RUNTIME DATA HERE

Table 2. NAS Benchmark Run on 62 node Test Cluster

NAS Parallel Benchmarks			
Name	MPI C	MPI D	OpenMP C
Runtime (seconds)	2.24	39.36	102.91

Table 3. Simple Train Test Runs for a Single Segmented Simulation

Train Single Segmented Simulation – CSCI GPU machine (16 Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz cores)				
# nodes:	**1**	**1**	**1**	**1**
# threads / node:	**1**	**8**	**16**	**32**
Trials (time in seconds)	13.5847	2.355345	1.436542	1.433803
	13.604571	2.361201	1.44643	1.521244
	13.609432	2.36838	1.434071	1.452405
	13.620235	2.364279	1.431289	1.48212
	13.604831	2.363689	1.437159	1.45836
	13.609616	2.361085	1.494593	1.498602
	13.610614	2.36292	1.451776	1.48352
	13.601187	2.369198	1.44605	1.462887
	13.57455	2.427286	1.543422	1.471853
	13.570603	2.362629	1.431287	1.491564
Avg Runtime (seconds)	13.5990339	2.3696012	1.4552619	1.4756358
SU	**1.00**	**5.74**	**9.34**	**9.22**
# of threads	**1**	**8**	**16**	**32**
Sequential %	100.00%	5.63%	4.75%	7.98%
Parallel %	**0.00%**	**94.37%**	**95.25%**	**92.02%**

Table 4. GMAT Core Propagator Parallel Results – Shows Slow-down with Correct Results

Number of Threads	Start Position (X, Y, Z)	Final Position (X, Y, Z)	Runtime for RungeKutta::RawStep() (in secs)	Runtime for Propagate::Execute() (in secs)	Overall Mission Runtime (in secs)
1	(7100, 0, 1300)	(-510.4660737341223, 7304.983062653756, 736.2838199799176)	17.590199	19.570792	20.794

2	(7100, 0, 1300)	(-510.4660737210272, 7304.98306265423, 736.2838199825665)	21.386148	23.558907	24.365
4	(7100, 0, 1300)	(-510.4660737262405, 7304.983062653983, 736.283819981556)	25.718094	28.406107	29.320
8	(7100, 0, 1300)	(-510.4660737935445, 7304.983062649158, 736.2838199676164)	52.090615	56.722510	57.641

Table 5. GMAT Monte Carlo Scale-up and Scale-out Results

Cluster GMAT - 62 Simulations						
Uses MPI	**No**	**Yes**	**Yes**	**No**	**No**	**Yes**
Uses OpenMP	**No**	**No**	**No**	**Yes - SIMD**	**Yes - Block**	**Yes - Block**
# Nodes	**1**	**62**	**62**	**1**	**1**	**62**
Threads / Node	**1**	**1**	**1**	**1**	**4**	**4**
Avg Runtime (seconds)	1352.830031	24.21332725	24.33003906	1361.994834	2942.684812	56.6682888
SU	**1.00**	**55.87**	**55.60**	**0.99**	**0.46**	**23.87**
Sequential %	100.00%	0.18%	0.19%	100.00%	100.00%	3.80%
Parallel %	**0.00%**	**99.82%**	**99.81%**	**0.00%**	**0.00%**	**96.20%**

Notes from GMAT Setup:

cscigpu does not have tcsh, docker, or cmake, so GMAT cannot be compiled.

The NUC cluster does not have docker or cmake, so GMAT cannot be compiled.

Looked at SDSC OpenMP, MPI, and OpenMP + MPI Hybrid Slurm setup examples on the expanse server, found at: /cm/shared/examples/sdsc/mpi-openmp-hybrid/

Working on setting up scripts for testing on EXPANSE, wanted to get Train Sim results first, at this point.

Train Sim Testing - OCNL Cluster

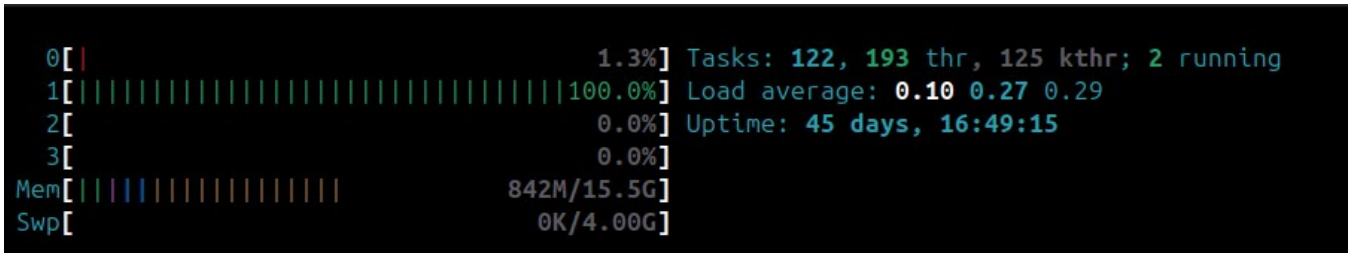
OCNL 251 nodes weren't working with MPI for some reason- I had the ssh key on all the nodes in both 251 and 244, was successfully able to ping all the nodes in both rooms, but MPI would only work with workers in 244, even when running the script from a computer in 251.

Output from lscpu:

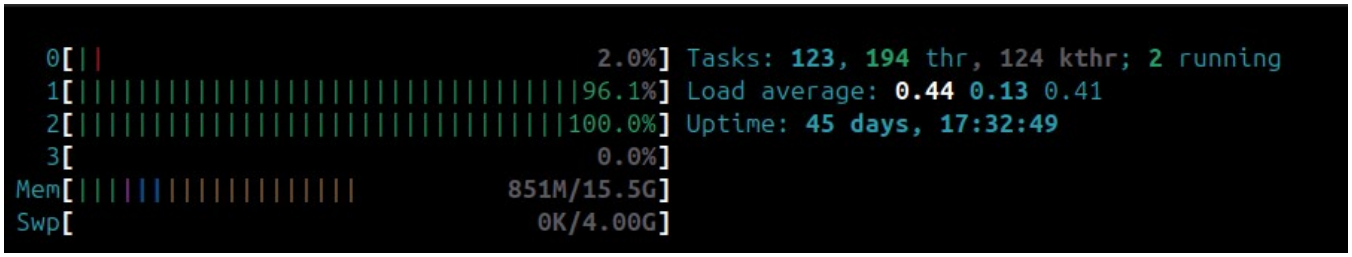
```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  4
  On-line CPU(s) list:   0-3
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Core(TM) i5-5250U CPU @ 1.60GHz
    CPU family:          6
    Model:               61
    Thread(s) per core:  2
    Core(s) per socket:  2
    Socket(s):           1
    Stepping:            4
    CPU(s) scaling MHz:  31%
    CPU max MHz:         2700.0000
    CPU min MHz:         500.0000
    BogoMIPS:            3192.82
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat ps
                         e36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1
                         gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopol
                         ogy nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl
                          vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic
                          movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm a
                         bm 3dnowprefetch cpuid_fault epb pti ssbd ibrs ibpb stibp tpr_shadow
                          flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi
                         2 erms invpcid rdseed adx smap intel_pt xsaveopt dtherm ida arat pln
                          pts vnmi md_clear flush_l1d ibpb_exit_to_user
Virtualization features:
  Virtualization:        VT-x
Caches (sum of all):
  L1d:                   64 KiB (2 instances)
  L1i:                   64 KiB (2 instances)
  L2:                    512 KiB (2 instances)
  L3:                    3 MiB (1 instance)
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0-3
Vulnerabilities:
  Gather data sampling:  Not affected
  Itlb multihit:         KVM: Mitigation: VMX disabled
  L1tf:                  Mitigation; PTE Inversion; VMX conditional cache flushes, SMT vulner
```

able
Mds: Mitigation; Clear CPU buffers; SMT vulnerable
Meltdown: Mitigation; PTI
Mmio stale data: Unknown: No mitigations
Reg file data sampling: Not affected
Retbleed: Not affected
Spec rstack overflow: Not affected
Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl
Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Spectre v2: Mitigation; Retpolines; IBPB conditional; IBRS_FW; STIBP conditional
; RSB filling; PBRSB-eIBRS Not affected; BHI Not affected
Srbds: Mitigation; Microcode
Tsx async abort: Not affected
Vmscape: Mitigation; IBPB before exit to userspace

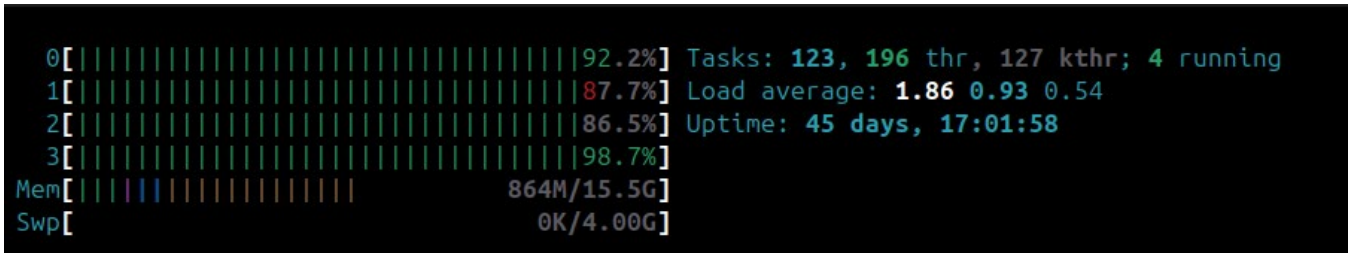
HTOP from one-threaded trials:



HTOP from two-threaded trials:



HTOP from four-threaded trials:



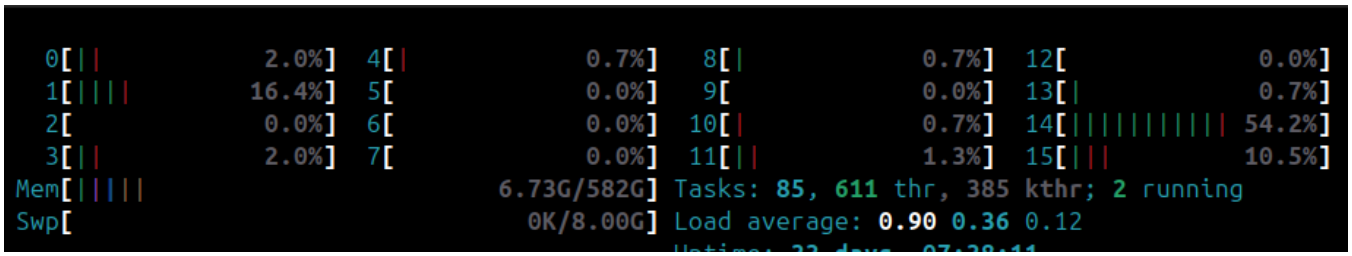
Train Sim Testing - CSCIGPU

Lscpu on cscigpu:

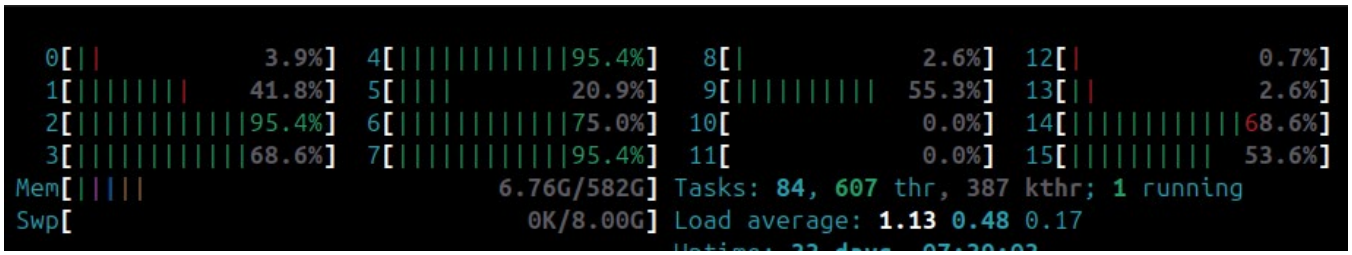
```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         45 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  16
  On-line CPU(s) list:   0-15
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz
    CPU family:          6
    Model:               85
    Thread(s) per core:  1
    Core(s) per socket:  1
    Socket(s):           16
    Stepping:            7
    BogoMIPS:            4190.15
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
                         pse36 clflush mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm
                         constant_tsc arch_perfmon nopl xtopology tsc_reliable nonstop_tsc
                         cpuid tsc_known_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 ss
                         e4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rd
                         rand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault ssbd ibrs i
                         bpb stibp ibrs_enhanced fsgsbase tsc_adjust bmi1 avx2 smep bmi2 i
                         nvpcid avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd
                         avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves arat pku ospke a
                         vx512_vnni md_clear flush_l1d arch_capabilities
Virtualization features:
  Hypervisor vendor:     VMware
  Virtualization type:   full
Caches (sum of all):
  L1d:                   512 KiB (16 instances)
  L1i:                   512 KiB (16 instances)
  L2:                    16 MiB (16 instances)
  L3:                    440 MiB (16 instances)
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0-15
Vulnerabilities:
  Gather data sampling:  Unknown: Dependent on hypervisor status
  Ghostwrite:            Not affected
  Indirect target selection: Mitigation; Aligned branch/return thunks
  Itlb multihit:         KVM: Mitigation: VMX unsupported
  L1tf:                  Not affected
```

Mds: Not affected
Meltdown: Not affected
Mmio stale data: Vulnerable: Clear CPU buffers attempted, no microcode; SMT Host s
tate unknown
Reg file data sampling: Not affected
Retbleed: Mitigation; Enhanced IBRS
Spec rstack overflow: Not affected
Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl
Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitizat
ion
Spectre v2: Mitigation; Enhanced / Automatic IBRS; IBPB conditional; PBRSB-eI
BRS SW sequence; BHI SW loop, KVM SW loop
Srbds: Not affected
Tsa: Not affected
Tsx async abort: Not affected
Vmscape: Not affected

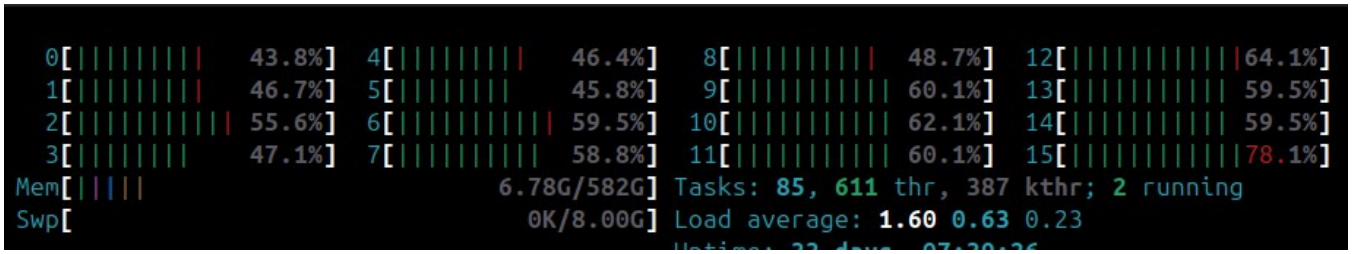
HTOP from one-threaded test



HTOP from eight-threaded test:

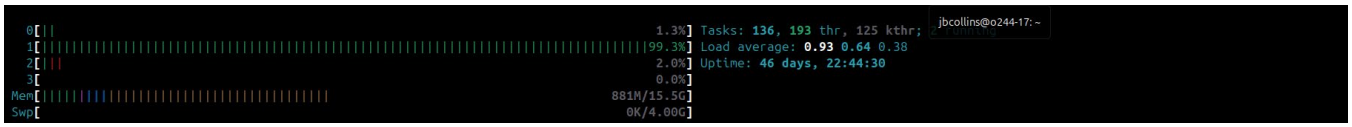


HTOP from 16-threaded test:

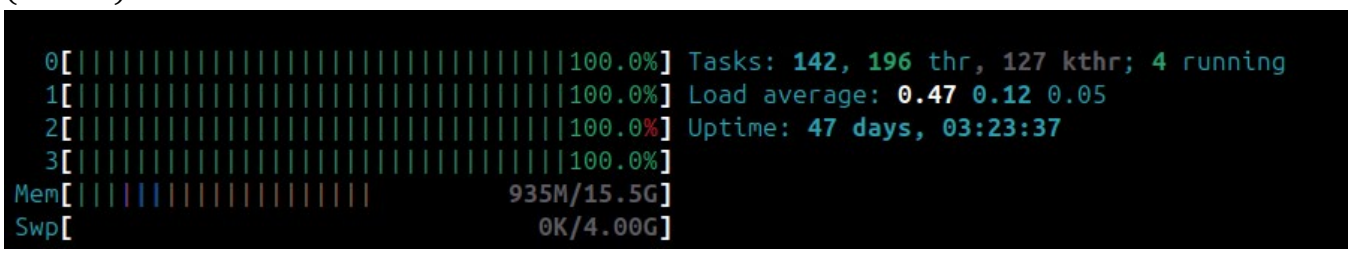


GMAT Simulation Tests - OCNL Cluster

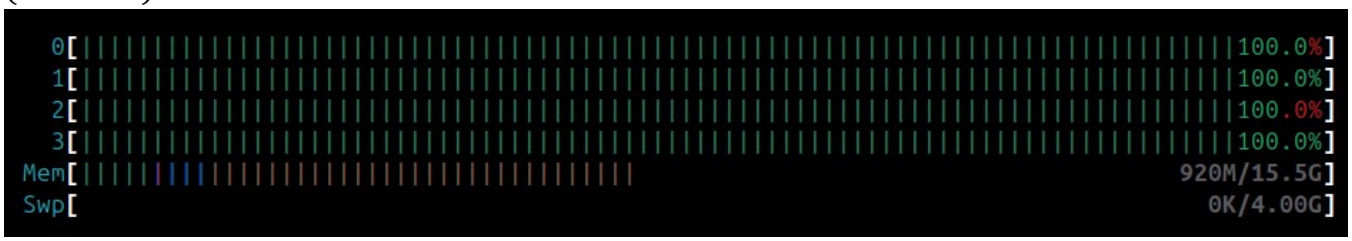
HTOP from single-core test:



HTOP from OpenMP block test (n_threads=4):
(no MPI):



(with MPI):



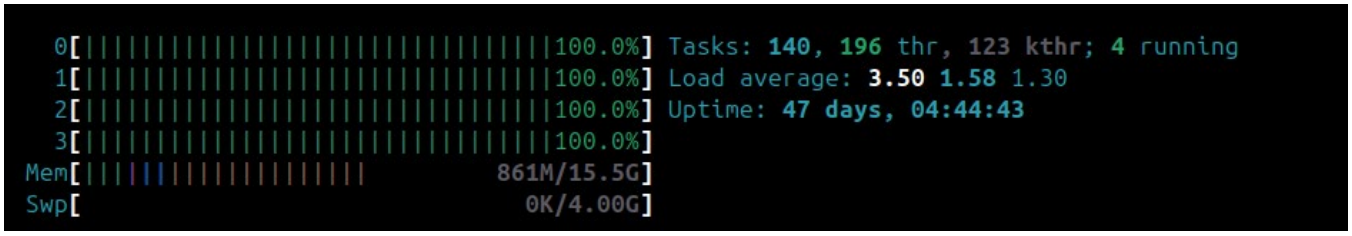
I had to update Angel's code to work with Chirag's updates - added an threads parameter which updates the scripts generated by the montecarlo_wrapper to include the ThreadCount line in a parameterized manner

NAS Benchmarks

OpenMP - Embarrassingly Parallel

git clone https://github.com/GMAP/NPB-CPP
cd NPB-CPP/NPB-OMP
make ep CLASS=C
cd bin
export OMP_NUM_THREADS=4
nohup time ./ep.C > ep-C.txt &

Htop during above execution:



Output will be saved to <benchmark-dir>/bin/ep-C.txt

Output:

```
NAS Parallel Benchmarks 4.1 Parallel C++ version with OpenMP - EP Benchmark

Number of random numbers generated:      8589934592

EP Benchmark Results:

CPU Time =  102.9082
N = 2^   32
No. Gaussian Pairs =      3373275903
Sums =     4.764367927995942e+04    -8.084072988039244e+04
Counts:
 0    1572172634
 1    1501108549
 2     281805648
 3      17761221
 4        424017
 5          3821
 6            13
 7             0
 8             0


EP Benchmark Completed
class_npb       =                 C
Size            =        8589934592
Total threads   =                 4
Iterations      =                 0
Time in seconds =            102.91
Mop/s total     =             83.47
Operation type  = Random numbers generated
Verification    =        SUCCESSFUL
Version         =               4.1
Compile date    =       07 Jan 2026
Compiler ver    =            13.3.0
OpenMP version  =            201511

Compile options:
  CC        = g++ -std=c++14
  CLINK     = $(CC)
  C_LIB     = -lm
```

```
  C_INC       = -I../common
  CFLAGS      = -O3 -fopenmp -mcmodel=medium
  CLINKFLAGS  = -O3 -fopenmp -mcmodel=medium
  RAND        = randdp


----------------------------------------------------------------------
  NPB-CPP is developed by:
     Dalvan Griebler
     Gabriell Araujo (Sequential Porting)
     Júnior Löff (Parallel Implementation)

  In case of questions or problems, please send an e-mail to us:
     dalvan.griebler; gabriell.araujo; junior.loff@edu.pucrs.br
----------------------------------------------------------------------

407.63user 0.06system 1:42.91elapsed 396%CPU (0avgtext+0avgdata 9344maxresident)k
0inputs+8outputs (0major+1447minor)pagefaults 0swaps
```

MPI - Embarrassingly Parallel

wget https://www.nas.nasa.gov/assets/npb/NPB3.3.1.tar.gz
tar -xvf NPB3.3.1.tar.gz
cd NPB3.3.1/NPB3.3-MPI

Edit config/make.def to match the following:

```
#---------------------------------------------------------------------------
#
#              SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS.
#
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
# Items in this file will need to be changed for each platform.
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
# Parallel Fortran:
#
# For CG, EP, FT, MG, LU, SP and BT, which are in Fortran, the following must
# be defined:
#
# MPIF77     - Fortran compiler
```

```
# FFLAGS     - Fortran compilation arguments
# FMPI_INC   - any -I arguments required for compiling MPI/Fortran
# FLINK      - Fortran linker
# FLINKFLAGS - Fortran linker arguments
# FMPI_LIB   - any -L and -l arguments required for linking MPI/Fortran
#
# compilations are done with $(MPIF77) $(FMPI_INC) $(FFLAGS) or
#                            $(MPIF77) $(FFLAGS)
# linking is done with       $(FLINK) $(FMPI_LIB) $(FLINKFLAGS)
#-----------------------------------------------------------------------

#-----------------------------------------------------------------------
# This is the fortran compiler used for MPI programs
#-----------------------------------------------------------------------
MPIF77 = mpif77
# This links MPI fortran programs; usually the same as ${MPIF77}
FLINK = $(MPIF77)

#-----------------------------------------------------------------------
# These macros are passed to the linker to help link with MPI correctly
#-----------------------------------------------------------------------
FMPI_LIB  = -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib/debug -
L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib

#-----------------------------------------------------------------------
# These macros are passed to the compiler to help find 'mpif.h'
#-----------------------------------------------------------------------
FMPI_INC = -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/include/

#-----------------------------------------------------------------------
# Global *compile time* flags for Fortran programs
#-----------------------------------------------------------------------
FFLAGS          = -O0 -fallow-argument-mismatch

#-----------------------------------------------------------------------
# Global *link time* flags. Flags for increasing maximum executable
# size usually go here.
#-----------------------------------------------------------------------
FLINKFLAGS = -O


#-----------------------------------------------------------------------
```

```
# Parallel C:
#
# For IS, which is in C, the following must be defined:
#
# MPICC     - C compiler
# CFLAGS    - C compilation arguments
# CMPI_INC  - any -I arguments required for compiling MPI/C
# CLINK     - C linker
# CLINKFLAGS - C linker flags
# CMPI_LIB  - any -L and -l arguments required for linking MPI/C
#
# compilations are done with $(MPICC) $(CMPI_INC) $(CFLAGS) or
#                            $(MPICC) $(CFLAGS)
# linking is done with       $(CLINK) $(CMPI_LIB) $(CLINKFLAGS)
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
# This is the C compiler used for MPI programs
#---------------------------------------------------------------------------
MPICC = mpicc
# This links MPI C programs; usually the same as ${MPICC}
CLINK = $(MPICC)

#---------------------------------------------------------------------------
# These macros are passed to the linker to help link with MPI correctly
#---------------------------------------------------------------------------
CMPI_LIB  = -L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib/debug -
L/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/lib

#---------------------------------------------------------------------------
# These macros are passed to the compiler to help find 'mpi.h'
#---------------------------------------------------------------------------
CMPI_INC = -I/opt/intel/compilers_and_libraries_2020.0.166/linux/mpi/intel64/include/

#---------------------------------------------------------------------------
# Global *compile time* flags for C programs
#---------------------------------------------------------------------------
CFLAGS          = -O0

#---------------------------------------------------------------------------
# Global *link time* flags. Flags for increasing maximum executable
# size usually go here.
```

```
#---------------------------------------------------------------------------
CLINKFLAGS = -O0


#---------------------------------------------------------------------------
# MPI dummy library:
#
# Uncomment if you want to use the MPI dummy library supplied by NAS instead
# of the true message-passing library. The include file redefines several of
# the above macros. It also invokes make in subdirectory MPI_dummy. Make
# sure that no spaces or tabs precede include.
#---------------------------------------------------------------------------
# include ../config/make.dummy


#---------------------------------------------------------------------------
# Utilities C:
#
# This is the C compiler used to compile C utilities.  Flags required by
# this compiler go here also; typically there are few flags required; hence
# there are no separate macros provided for such flags.
#---------------------------------------------------------------------------
CC       = gcc -g


#---------------------------------------------------------------------------
# Destination of executables, relative to subdirs of the main directory. .
#---------------------------------------------------------------------------
BINDIR           = ../bin


#---------------------------------------------------------------------------
# Some machines (e.g. Crays) have 128-bit DOUBLE PRECISION numbers, which
# is twice the precision required for the NPB suite. A compiler flag
# (e.g. -dp) can usually be used to change DOUBLE PRECISION variables to
# 64 bits, but the MPI library may continue to send 128 bits. Short of
# recompiling MPI, the solution is to use MPI_REAL to send these 64-bit
# numbers, and MPI_COMPLEX to send their complex counterparts. Uncomment
# the following line to enable this substitution.
#
# NOTE: IF THE I/O BENCHMARK IS BEING BUILT, WE USE CONVERTFLAG TO
#       SPECIFIY THE FORTRAN RECORD LENGTH UNIT. IT IS A SYSTEM-SPECIFIC
```

```
#      VALUE (USUALLY 1 OR 4). UNCOMMENT THE SECOND LINE AND SUBSTITUTE
#      THE CORRECT VALUE FOR "length".
#      IF BOTH 128-BIT DOUBLE PRECISION NUMBERS AND I/O ARE TO BE ENABLED,
#      UNCOMMENT THE THIRD LINE AND SUBSTITUTE THE CORRECT VALUE FOR
#      "length"
#---------------------------------------------------------------------------
# CONVERTFLAG      = -DCONVERTDOUBLE
# CONVERTFLAG      = -DFORTRAN_REC_SIZE=length
# CONVERTFLAG      = -DCONVERTDOUBLE -DFORTRAN_REC_SIZE=length


#---------------------------------------------------------------------------
# The variable RAND controls which random number generator
# is used. It is described in detail in README.install.
# Use "randi8" unless there is a reason to use another one.
# Other allowed values are "randi8_safe", "randdp" and "randdpvec"
#---------------------------------------------------------------------------
#RAND   = randi8
RAND   = randi8_safe
# The following is highly reliable but may be slow:
# RAND   = randdp
#END MAKEFILE DEFINITION

make "ep" NPROCS=248 CLASS="C" SUBTYPE="simple"
cp ../../hello_cluster/c1c2_hosts . # (or wherever you have c1c2_hosts)
mpiexec -n 248 -f c1c2_hosts ./bin/ep.C.248

Output:
 NAS Parallel Benchmarks 3.3 -- EP Benchmark

 Number of random numbers generated:     8589934592
 Number of active processes:                 248

EP Benchmark Results:

CPU Time =    2.2421
N = 2^   32
No. Gaussian Pairs =    3373275903.
Sums =     4.764367927991106D+04   -8.084072988047442D+04
Counts:
  0    1572172634.
  1    1501108549.
```

```
 2    281805648.
 3     17761221.
 4       424017.
 5         3821.
 6           13.
 7            0.
 8            0.
 9            0.


 EP Benchmark Completed.
 Class           =                     C
 Size            =            8589934592
 Iterations      =                     0
 Time in seconds =                  2.24
 Total processes =                   248
 Compiled procs  =                   248
 Mop/s total     =               3831.18
 Mop/s/process   =                 15.45
 Operation type  = Random numbers generated
 Verification    =            SUCCESSFUL
 Version         =                 3.3.1
 Compile date    =           08 Jan 2026

 Compile options:
   MPIF77      = mpif77
   FLINK       = $(MPIF77)
   FMPI_LIB    = -L/opt/intel/compilers_and_libraries_2020.0...
   FMPI_INC    = -I/opt/intel/compilers_and_libraries_2020.0...
   FFLAGS      = -O0 -fallow-argument-mismatch
   FLINKFLAGS  = -O
   RAND        = randi8_safe


 Please send feedbacks and/or the results of this run to:

 NPB Development Team
 Internet: npb@nas.nasa.gov
```

Output from class D benchmark:

```
 NAS Parallel Benchmarks 3.3 -- EP Benchmark
```

Number of random numbers generated: 137438953472
Number of active processes: 248

EP Benchmark Results:

CPU Time = 39.3579
N = 2^ 36
No. Gaussian Pairs = 53972171957.
Sums = 1.982481200938577D+05 -1.020596636363982D+05
Counts:
0 25154622775.
1 24017899906.
2 4508609839.
3 284201296.
4 6776403.
5 61541.
6 197.
7 0.
8 0.
9 0.

EP Benchmark Completed.
Class = D
Size = 137438953472
Iterations = 0
Time in seconds = 39.36
Total processes = 248
Compiled procs = 248
Mop/s total = 3492.03
Mop/s/process = 14.08
Operation type = Random numbers generated
Verification = SUCCESSFUL
Version = 3.3.1
Compile date = 08 Jan 2026

Compile options:
MPIF77 = mpif77
FLINK = $(MPIF77)
FMPI_LIB = -L/opt/intel/compilers_and_libraries_2020.0...
FMPI_INC = -I/opt/intel/compilers_and_libraries_2020.0...
FFLAGS = -O0 -fallow-argument-mismatch

```
FLINKFLAGS   = -O
RAND         = randi8_safe
```

Please send feedbacks and/or the results of this run to:

NPB Development Team
Internet: npb@nas.nasa.gov