The communication among the processors is done by making a communication system among all the senders and receivers. The MPI are given ranks to know what processor they are running and will send messages from a central processor 0. It will receive the message through all the other processors. After all the processing is done, then MPI will gather the results together and combine results to get the final output.

For 1024 problem size
np=1
Time: 0.0331161 s
Perf: 64.8471 GFlops
np=2
Time: 0.0215535 s
Perf: 99.635 GFlops
np=4
Time: 0.0277901 s
Perf: 77.2752 GFlops
np=8
Time: 0.0391152 s
Perf: 54.9015 GFlops
np=16
Time: 1.04513 s
Perf: 2.05476 GFlops
np=32
Time: 7.49758 s
Perf: 0.286424 GFlops

For 2048
np=1
Time: 0.252538 s
Perf: 68.0288 GFlops
np=2
Time: 0.154005 s
Perf: 111.554 GFlops
np=4
Time: 0.137389 s

Perf: 125.046 GFlops
np=8
Time: 0.170552 s
Perf: 100.731 GFlops
np=16
Time: 5.39351 s
Perf: 3.18529 GFlops
np=32
Time: 39.4372 s
Perf: 0.435626 GFlops

For 4096
np=1
Time: 2.26157 s
Perf: 60.7716 GFlops
np=2
Time: 1.24368 s
Perf: 110.51 GFlops
np=4
Time: 0.76862 s
Perf: 178.813 GFlops
np=8
Time: 1.20864 s
Perf: 113.714 GFlops
np=16
Time: 15.0518 s
Perf: 9.13107 GFlops
np=32
Time: 110.572 s
Perf: 1.24298 GFlops

For the 1024 small problem size, it looks like it peaked np=2. The processors added after that decreased the number of GFlops, but at np=1 it was actually fewer GFlops than np=2.

Now for the 2048 problem size and the 4096 problem size, it linearly increased until np=4 for where it peaked, and then np=8 and greater for the number of processes it dropped dramatically. 4096 had the greatest throughput overall because there was more of the problem to take advantage of its optimizations and parallelizations.

The MPI implementation was focused on how to use the MPI commands of communication and taking advantage of message processing through send, receiving, broadcasting, scattering, and gathering. Also used a buffer similar to the register of the OpenMP implementation. The OpenMP implementation was more focused on how to make it optimal through optimizations of loop unrolling, registers, and block size for the blocked implementation. Although MPI for 4096 increased throughput as the number of threads increased, it eventually dipped after np=8. However, in OpenMP for np=8, it still increased throughput. There were still more optimizations that could be made with a greater number of threads/processes for OpenMP. In terms of overall performance between the two, the implementation for MPI turned out to be greater in terms of throughput. At least for np=4, I think this is because for those threads, MPI has has less messages to send and receive, but still enough threads to perform parallelization.