> By Jiahui Tang

Spec:

- OS: AWS VM Ubuntu 18.04
- Cluster: (x2.2large) * 2 instances
  - Latency:

    ```
    rtt min/avg/max/mdev = 0.349/0.479/2.914/0.239 ms
    ```
  - Bandwidth:

    ```
    Amazon says it's 5 Gbps
    ```
- Compiler: gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
- More details below:
- Network Latency:

```
ubuntu@ip-172-30-8-69:~$ ping node1
PING ip-172-30-8-84 (172.30.8.84) 56(84) bytes of data.
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=242 ttl=64 time=0.428 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=243 ttl=64 time=0.440 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=244 ttl=64 time=0.517 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=245 ttl=64 time=0.455 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=246 ttl=64 time=0.366 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=247 ttl=64 time=0.486 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=248 ttl=64 time=0.424 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=249 ttl=64 time=0.418 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=250 ttl=64 time=0.496 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=251 ttl=64 time=0.447 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=252 ttl=64 time=0.482 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=253 ttl=64 time=0.454 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=254 ttl=64 time=0.418 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=255 ttl=64 time=0.495 ms
64 bytes from ip-172-30-8-84 (172.30.8.84): icmp_seq=256 ttl=64 time=0.375 ms
```

```
--- ip-172-30-8-84 ping statistics ---
386 packets transmitted, 145 received, 62% packet loss, time 394111ms
rtt min/avg/max/mdev = 0.349/0.479/2.914/0.239 ms
```

- Kernal Version:

```
ubuntu@ip-172-30-8-69:~$ uname -a
Linux ip-172-30-8-69 5.4.0-1038-aws #40~18.04.1-Ubuntu SMP Sat Feb 6 01:56:56 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
```

- MPI Info:

```
ubuntu@ip-172-30-8-84:~$ mpiexec --version
HYDRA build details:
    Version:                                 3.3a2
    Release Date:                            Sun Nov 13 09:12:11 MST 2016
    CC:                              gcc   -Wl,-Bsymbolic-functions -Wl,-z,relro
    CXX:                             g++   -Wl,-Bsymbolic-functions -Wl,-z,relro
    F77:                             gfortran  -Wl,-Bsymbolic-functions -Wl,-z,relro
    F90:                             gfortran  -Wl,-Bsymbolic-functions -Wl,-z,relro
    Configure options:                       '--disable-option-checking' '--prefix=/usr' '--build=x86_64-linux-gnu' '--includedir=${prefix}/include' '--mandir=${prefix}/share/man' '--infodir=${prefix}/share/info' '--sysconfdir=/etc' '--lo
calstatedir=/var' '--disable-silent-rules' '--libdir=${prefix}/lib/x86_64-linux-gnu' '--libexecdir=${prefix}/lib/x86_64-linux-gnu' '--disable-maintainer-mode' '--disable-dependency-tracking' '--with-libfabric' '--enable-shared' '--enable-
fortran=all' '--disable-rpath' '--disable-wrapper-rpath' '--sysconfdir=/etc/mpich' '--libdir=/usr/lib/x86_64-linux-gnu' '--includedir=/usr/include/mpich' '--docdir=/usr/share/doc/mpich' '--with-hwloc-prefix=system' '--enable-checkpointing
' '--with-hydra-ckpointlib=blcr' 'CPPFLAGS= -Wdate-time -D_FORTIFY_SOURCE=2 -I/build/mpich-O9at2o/mpich-3.3~a2/src/mpl/include -I/build/mpich-O9at2o/mpich-3.3~a2/src/mpl/include -I/build/mpich-O9at2o/mpich-3.3~a2/src/openpa/src -I/build/m
pich-O9at2o/mpich-3.3~a2/src/openpa/src -D_REENTRANT -I/build/mpich-O9at2o/mpich-3.3~a2/src/mpi/romio/include' 'CFLAGS= -g -O2 -fdebug-prefix-map=/build/mpich-O9at2o/mpich-3.3~a2=. -fstack-protector-strong -Wformat -Werror=format-security
 -O2' 'CXXFLAGS= -g -O2 -fdebug-prefix-map=/build/mpich-O9at2o/mpich-3.3~a2=. -fstack-protector-strong -Wformat -Werror=format-security -O2' 'FFLAGS= -g -O2 -fdebug-prefix-map=/build/mpich-O9at2o/mpich-3.3~a2=. -fstack-protector-strong -O
2' 'FCFLAGS= -g -O2 -fdebug-prefix-map=/build/mpich-O9at2o/mpich-3.3~a2=. -fstack-protector-strong -O2' 'build_alias=x86_64-linux-gnu' 'MPICHLIB_CFLAGS=-g -O2 -fdebug-prefix-map=/build/mpich-O9at2o/mpich-3.3~a2=. -fstack-protector-strong
-Wformat -Werror=format-security' 'MPICHLIB_CPPFLAGS=-Wdate-time -D_FORTIFY_SOURCE=2' 'MPICHLIB_CXXFLAGS=-g -O2 -fdebug-prefix-map=/build/mpich-O9at2o/mpich-3.3~a2=. -fstack-protector-strong -Wformat -Werror=format-security' 'MPICHLIB_FFL
AGS=-g -O2 -fdebug-prefix-map=/build/mpich-O9at2o/mpich-3.3~a2=. -fstack-protector-strong' 'MPICHLIB_FCFLAGS=-g -O2 -fdebug-prefix-map=/build/mpich-O9at2o/mpich-3.3~a2=. -fstack-protector-strong' 'LDFLAGS=-Wl,-Bsymbolic-functions -Wl,-z,r
elro' 'FC=gfortran' 'F77=gfortran' 'MPILIBNAME=mpich' '--cache-file=/dev/null' '--srcdir=.' 'CC=gcc' 'LIBS=' 'MPLLIBNAME=mpl'
    Process Manager:                         pmi
    Launchers available:                     ssh rsh fork slurm ll lsf sge manual persist
    Topology libraries available:            hwloc
    Resource management kernels available:   user slurm ll lsf sge pbs cobalt
    Checkpointing libraries available:       blcr
    Demux engines available:                 poll select
ubuntu@ip-172-30-8-84:~$
```

- CPU Info:

```
ubuntu@ip-172-30-8-69:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
stepping        : 1
microcode       : 0xb000038
cpu MHz         : 2300.073
cache size      : 46080 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 8
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx pdpe1gb rdt
scp lm constant_tsc rep_good nopl xtopology cpuid pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
xsave avx f16c rdrand hypervisor lahf_lm abm cpuid_fault invpcid_single pti fsgsbase bmi1 avx2 smep bmi2 erms invpcid xsaveopt
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs itlb_multihit
bogomips        : 4600.10
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

```
mpiuser@ip-172-30-8-69:~/cloud$ lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              8
On-line CPU(s) list: 0-7
Thread(s) per core:  1
Core(s) per socket:  8
Socket(s):           1
NUMA node(s):        1
Vendor ID:           GenuineIntel
CPU family:          6
Model:               79
Model name:          Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
Stepping:            1
CPU MHz:             2299.865
BogoMIPS:            4600.16
Hypervisor vendor:   Xen
Virtualization type: full
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            46080K
NUMA node0 CPU(s):   0-7
```

- Compiler:

```
ubuntu@ip-172-30-8-69:~$ mpicc --version
gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

5.2. Hybrid Parallel Processing (15 points)

OpenMP by itself is constrained to a single node. In the world of very large scale infrastructures, the motivation for using OpenMP is possible performance gains when combined with MPI. Develop a hybrid version of `jacobi1d_mpi.c` by including OpenMP directives to parallelize the two main execution loops. Include one `#pragma` for each loop. Compile and execute the code on multiple nodes and threads per node. Compare execution time in these three scenarios:

- SMP Nodes
  - Single MPI task launched per node
  - Parallel Threads share all node memory
  - For example: 4 threads/task and 1 task/node on system based on nodes with one 4-core CPU
- SMP Sockets (CPUs)
  - Single MPI task launched on each socket (CPU)
  - Parallel Thread set shares socket (CPU) memory, e.g. 4 threads/socket
  - For example: 2 threads/task and 2 tasks/node on system based on nodes with two 2-core CPUs
- No Shared Memory (all MPI)
  - Each core on a node is assigned an MPI task
  - For example: 1 thread/task and 4 tasks/node on system based on nodes with 4 cores

If you are using a 2-node MPI cluster on AWS you can emulate the three scenarios by running 4 threads/task and 1 task/node in the SMP Nodes mode; 2 threads/task and 2 tasks/node in the SMP Sockets (CPUs) mode, and 1 thread/task and 4 tasks/node in the No Shared Memory mode.

Submission
- `P52.pdf`: with replicability information (see Submission note above), Discussion of the results
- `P52.c`: Complete source code. Please insert comments in your source code to explain the type of parallelization you are applying

5.2 Table for Different Hybrid Mode in elapsed execution time

With `ncells = 10^8`, `step=100`, `2 nodes` from previous question.

Unit: Seconds

Hybrid Mode	SMP Nodes *(4 threads/task and 1 task/node)*	SMP Sockets *(2 threads/task and 2 tasks/node)*	No Shared Memory *(1 thread/task and 4 tasks/node)*
Elapsed Time	6.60708	6.56121	6.55368

> Result shows similar elapsed time for three hybrid models. I think they could all approximately and theoritically achieve `8x` speedup. As we know from 5.1.3, the sequential runtime is fluctuating around `48 seconds` for two nodes distributed memory parallel processing. Here the elapsed time for three modes are all around `6 seconds`, approximately reach `8 x` speedup.
>
> It makes sense because for SMP node: it has $4 * 1 * 2 = 8$ threads running concurrently; For SMP sockets, it has $2 * 2 * 2 = 8$; and similarly for no shared memory, it has $1 * 4 * 2 = 8$ threads.

> However, there is a slightly better performance for **No Shared Memory**, because it has 1 thread/task, no two threads will share same memory and thus it will incur less communication and waiting overhead. It takes the least elapsed time among three.
>
> Generally, here for three different hybrid models, no matter how we combine shared-memory within and distributed-memory across nodes, whether it's Single MPI task launched per node, Single MPI task launched on each socket, Each core on a node assigned an MPI task, they all parallelize the two main for loops into 8 sub threads. Their result is relatively the same. The hybrid model start with MPI initialization first, and then create OMP parallel regions within MPI task/processes. It then call MPI library from serial region or a single thread within parallel region. Just these 8 tasks are in different place, maybe in shared memory, or in distributed memory across node. Thus, the theoritical speedup should all be around `48/6 = 8` . Plus there is as there are only 8 threads, allocated to two nodes, there'no thread meets overhead threshold for communication and sychronization, thus all three hybrid types are close to theory value.

References:

```
mpiuser@ip-172-30-8-69:~/cloud$ export OMP_NUM_THREADS=4
mpiuser@ip-172-30-8-69:~/cloud$ mpirun -np 2 -hosts master,node1 ./p52 100000000 100
Processes: 2
n: 100000000
nsteps: 100
Elapsed time: 6.60708 s
mpiuser@ip-172-30-8-69:~/cloud$ export OMP_NUM_THREADS=2
mpiuser@ip-172-30-8-69:~/cloud$ mpirun -np 4 -hosts master,node1 ./p52 100000000 100
Processes: 4
n: 100000000
nsteps: 100
Elapsed time: 6.56121 s
mpiuser@ip-172-30-8-69:~/cloud$ export OMP_NUM_THREADS=1
mpiuser@ip-172-30-8-69:~/cloud$ mpirun -np 8 -hosts master,node1 ./p52 100000000 100
Processes: 8
n: 100000000
nsteps: 100
Elapsed time: 6.55368 s
mpiuser@ip-172-30-8-69:~/cloud$
```

--- old version ----

with `–np` flag wrong (it should be total processes in two nodes, thus should times 2)

```
mpiuser@ip-172-30-8-69:~/cloud$ export OMP_NUM_THREADS=4
mpiuser@ip-172-30-8-69:~/cloud$ mpirun -np 1 -hosts master,node1 ./p52 100000000 100
Processes: 1
n: 100000000
nsteps: 100
Elapsed time: 12.742 s
mpiuser@ip-172-30-8-69:~/cloud$ export OMP_NUM_THREADS=2
mpiuser@ip-172-30-8-69:~/cloud$ mpirun -np 2 -hosts master,node1 ./p52 100000000 100
Processes: 2
n: 100000000
nsteps: 100
Elapsed time: 12.3486 s
mpiuser@ip-172-30-8-69:~/cloud$ export OMP_NUM_THREADS=1
mpiuser@ip-172-30-8-69:~/cloud$ mpirun -np 4 -hosts master,node1 ./p52 100000000 100
Processes: 4
n: 100000000
nsteps: 100
Elapsed time: 12.3126 s
mpiuser@ip-172-30-8-69:~/cloud$
```