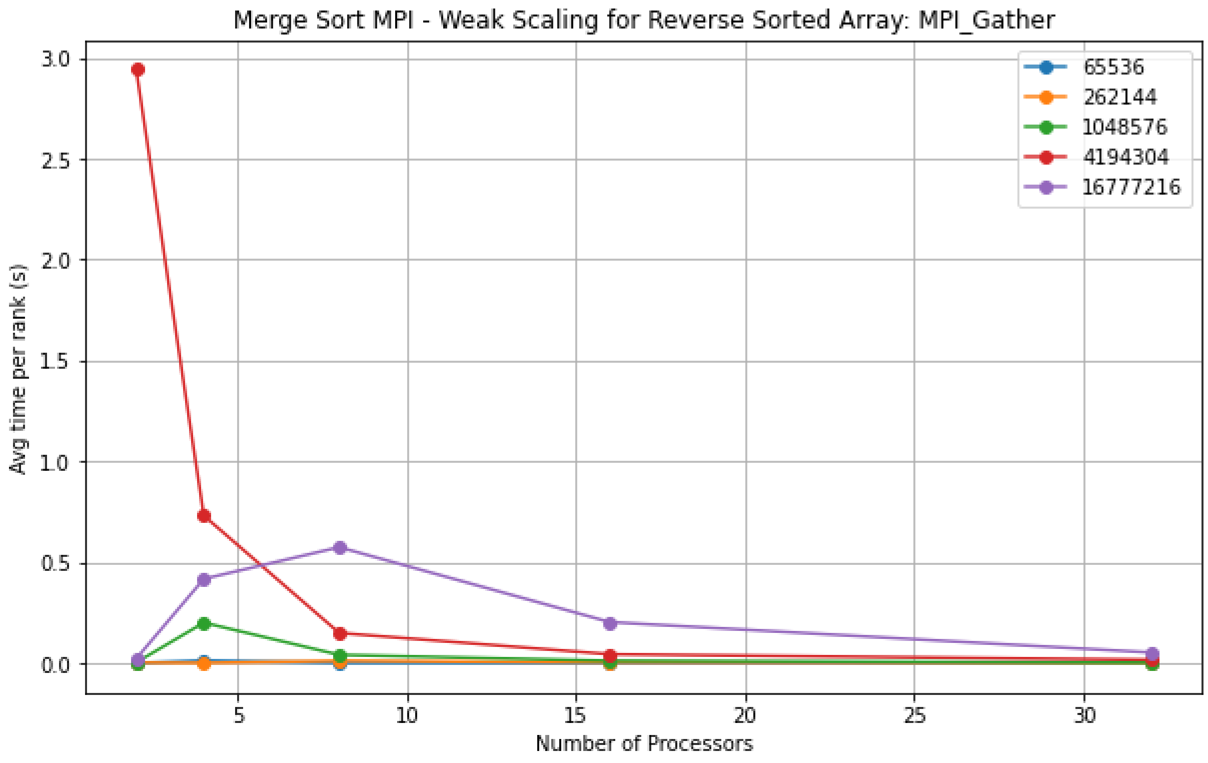
In [113…

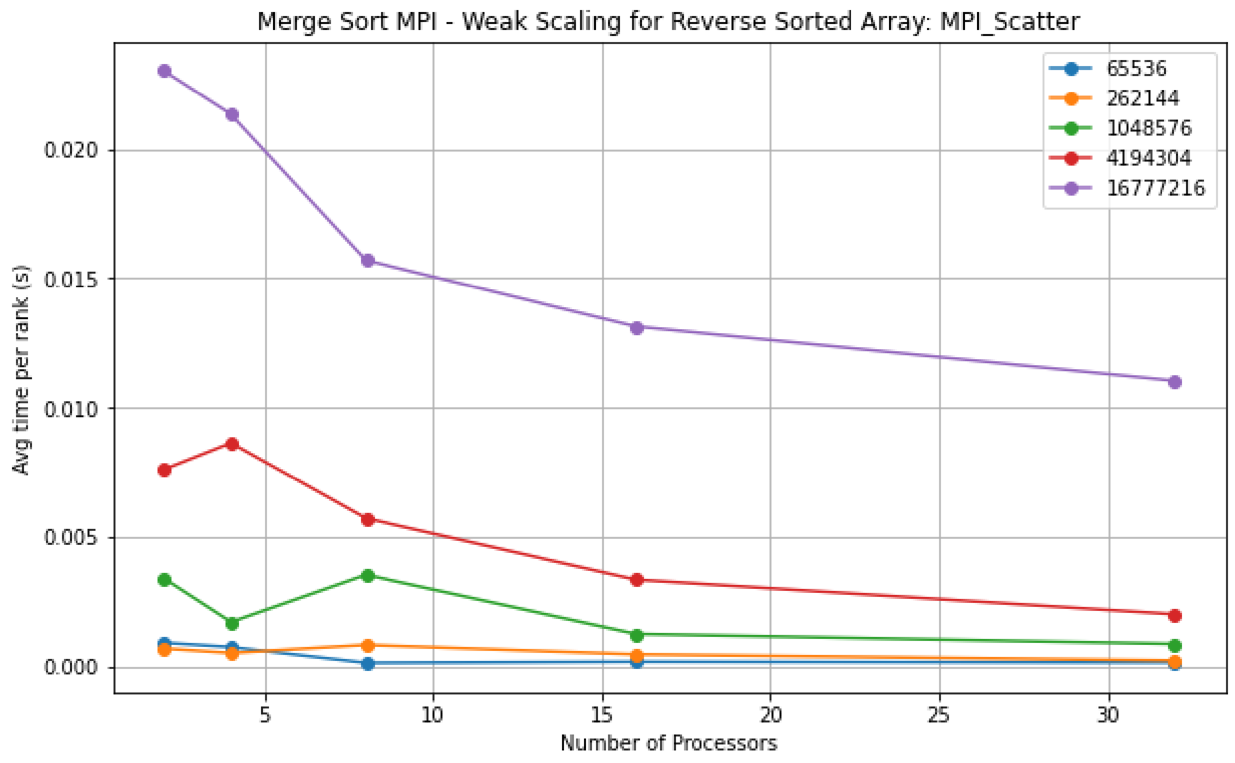
```
for region, name in zip(regions, names):
    plt.figure(figsize=(10, 6))  # Adjust the figure size if needed
    legend_labels = []
    for column in region.columns:
        first_index = column[0]  # Extract the first index
        legend_labels.append(first_index)
        plt.plot(region.index, region.xs(column, axis=1), marker='o', label=column)

    plt.xlabel('Number of Processors')
    plt.ylabel('Avg time per rank (s)')
    plt.title(f'Merge Sort MPI - Weak Scaling for Reverse Sorted Array: {name}')
    plt.legend(legend_labels)
    plt.grid(True)
    plt.show()
```

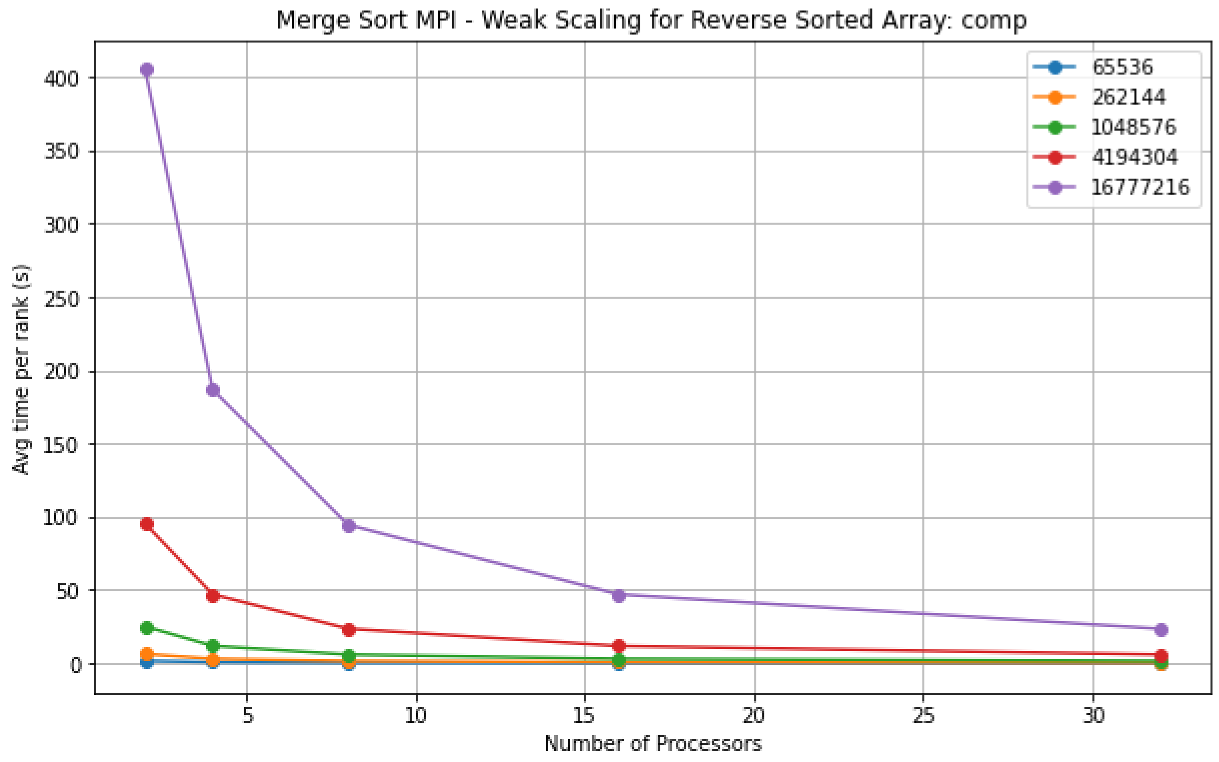

Merge Sort MPI - Weak Scaling for Reverse Sorted Array: main
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0
100
200
300
400
500
5
10
15
20
25
30

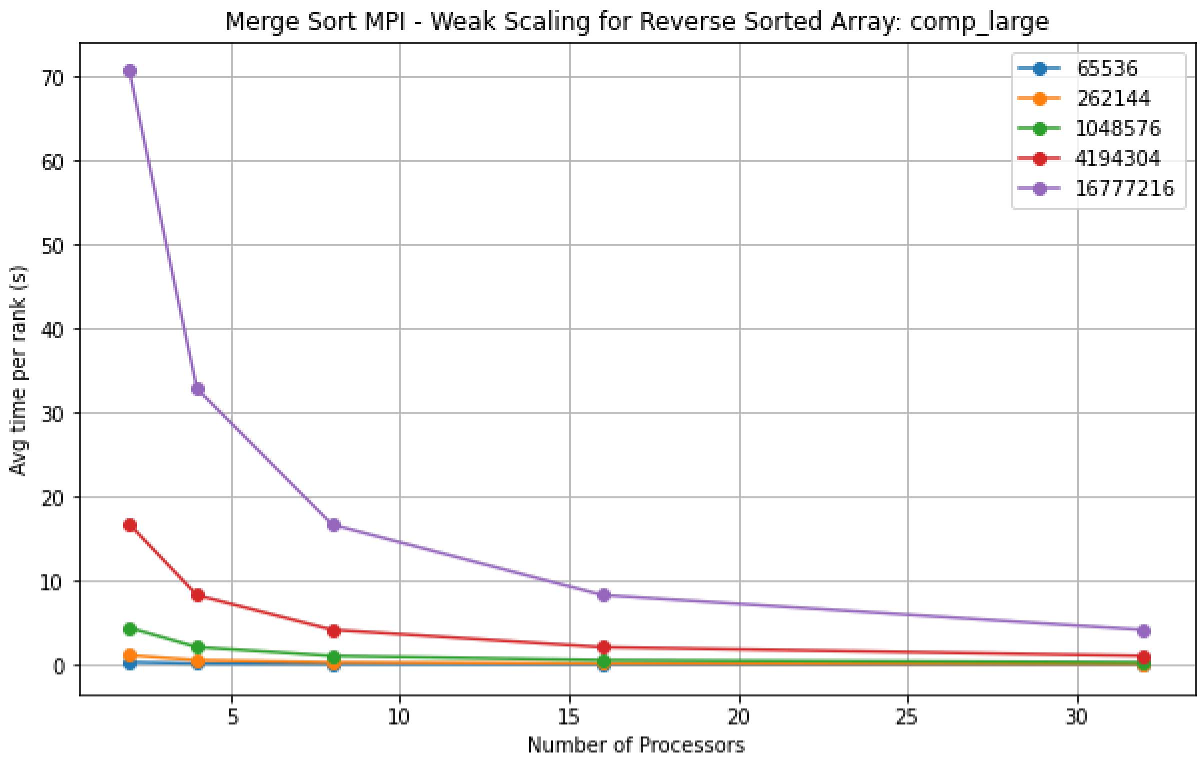

Merge Sort MPI - Weak Scaling for Reverse Sorted Array: comm
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30

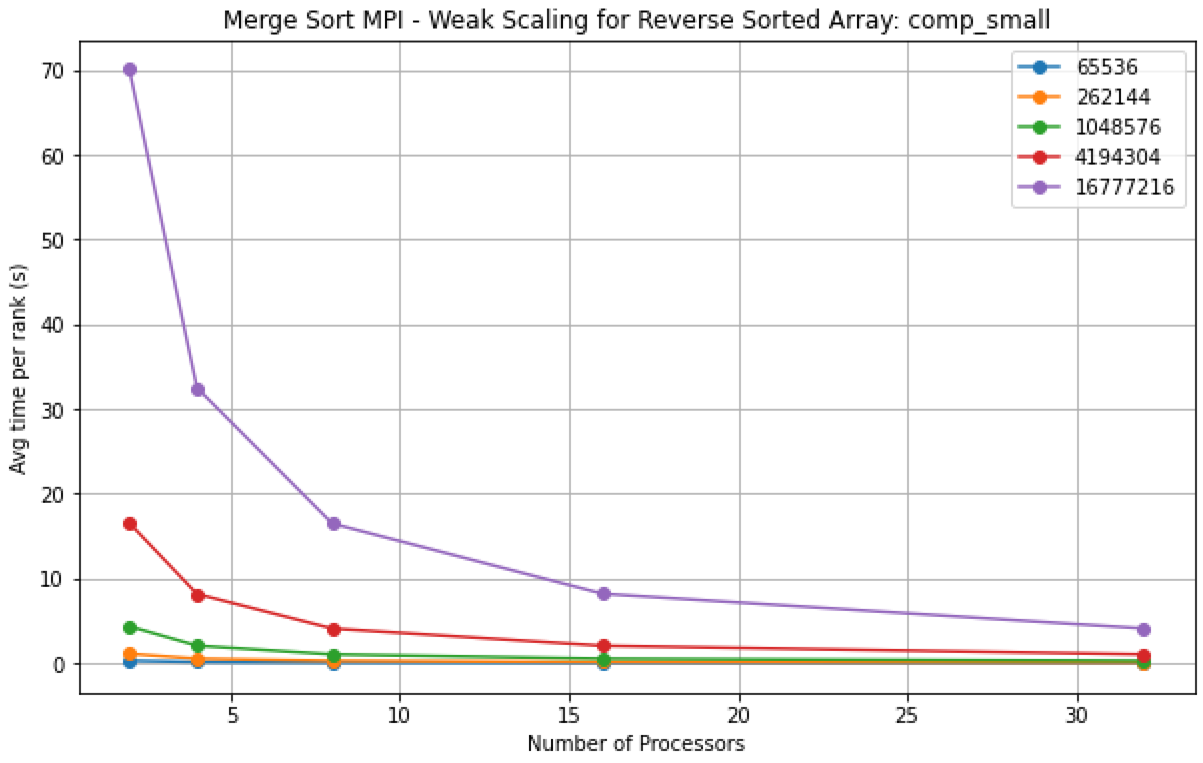

Merge Sort MPI - Weak Scaling for Reverse Sorted Array: comm_large
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30

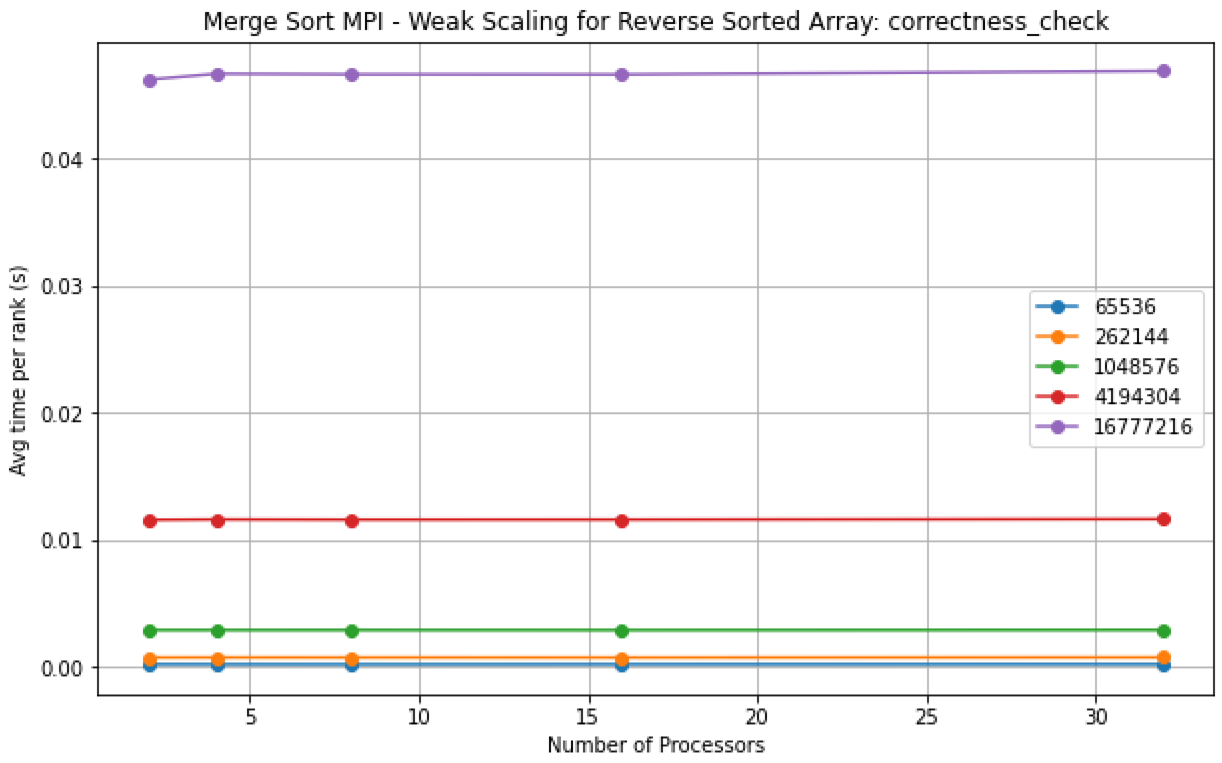

Merge Sort MPI - Weak Scaling for Reverse Sorted Array: MPI_Gather
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30

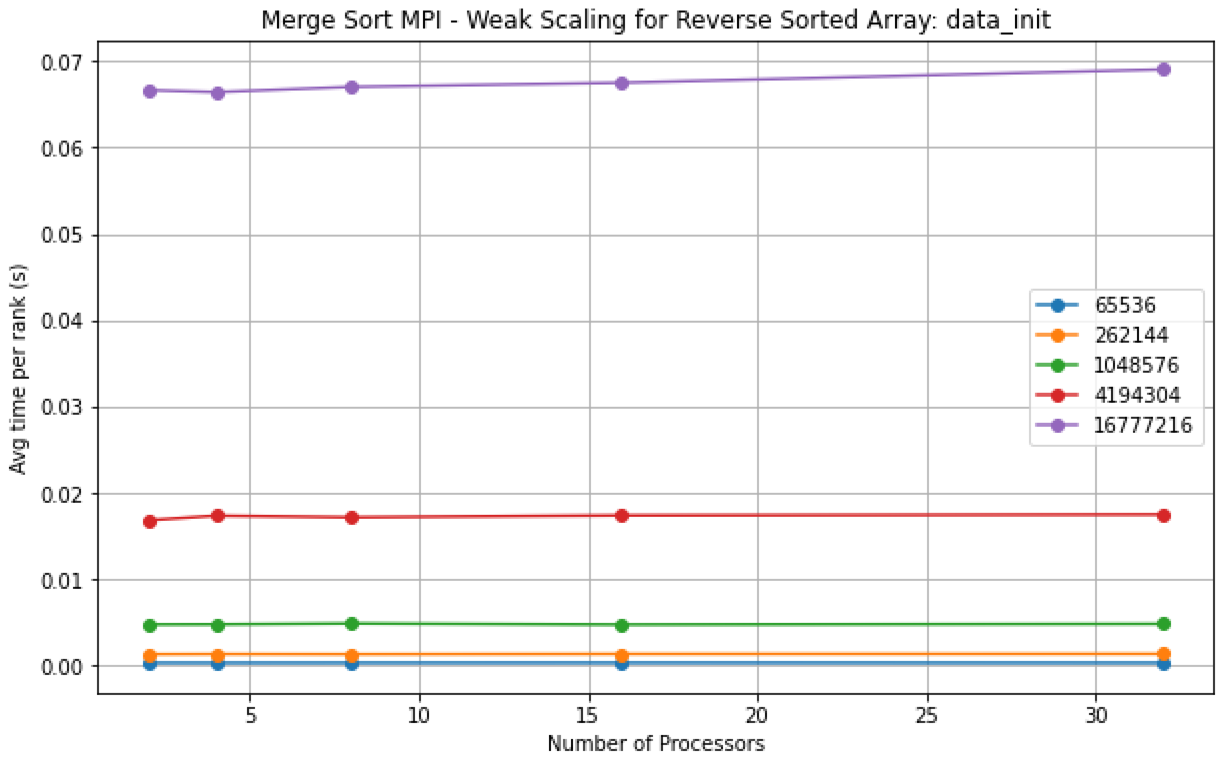

Merge Sort MPI - Weak Scaling for Reverse Sorted Array: MPI_Scatter
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0.000
0.005
0.010
0.015
0.020
5
10
15
20
25
30


Merge Sort MPI - Weak Scaling for Reverse Sorted Array: comp
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0
50
100
150
200
250
300
350
400
5
10
15
20
25
30


Merge Sort MPI - Weak Scaling for Reverse Sorted Array: comp_large
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0
10
20
30
40
50
60
70
5
10
15
20
25
30


Merge Sort MPI - Weak Scaling for Reverse Sorted Array: comp_small
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0
10
20
30
40
50
60
70
5
10
15
20
25
30





1% Perturbed

In [114…

```
tk1 = th.Thicket.from_caliperreader(glob("cali_data_missingLast2ArraySizes/*-2.cali"))
tk1.dataframe = tk1.dataframe.drop(["nid", "spot.channel", "Total time", "Min time/rar

gb1 = tk1.groupby("InputSize")

ctk1 = th.Thicket.concat_thickets(
    thickets=list(gb1.values()),
    headers=list(gb1.keys()),
    axis="columns",
    metadata_key="num_procs"
)
```

```
5  thickets created...
{65536: <thicket.thicket.Thicket object at 0x2b16957ed850>, 262144: <thicket.thicket.
Thicket object at 0x2b16bb17ac40>, 1048576: <thicket.thicket.Thicket object at 0x2b16
bb2d0a00>, 4194304: <thicket.thicket.Thicket object at 0x2b16ba0de940>, 16777216: <th
icket.thicket.Thicket object at 0x2b16bb221850>}
```

In [115…

```
ctk1.dataframe = ctk1.dataframe.reset_index().drop(("node"), axis=1)
ctk1.dataframe = ctk1.dataframe.rename({("name", ""): "name", ("num_procs", ""): "num_

main = ctk1.dataframe.loc["main"]
comm = ctk1.dataframe.loc["comm"]
comm_large = ctk1.dataframe.loc["comm_large"]
MPI_Gather = ctk1.dataframe.loc["MPI_Gather"]
MPI_Scatter = ctk1.dataframe.loc["MPI_Scatter"]
comp = ctk1.dataframe.loc["comp"]
comp_large = ctk1.dataframe.loc["comp_large"]
comp_small = ctk1.dataframe.loc["comp_small"]
correctness_check = ctk1.dataframe.loc["correctness_check"]
data_init = ctk1.dataframe.loc["data_init"]
```

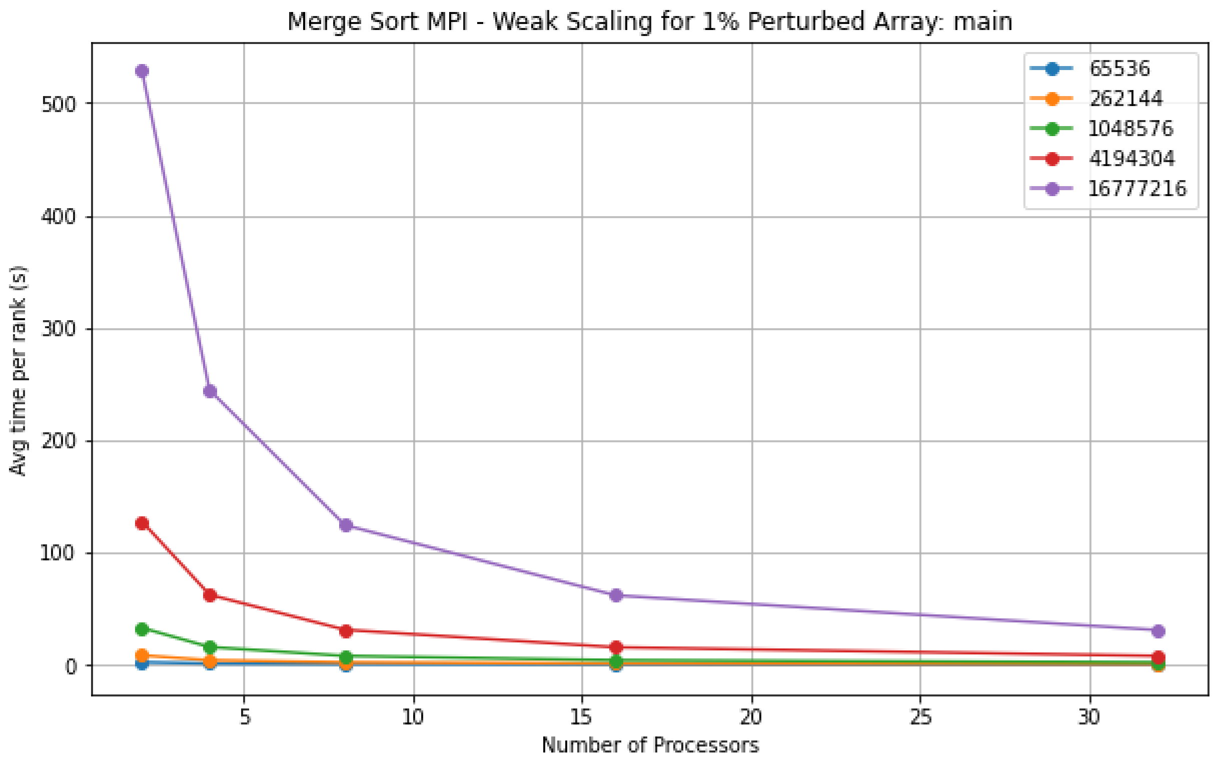
```
<ipython-input-115-01dc930deb89>:1: PerformanceWarning: dropping on a non-lexsorted m
ulti-index without a level parameter may impact performance.
  ctk1.dataframe = ctk1.dataframe.reset_index().drop(("node"), axis=1)
```

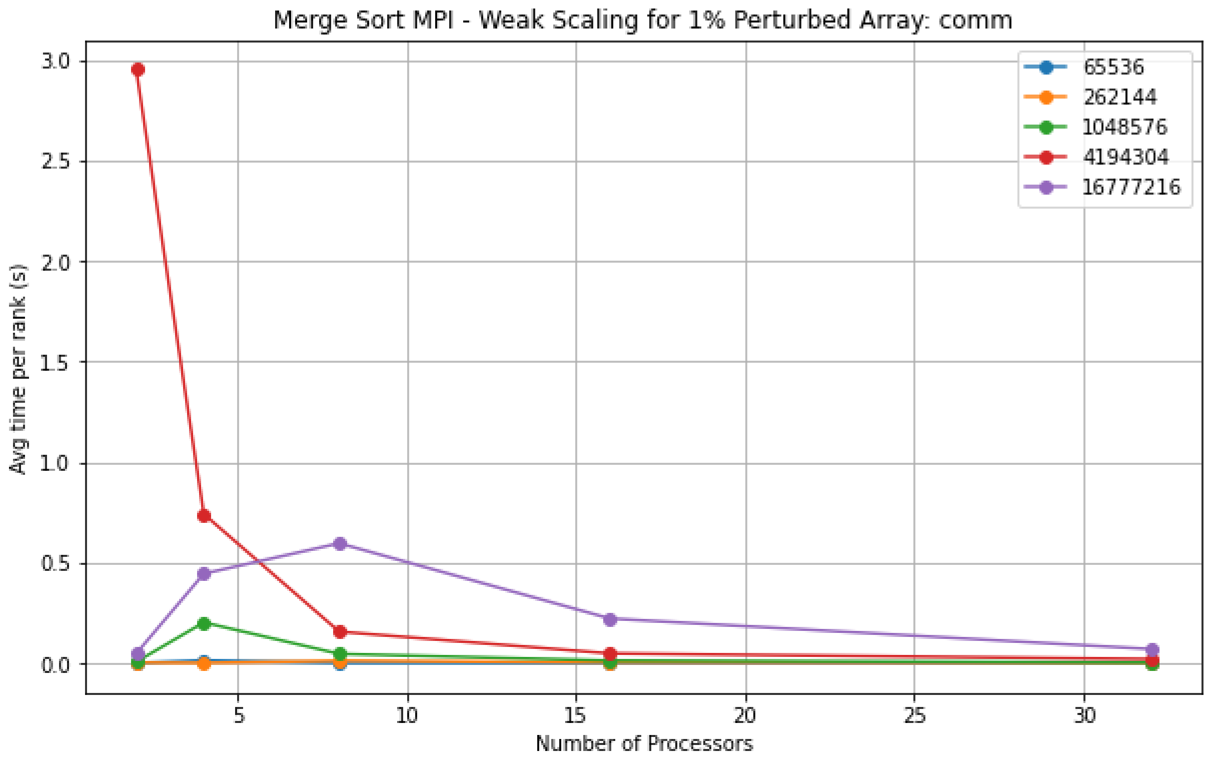
In [116…

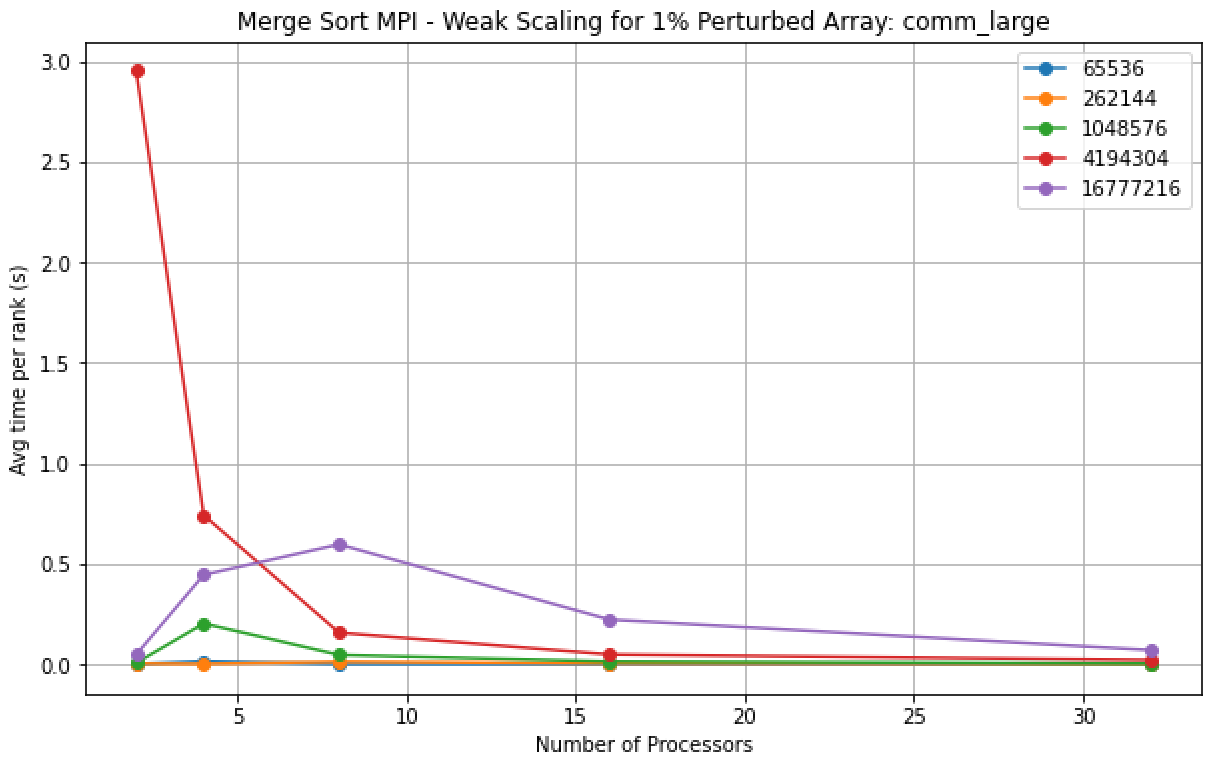
```
regions = [main, comm, comm_large, MPI_Gather, MPI_Scatter, comp, comp_large, comp_sma
names = ["main", "comm", "comm_large", "MPI_Gather", "MPI_Scatter", "comp", "comp_larg
```

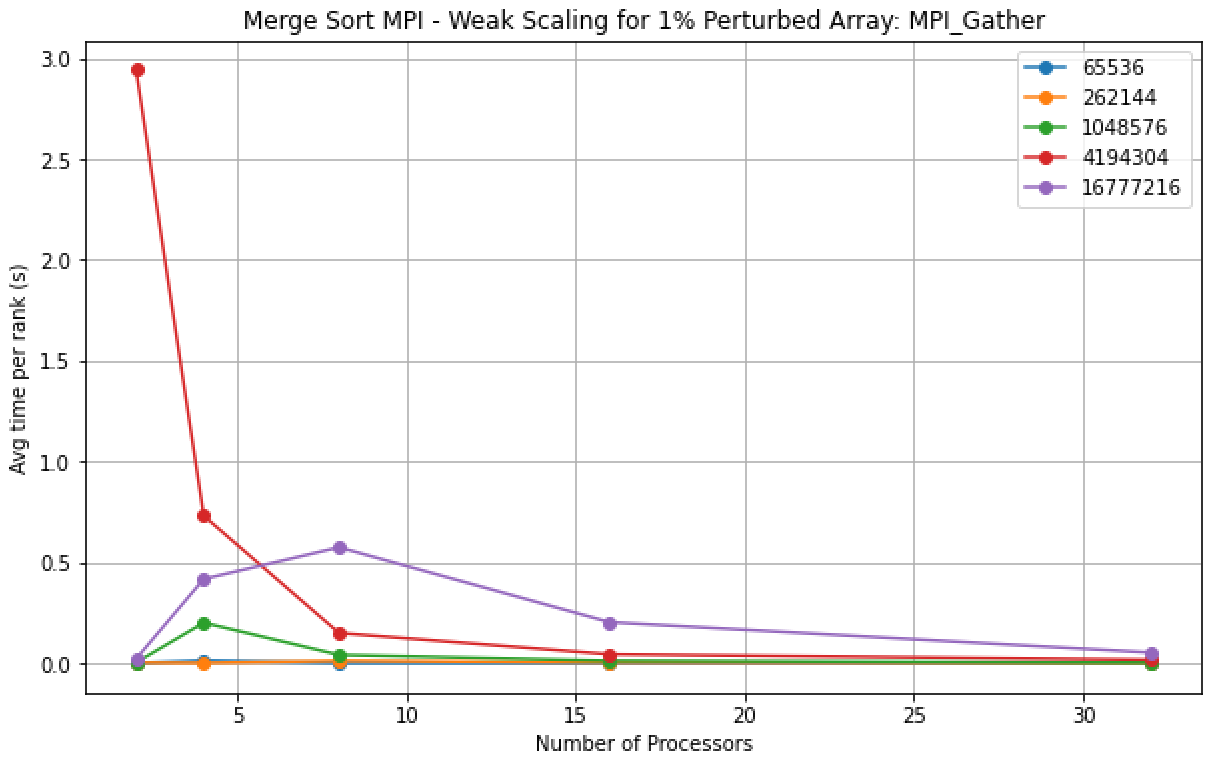
In [117…

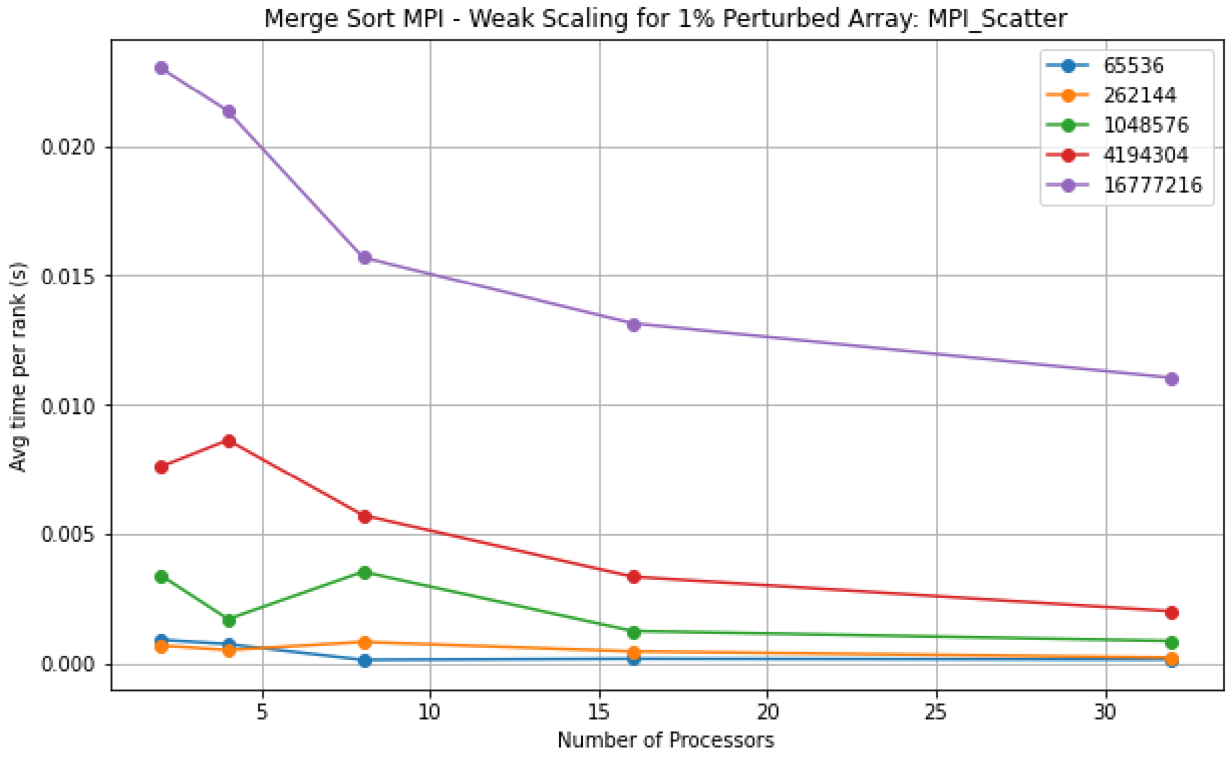
```
for region, name in zip(regions, names):
    plt.figure(figsize=(10, 6))  # Adjust the figure size if needed
    legend_labels = []
    for column in region.columns:
        first_index = column[0]  # Extract the first index
        legend_labels.append(first_index)
        plt.plot(region.index, region.xs(column, axis=1), marker='o', label=column)

    plt.xlabel('Number of Processors')
    plt.ylabel('Avg time per rank (s)')
    plt.title(f'Merge Sort MPI - Weak Scaling for 1% Perturbed Array: {name}')
    plt.legend(legend_labels)
    plt.grid(True)
    plt.show()
```

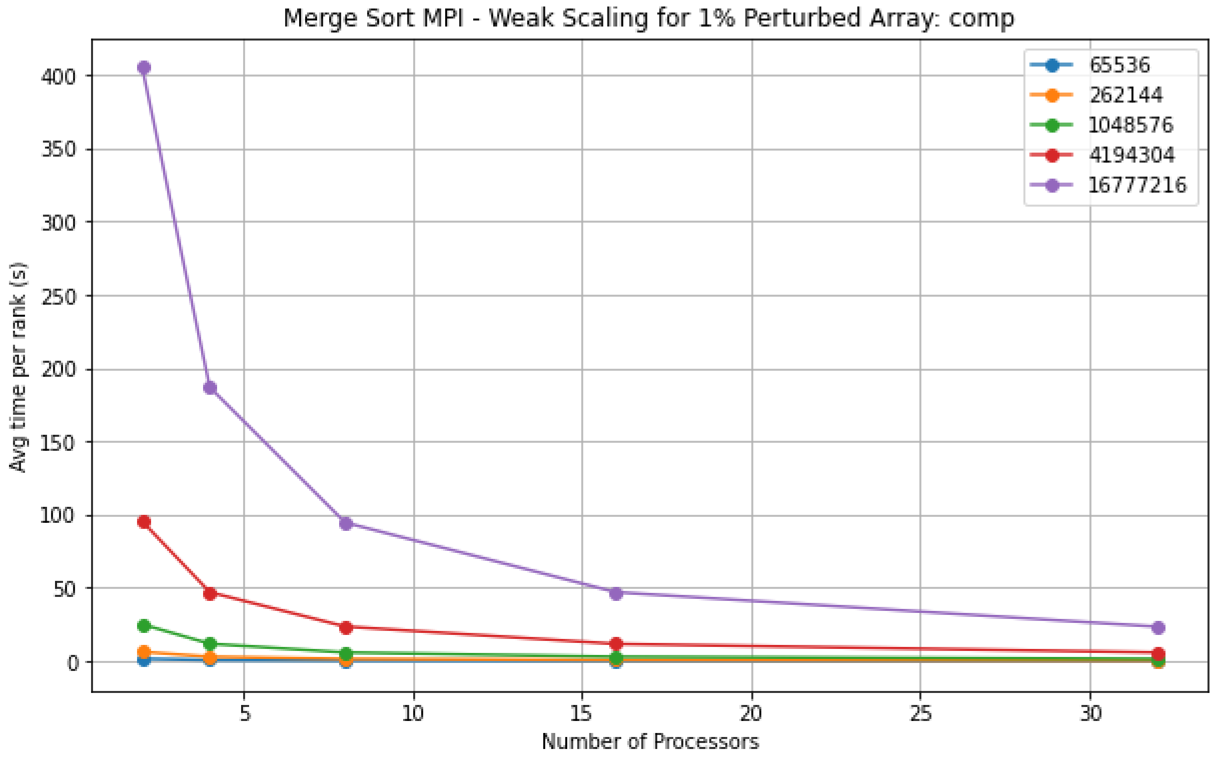

Merge Sort MPI - Weak Scaling for 1% Perturbed Array: main
Avg time per rank (s)
Number of Processors
0
100
200
300
400
500
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216

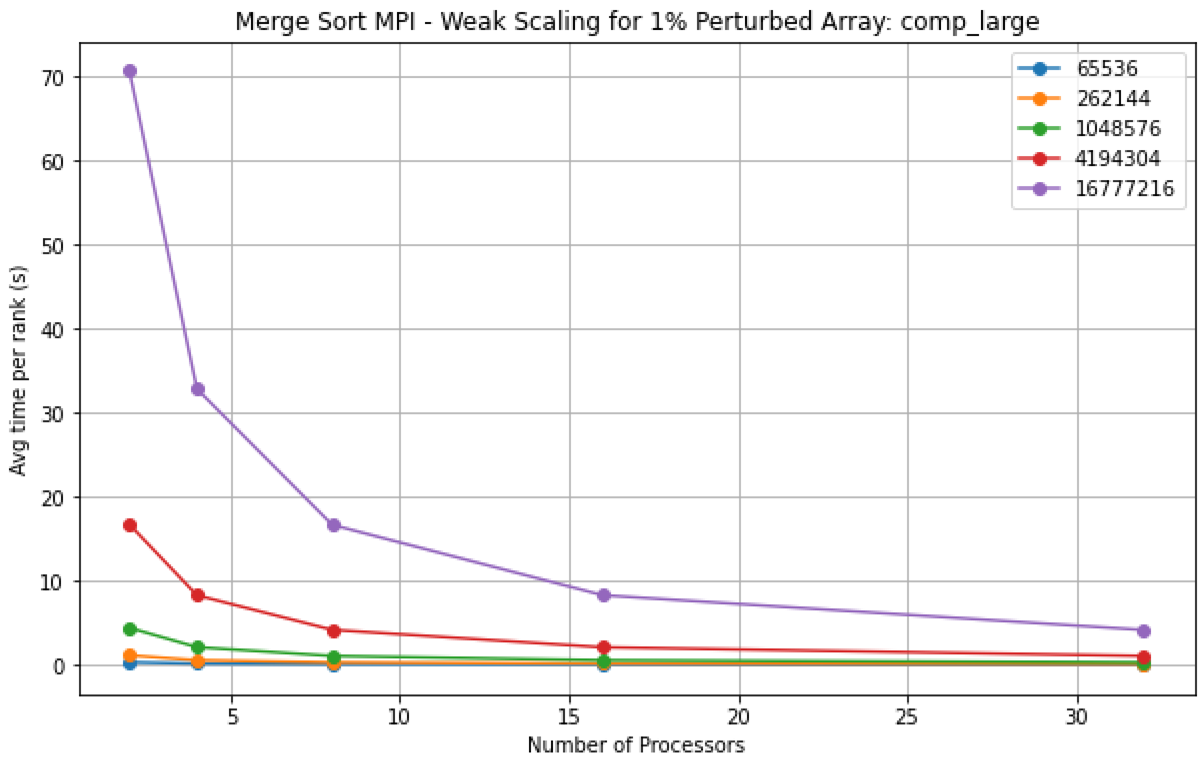

Merge Sort MPI - Weak Scaling for 1% Perturbed Array: comm
Avg time per rank (s)
Number of Processors
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216

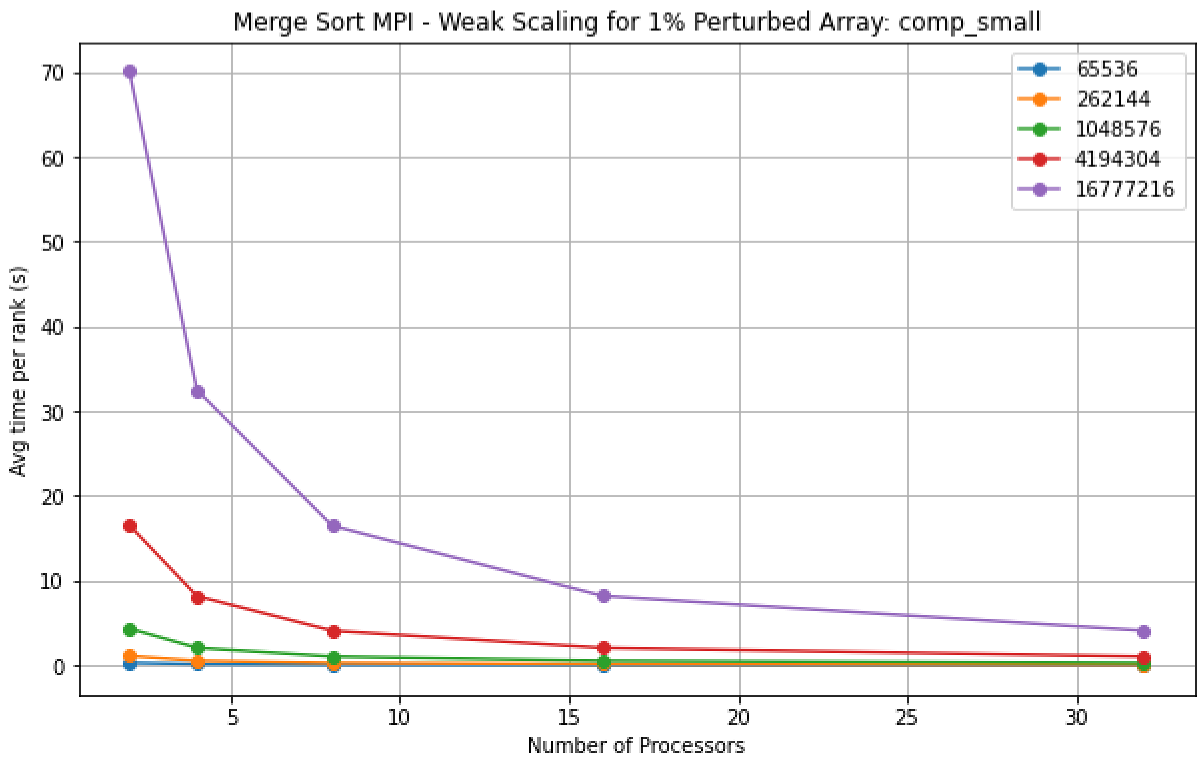

Merge Sort MPI - Weak Scaling for 1% Perturbed Array: comm_large
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30

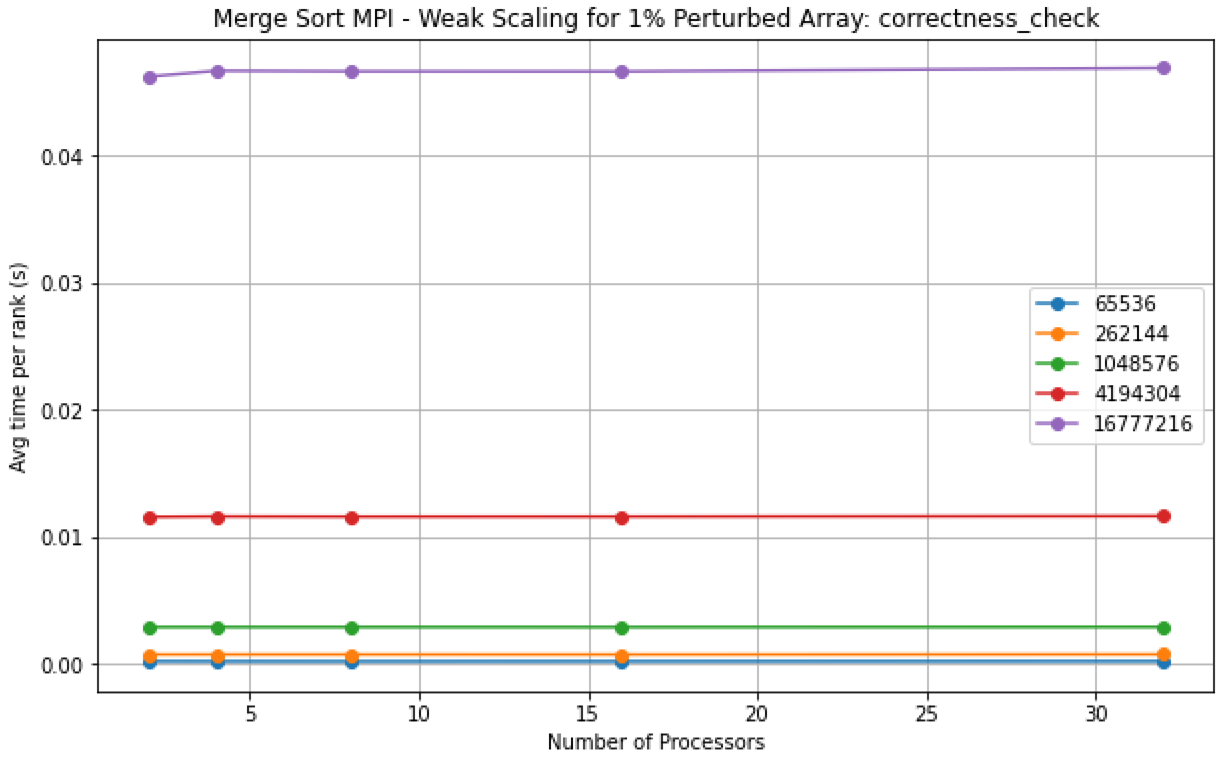

Merge Sort MPI - Weak Scaling for 1% Perturbed Array: MPI_Gather
Avg time per rank (s)
Number of Processors
65536
262144
1048576
4194304
16777216
0.0
0.5
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30

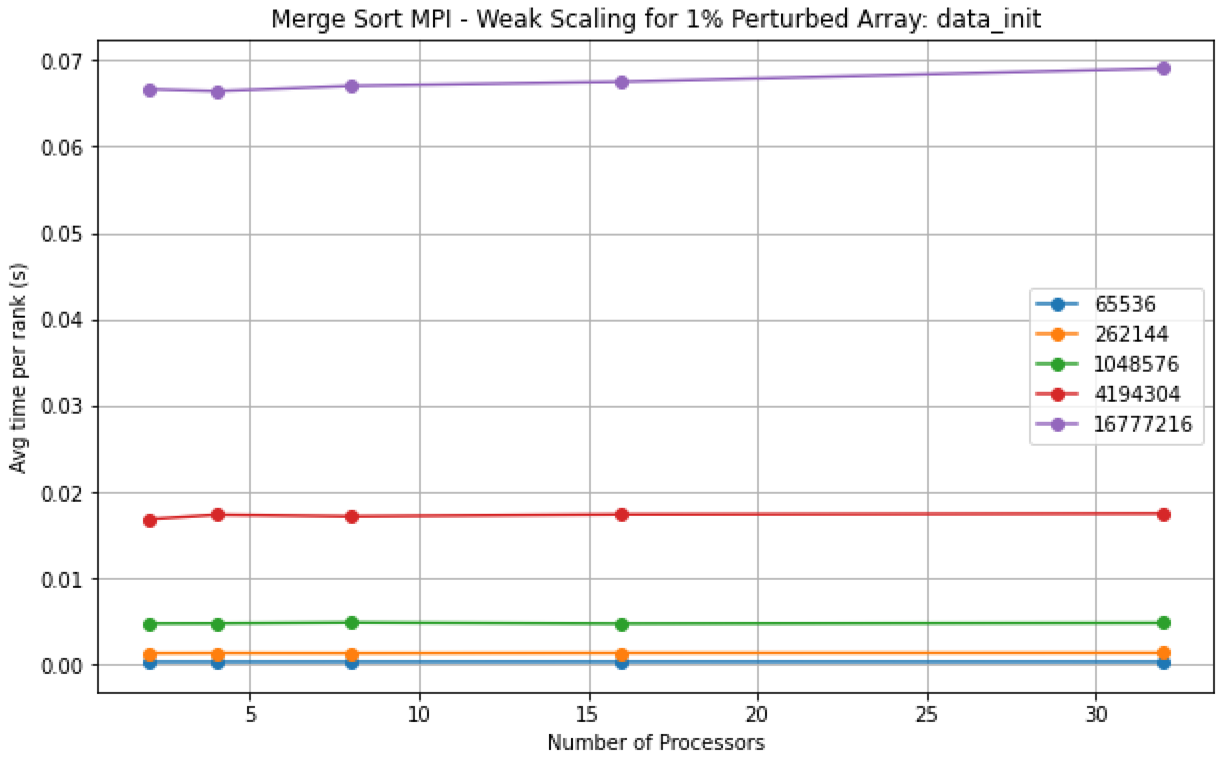

Merge Sort MPI - Weak Scaling for 1% Perturbed Array: MPI_Scatter
Avg time per rank (s)
Number of Processors
0.000
0.005
0.010
0.015
0.020
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216


Merge Sort MPI - Weak Scaling for 1% Perturbed Array: comp
Avg time per rank (s)
Number of Processors
0
50
100
150
200
250
300
350
400
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216


Merge Sort MPI - Weak Scaling for 1% Perturbed Array: comp_large
Avg time per rank (s)
Number of Processors
0
10
20
30
40
50
60
70
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216


Merge Sort MPI - Weak Scaling for 1% Perturbed Array: comp_small
Avg time per rank (s)
Number of Processors
0
10
20
30
40
50
60
70
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216


Merge Sort MPI - Weak Scaling for 1% Perturbed Array: correctness_check
Avg time per rank (s)
Number of Processors
0.00
0.01
0.02
0.03
0.04
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216


Merge Sort MPI - Weak Scaling for 1% Perturbed Array: data_init
Avg time per rank (s)
Number of Processors
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
5
10
15
20
25
30
65536
262144
1048576
4194304
16777216

```
In [ ]:
```