For the following configurations:
1. 1 node, 1 GPU, 16Kx16K world size each MPI rank, 128 iterations with 1024 CUDA thread block size and pattern 3.
Execution time for rank 0 is 19.927015 s, using MPI_Wtime command.
Since a world of 16K * 16K world size that runs for 128 iterations will have $3.43597*10^{10}$ cell updates, the "cell updates per second" is $1.7243*10^{9}$.

```
++ _ModuleTable_Sz_=3
++ export _ModuleTable_Sz_
++ : -s sh
+ eval
+ mpirun -hostfile /tmp/hosts.9
MPI Wtime is 19.927015
+ rm /tmp/hosts.97503
~
~
```

2. 1 node, 2 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 1024 CUDA thread block size and pattern 3.
Execution time for rank 0 is 20.004488 s, using MPI_Wtime command.
Since a world of 16K * 16K world size that runs for 128 iterations will have $3.43597*10^{10}$ cell updates, the "cell updates per second" is $1.7176*10^{9}$.

```
++ _ModuleTable_Sz_=3
++ export _ModuleTable_Sz_
++ : -s sh
+ eval
+ mpirun -hostfile /tmp/hosts.
MPI Wtime is 20.004488
+ rm /tmp/hosts.97505
~
~
```

3. 1 node, 3 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 1024 CUDA thread block size and pattern 3.
Execution time rank 0 is 20.112175 s, using MPI_Wtime command.
Since a world of 16K * 16K world size that runs for 128 iterations will have $3.43597*10^{10}$ cell updates, the "cell updates per second" is $1.7084*10^{9}$.

```
++ _ModuleTable_Sz_=3
++ export _ModuleTable_Sz_
++ : -s sh
+ eval
+ mpirun -hostfile /tmp/hosts
MPI Wtime is 20.112175
+ rm /tmp/hosts.97506
~
~
```

4. 1 node, 4 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 1024 CUDA thread block size and pattern 3.
Execution time for rank 0 is 20.092383 s, using MPI_Wtime command.
Since a world of 16K * 16K world size that runs for 128 iterations will have $3.43597*10^{10}$ cell updates, the "cell updates per second" is $1.7101*10^{9}$.

```
++ _ModuleTable_Sz_=3
++ export _ModuleTable_Sz_
++ : -s sh
+ eval
+ mpirun -hostfile /tmp/hosts.97510
MPI Wtime is 20.092383
+ rm /tmp/hosts.97510
~
~
```

5. 1 node, 5 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 1024 CUDA thread block size and pattern 3.
Execution time for rank 0 is 20.764897 s, using MPI_Wtime command.
Since a world of 16K * 16K world size that runs for 128 iterations will have $3.43597*10^{10}$ cell updates, the "cell updates per second" is $1.6547*10^{9}$.

```
++ _ModuleTable_Sz_=3
++ export _ModuleTable_Sz_
++ : -s sh
+ eval
+ mpirun -hostfile /tmp/hosts.97515
MPI Wtime is 20.764897
+ rm /tmp/hosts.97515
~
~
```

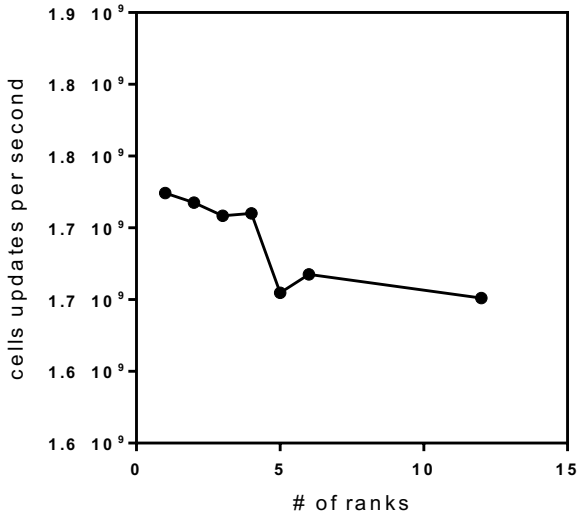
6. 1 node, 6 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 1024 CUDA thread block size and pattern 3.
Execution time for rank 0 is 20.605125 s, using MPI_Wtime command.
Since a world of 16K * 16K world size that runs for 128 iterations will have $3.43597*10^{10}$ cell updates, the "cell updates per second" is $1.6675*10^{9}$.

```
++ _ModuleTable_Sz_=3
++ export _ModuleTable_Sz_
++ : -s sh
+ eval
+ mpirun -hostfile /tmp/hosts.97521
MPI Wtime is 20.605125
+ rm /tmp/hosts.97521
~
~
```

7. 2 nodes, 12 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 1024 CUDA thread block size and pattern 3.
Execution time for rank 0 is 20.810619 s, using MPI_Wtime command.
Since a world of 16K * 16K world size that runs for 128 iterations will have $3.43597*10^{10}$ cell updates, the "cell updates per second" is $1.6511*10^{9}$.

```
++ _ModuleTable_Sz_=3
++ export _ModuleTable_Sz_
++ : -s sh
+ eval
+ mpirun -hostfile /tmp/hosts.97526
MPI Wtime is 20.810619
+ rm /tmp/hosts.97526
~
~
```

The plot of cells updates per second versus number of ranks is shown below.



From this figure, we know that the first configuration which is only one process yields the fastest "cell updates per second" rate.

When the number of ranks/processes is increased, the time spent on the communication between processes either with point-to-point (send/receive) communication, or synchronization (wait) would be increased. This is the reason why the first configuration with only one process yields the highest-performance computing.