LA-UR-23-33941

Approved for public release; distribution is unlimited.

Title:	Two MCNP Models for Computational Performance Benchmarking
Author(s):	Kulesza, Joel A.
Intended for:	External Collaboration
Issued:	2024-03-11 (rev.1)

Two MCNP Models for Computational Performance Benchmarking

Joel A. Kulesza
Los Alamos National Laboratory
Monte Carlo Codes Group (XCP-3)

1 Introduction

The purpose of this report is to describe two non-trivial MCNP models and execution configurations that are suitable to characterizing computational performance. To that end, this report uses a constructive solid geometry (CSG) representation of the Oak Ridge National Laboratory (ORNL) Pool Critical Assembly (PCA) [1–3] to perform a k-eigenvalue calculation and an unstructured mesh (UM) representation of the International Commission on Radiological Protection (ICRP) publication 145 (ICRP145) male human phantom [4, 5] to perform a fixed-source calculation assuming that a 1 MeV photon source is distributed throughout the phantom's liver.

The results are given in Tables 1 and 2. Instructions for acquiring and modifying the input files to reproduce this work are given in Appendix A and instructions for executing the MCNP code are given in Appendix B. Linux `ldd` output for both executables used in this work is given in Appendix C.

Revision 1 of this report corrects Table 1 and 2 values, which were improperly attributed to each calculation on Rocinante, and updates Listing 6 to associate 14 threads per NUMA domain (rather than the original 56) to appropriately fit to the processor's architecture. In addition, Listings 5 and 6 add STDOUT and STDERR redirection to a ".dayfile" and ".errfile", respectively.

1.1 Pool Critical Assembly Overview

Since its publication in 1997, the Oak Ridge National Laboratory (ORNL) Pool Critical Assembly (PCA) pressure vessel wall benchmark facility (PVWBF) documented in [1] has been a standard benchmark for qualifying light water reactor radiation transport methods and data. As such, it has been extensively studied and provides a realistic albeit relatively

easily described system that provides experimental data to validate both k-eigenvalue and neutron activation analyses. For this work, a k-eigenvalue calculation is performed because it requires only a single MCNP execution arising from a single input file.

1.2 ICRP145 Male Human Phantom Overview

Since roughly the late 1970s, the ICRP, often in concert with the International Commission on Radiation Units (ICRU), has investigated methods to characterize harm to humans caused by incident radiation. These methods have evolved over time from using a 30 cm sphere with a tissue-equivalent material [6], to human "phantom" models using Cartesian voxels [7], to more-detailed human phantom models constructed using tetrahedral UM elements [8]. This latest approach represents the state of the art and contains a subset of human phantom models available at https://mesh-phantom.com to perform MCNP analyses. For the purpose of this work, a fixed-source coupled photon-electron calculation is made using the adult male phantom provided with the supplemental materials of [8] that includes an already-defined internal photon source.

2 High-Performance Computer Summary

This section provides information on the two high-performance computers (HPCs) used to provide representative timing results for the aforementioned MCNP calculations. For this work, all executions use MCNP6.3 [9, 10].

2.1 Snow

The Los Alamos National Laboratory Snow supercomputer is composed of 368 nodes with 36 cores (2× E5 2695v4 (i.e., Broadwell) Intel processors running at 2.1 GHz) per node and 128 GB (DDR4-2400) of memory per node. Thus, the computer has 13,248 total CPU cores and 47.1 TB of total memory. The computer runs the Tri-Lab Operating System Stack, a version of Linux based on Red Hat (RHEL 7.9, kernel version 3.10.0-1160.95.1.1chaos.ch6.x86_64). OpenMPI is used to provide task parallelism, while OpenMP (OMP) is used for thread parallelism.

2.2 Rocinante

The Los Alamos National Laboratory Rocinante (also known colloquially as "Roci") supercomputer is composed of 508 nodes with 112 cores (2× Platinum 8480 (i.e., Sapphire Rapids) Intel processors running at 2.0 GHz) per node. Of the 508 nodes, 128 have high-bandwidth memory (128 GB of memory per node) and 380 have double data-rate memory (256 GB

of memory per node). Thus, the computer has 56,896 total CPU cores and 16.4 TB of high-bandwidth memory and 97.3 TB of double data-rate memory. The computer runs the HPE Cray Shasta operating system (SUSE Linux Enterprise Server 15 SP4, kernel 5.14.21-150400.24.46_12.0.72-cray_shasta_c). MPICH MPI is used to provide task parallelism, while OMP is used for thread parallelism.

3 Execution Configuration & Results

Because both aforementioned HPCs have dual-socket CPUs, the calculations in this work use both task (MPI) and thread (OMP) parallelism, as appropriate. That is, when each calculation uses multiple nodes, two MPI tasks are used on each node (one for each socket) and the appropriate number of threads per task (to use all processing cores on each socket). This enables the maximum amount of shared memory use, which ensures the calculations can be fit into memory. Note that when task parallelism is used, one MCNP task is left idle to aggregate results which leads to an incomplete use of resources. Further, for MCNP UM calculations, there is no thread parallelism during mesh input processing.

To make non-trivial use of the HPCs, to permit reasonable throughput in the job scheduling queue, and to permit relatively short but meaningfully measured execute times, up to approximately 300 cores were used for these calculations. On the Snow HPC, this means that a maximum of ten nodes are used to provide 360 cores while on the Rocinante HPC a maximum of three nodes are used to provide 336 cores.

The PCA input file is retrieved and modified as noted in Appendix A.1. The ICRP145 input files are retrieved and modified as noted in Appendix A.2. The total run time observed for both calculations on both HPCs is reported in Table 1. The Monte Carlo history throughput (i.e., average millions of histories per hour) is reported in Table 2. These results are also electronically attached to this document as CSV files.

References

[1] I. Remec and F. B. K. Kam, "Pool Critical Assembly Pressure Vessel Facility Benchmark," Oak Ridge National Laboratory, Oak Ridge, TN, USA, Tech. Rep. NUREG/CR-6454, Jul. 1997.

[2] J. A. Kulesza and R. L. Martz, "Evaluation of the Pool Critical Assembly Benchmark with Explicitly Modeled Geometry Using MCNP6," *Nuclear Technology*, vol. 197, no. 3, pp. 284–295, Mar. 2017. DOI: 10.1080/00295450.2016.1273711.

Table 1: Execution Configuration and Wall-Clock Run Time (HH:MM:SS)

Configuration	Snow			Rocinante		
	Cores	PCA Time	ICRP145 Time	Cores	PCA Time	ICRP145 Time
1 Socket (OMP)	18	03:53:16	08:53:17	56	01:05:46	03:26:42
1 Node (OMP)	36	03:50:08	08:49:46	112	00:32:26	01:54:39
2 Nodes (MPI+OMP)	72	01:17:50	03:16:08	224	00:16:01	01:05:26
3 Nodes (MPI+OMP)	108	00:47:47	02:06:24	336	00:11:06	00:50:46
4 Nodes (MPI+OMP)	144	00:34:54	01:39:38	—	—	—
5 Nodes (MPI+OMP)	180	00:27:34	01:22:56	—	—	—
6 Nodes (MPI+OMP)	216	00:22:57	01:11:16	—	—	—
7 Nodes (MPI+OMP)	252	00:19:39	01:02:41	—	—	—
8 Nodes (MPI+OMP)	288	00:17:17	00:57:51	—	—	—
9 Nodes (MPI+OMP)	324	00:15:34	00:52:02	—	—	—
10 Nodes (MPI+OMP)	360	00:14:13	00:49:25	—	—	—

Table 2: Execution Configuration and Throughput Rate (Million Histories/Hour)

Configuration	Snow			Rocinante		
	Cores	PCA Rate	ICRP145 Rate	Cores	PCA Rate	ICRP145 Rate
1 Socket (OMP)	18	6.45	7.12	56	23.19	19.42
1 Node (OMP)	36	6.54	7.15	112	47.95	39.25
2 Nodes (MPI+OMP)	72	19.44	21.13	224	100.72	78.95
3 Nodes (MPI+OMP)	108	31.85	34.89	336	150.17	124.29
4 Nodes (MPI+OMP)	144	43.94	48.53	—	—	—
5 Nodes (MPI+OMP)	180	55.96	61.93	—	—	—
6 Nodes (MPI+OMP)	216	67.60	75.66	—	—	—
7 Nodes (MPI+OMP)	252	79.00	89.46	—	—	—
8 Nodes (MPI+OMP)	288	90.20	102.68	—	—	—
9 Nodes (MPI+OMP)	324	101.15	116.70	—	—	—
10 Nodes (MPI+OMP)	360	111.53	129.72	—	—	—

[3] J. A. Kulesza, "Oak Ridge National Laboratory Pool Critical Assembly MCNP6 Criticality Calculation Input File," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-20-21532, Feb. 2020. DOI: 10.2172/1601379.

[4] Y. S. Yeom, M. C. Han, C. Choi, H. Han, B. Shin, T. Furuta, and C. H. Kim, "Computation Speeds and Memory Requirements of Mesh-Type ICRP Reference Computational Phantoms in Geant4, MCNP6, and PHITS," *Health Physics*, vol. 116, no. 5, pp. 664–676, May 2019. DOI: 10.1097/HP.0000000000000999.

[5] C. H. Kim, Y. S. Yeom, N. Petoussi-Henss, M. Zankl, W. E. Bolch, C. Lee, C. Choi, T. T. Nguyen, K. Eckerman, H. S. Kim, M. C. Han, R. Qiu, B. S. Chung, H. Han, and B. Shin, "Adult Mesh-type Reference Computational Phantoms," International Commission on Radiological Protection (ICRP), Ottawa, Ontario, Canada, Tech. Rep. ICRP Publication 145, 2020. URL: https://icrp.org/publication.asp?id=ICRPPublication145.

[6] ICRP, "Recommendations of the International Commission on Radiological Protection," *Annals of the ICRP*, vol. 1, no. 3, Jul. 1977, ICRP Publication 26. URL: https://journals.sagepub.com/toc/anib/1/3.

[7] ICRP, "Adult Reference Computational Phantoms," *Annals of the ICRP*, vol. 39, no. 2, Apr. 2009, ICRP Publication 110. URL: https://journals.sagepub.com/toc/anib/39/2.

[8] ICRP, "Adult Mesh-Type Reference Computational Phantoms," *Annals of the ICRP*, vol. 49, no. 3, Oct. 2020, ICRP Publication 145. URL: https://journals.sagepub.com/toc/anib/49/3.

[9] M. E. Rising, J. C. Armstrong, S. R. Bolding, F. B. Brown, J. S. Bull, T. P. Burke, A. R. Clark, D. A. Dixon, R. A. Forster, III, J. F. Giron, T. S. Grieve, H. G. Hughes, III, C. J. Josey, J. A. Kulesza, R. L. Martz, A. P. McCartney, G. W. McKinney, S. W. Mosher, E. J. Pearson, C. J. Solomon, Jr., S. Swaminarayan, J. E. Sweezy, S. C. Wilson, and A. J. Zukaitis, "MCNP® Code Version 6.3.0 Release Notes," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-22-33103, Rev. 1, Jan. 2023. DOI: 10.2172/1909545.

[10] J. A. Kulesza, T. R. Adams, J. C. Armstrong, S. R. Bolding, F. B. Brown, J. S. Bull, T. P. Burke, A. R. Clark, R. A. Forster, III, J. F. Giron, A. S. Grieve, C. J. Josey, R. L. Martz, G. W. McKinney, E. J. Pearson, M. E. Rising, C. J. Solomon, Jr., S. Swaminarayan, T. J. Trahan, S. C. Wilson, and A. J. Zukaitis, "MCNP® Code Version 6.3.0 Theory & User Manual," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-22-30006, Rev. 1, Sep. 2022. DOI: 10.2172/1889957.

A Where to Get, and How to Modify, Input Files

This appendix describes how to obtain and modify the input files used in this report. To permit straightforward and reproducible modifications, `diff`-based patch files are provided in listings and are electronically attached to this PDF.

A.1 Pool Critical Assembly

The MCNP input file needed for this work is electronically attached to [3] with instructions for retrieving it given in that document.

The only change necessary to the input file is to adjust the number of neutrons per generation. This values is increased from 10,000 to provide meaningful execution times. However, an improvement is also made to the output format of mesh tallies to use the new, recommended, format available in MCNP6.3: HDF5+XDMF. A patch file to perform these changes is given in Listing 1 and can be applied with the command `patch -i pca.patch.txt pca_kcode.mcnp.inp.txt`.

A.2 ICRP145 Male Human Phantom

Because of how MCNP UM calculations are structured, there are several input files needed for this calculation. All input files are available in the ZIP file available for download as the Supplemental Material[1] from the ICRP145 webpage[2].

The files within the aforementioned ZIP file are organized into directories, but for this work, the files are arranged to reside in the same directory to simplify the directory structure. The changes needed to the MCNP input file to accommodate this reorganization are captured in the patch file in Listing 2. The five files needed for this work are identified in Fig. 1. Because all files reside in the same directory for this work, they must be copied from the structure shown in Fig. 1 to the execution directory.

In addition to the aforementioned file-path changes, the random number generator is changed from the default of 1 to 2 and a random seed is specified, HDF5-formatted UM output is added, the `PRDMP` card is changed to enable timing information output, and the number of histories is increased to provide meaningful execution times.

[1] https://www.icrp.org/docs/P145%20Electronic%20files.zip
[2] https://www.icrp.org/publication.asp?id=ICRP%20Publication%20145

In summary, the patch file to perform all necessary changes is given in Listing 2 and can be applied with the command `patch -i icrp145.patch.txt MRCP-AM_internal_input`.

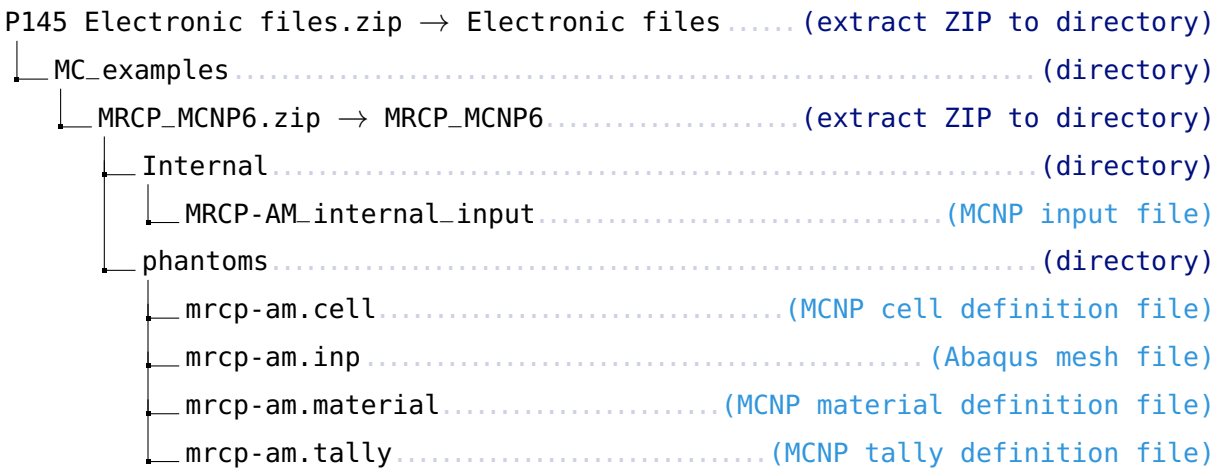

Figure 1: ICRP145 Supplemental Material Input File Locations

Listing 1: pca.patch.txt

```
--- orig/pca_kcode_orig.mcnp.inp  2023-11-29 11:43:34.000000000 -0700
+++ pca_kcode.mcnp.inp  2023-12-01 22:56:52.000000000 -0700
@@ -2379,7 +2379,7 @@
 c cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 c
 fmesh4:n geom   = xyz
-         out    = col
+         out    = xdmf
          origin = -20.41 -19.275 -36.35375
          iints  =   20       20
          imesh  =  0.0    20.25
@@ -2389,7 +2389,7 @@
          kmesh  =  0.0    36.35375
 c
 fmesh14:n geom   = xyz
-          out    = col
+          out    = xdmf
           origin = -20.41 -19.275 -36.35375
           iints  =    1      1     1      1      1
           imesh  =  -12.15  -4.05  4.05  12.15  20.25
@@ -2404,4 +2404,4 @@
 c
 rand gen=2 seed=12345
 print
-kcode 10000 1.000 50 250
+kcode 100000 1.000 50 250
```

Listing 2: icrp145.patch.txt

```
--- Electronic files/MC_examples/MRCP_MCNP6/Internal/MRCP-AM_internal_input
    2018-08-14 13:50:54.000000000 -0600
+++ mrcp-am_internal_input_sn_036.mcnp.inp  2023-12-05 07:30:39.000000000 -0700
@@ -10,7 +10,7 @@
 C ------------------------------------------------------------------------------
 C PSUEDO CELLS FOR ABAQUS
 C ------------------------------------------------------------------------------
-read file=../phantoms/mrcp-am.cell
+read file=mrcp-am.cell
 C ------------------------------------------------------------------------------
 C LEGACY CELLS
 C ------------------------------------------------------------------------------
@@ -31,22 +31,23 @@
 mode p e                        $ Track photons and electrons
 mphys on                        $ Turn on model physics
 imp:p,e 1 189r 0                $ Importance
-prdmp 10000000 10000000  -1     $ Print and dump cycle
-rand  seed=RNSEED               $ Random seed
-nps 10000000                    $ Number of particles
+prdmp 10000000 10000000  1      $ Print and dump cycle
+rand gen=2 seed=12345           $ Random seed
+nps 60000000                    $ Number of particles
 sdef par=p erg=1 pos=volumer    $ General source definition
 C ------------------------------------------------------------------------------
 C TALLIES FOR EACH ORGAN/TISSUE
 C ------------------------------------------------------------------------------
-read file=../phantoms/mrcp-am.tally
+read file=mrcp-am.tally
 C ------------------------------------------------------------------------------
 C MATERIAL DATA FOR EACH ORGAN/TISSUE
 C ------------------------------------------------------------------------------
-read file=../phantoms/mrcp-am.material
+read file=mrcp-am.material
 C ------------------------------------------------------------------------------
 C EMBED is required for embedding a mesh geometry into MCNP6 input
 embed2 meshgeo=abaqus                       $ Format specification
-       mgeoin=../phantoms/mrcp-am.inp       $ Name of the input file
+       mgeoin=mrcp-am.inp                   $ Name of the input file
+       hdf5file=mrcp-am.h5
        background=15000                     $ Cell number of the background
        matcell=    1   100     2   200     3   300     4   301     5   302
                    6   303     7   400     8   401     9   402    10   403
```

B Executing the MCNP Code

The Snow executions described in this report use OpenMPI 4.1.1 and the Rocinante executions use Cray MPICH 8.1.25, and both executables use OpenMP threading. Both HPCs use the Slurm queuing system to manage jobs submitted from their user base. A typical Snow Slurm queue submission script looks like that shown in Listing 3 for PCA and a typical Rocinante Slurm script looks like that shown in Listing 4 for the ICRP145 Male Human Phantom.

Listing 3: pca_kcode_sn_072.mcnp.inp.sbatch.txt

```
#!/bin/bash
#
#SBATCH --time=12:00:00
#SBATCH --nodes=2
#SBATCH --output=pca_kcode_sn_072.mcnp.inp.dayfile
#SBATCH --error=pca_kcode_sn_072.mcnp.inp.errfile

echo "Job ID: ${SLURM_JOBID}"
echo "Hostname: ‘hostname‘"
echo "Job  Started: " ‘date‘

cd ${SLURM_SUBMIT_DIR}
module load mcnp6/6.3
srun --nodes 2 --tasks-per-node 2 --cpus-per-task 18 \
  mcnp6.mpi tasks 18 n= pca_kcode_sn_072.mcnp.inp
sleep 2

echo "Job Finished: " ‘date‘
```

Listing 4: mrcp-am_internal_input_ro_224.mcnp.inp.sbatch.txt

```
#!/bin/bash
#
#SBATCH --time=12:00:00
#SBATCH --nodes=2
#SBATCH --output=mrcp-am_internal_input_ro_224.mcnp.inp.dayfile
#SBATCH --error=mrcp-am_internal_input_ro_224.mcnp.inp.errfile

echo "Job ID: ${SLURM_JOBID}"
echo "Hostname: ‘hostname‘"
echo "Job  Started: " ‘date‘

cd ${SLURM_SUBMIT_DIR}
module load mcnp6/6.3
srun --nodes 2 --tasks-per-node 8 --cpus-per-task 14 --cpu-bind=ldoms \
  mcnp6.mpi tasks 14 n= mrcp-am_internal_input_ro_224.mcnp.inp
sleep 2

echo "Job Finished: " ‘date‘
```

C Executable `ldd` Information

This appendix gives `ldd` traceability information on the two MPI executables used in this work, where the output for Snow is shown in Listing 5 and the output for Rocinante is shown in Listing 6.

Listing 5: `ldd mcnp6.mpi` (Snow)

```
linux-vdso.so.1 =>  (0x00007ffddcbd8000)
libSM.so.6 => /lib64/libSM.so.6 (0x00002ad3932ec000)
libICE.so.6 => /lib64/libICE.so.6 (0x00002ad3934f4000)
libX11.so.6 => /lib64/libX11.so.6 (0x00002ad393710000)
libXext.so.6 => /lib64/libXext.so.6 (0x00002ad393a4e000)
libiomp5.so => .../lib/libiomp5.so (0x00002ad393c60000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x00002ad39408c000)
libm.so.6 => /lib64/libm.so.6 (0x00002ad3942a8000)
libdl.so.2 => /lib64/libdl.so.2 (0x00002ad3945aa000)
libz.so.1 => .../lib/libz.so.1 (0x00002ad3947ae000)
libmpi.so.40 => /usr/projects/hpcsoft/toss3/snow/openmpi/4.1.1-gcc-4.8.5/lib/libmpi.
  so.40 (0x00002ad3949c5000)
libirng.so => .../lib/libirng.so (0x00002ad394eae000)
libstdc++.so.6 => .../lib/libstdc++.so.6 (0x00002ad39521a000)
libcilkrts.so.5 => .../lib/libcilkrts.so.5 (0x00002ad3955f4000)
libc.so.6 => /lib64/libc.so.6 (0x00002ad395836000)
/lib64/ld-linux-x86-64.so.2 (0x00002ad3930c8000)
libgcc_s.so.1 => .../lib/libgcc_s.so.1 (0x00002ad395c04000)
libuuid.so.1 => /lib64/libuuid.so.1 (0x00002ad395e1c000)
libxcb.so.1 => /lib64/libxcb.so.1 (0x00002ad396021000)
librt.so.1 => /lib64/librt.so.1 (0x00002ad396249000)
liblustreapi.so.1 => /lib64/liblustreapi.so.1 (0x00002ad396451000)
libpsm2.so.2 => /lib64/libpsm2.so.2 (0x00002ad396693000)
libpsm_infinipath.so.1 => /lib64/libpsm_infinipath.so.1 (0x00002ad3968fb000)
libopen-rte.so.40 => /usr/projects/hpcsoft/toss3/snow/openmpi/4.1.1-gcc-4.8.5/lib/
  libopen-rte.so.40 (0x00002ad396b51000)
libopen-orted-mpir.so => /usr/projects/hpcsoft/toss3/snow/openmpi/4.1.1-gcc-4.8.5/
  lib/libopen-orted-mpir.so (0x00002ad396e6d000)
libopen-pal.so.40 => /usr/projects/hpcsoft/toss3/snow/openmpi/4.1.1-gcc-4.8.5/lib/
  libopen-pal.so.40 (0x00002ad39706f000)
libxpmem.so.0 => /lib64/libxpmem.so.0 (0x00002ad3975e6000)
```

```
libfabric.so.1 => /lib64/libfabric.so.1 (0x00002ad3977e9000)
librdmacm.so.1 => /lib64/librdmacm.so.1 (0x00002ad397b4d000)
libibverbs.so.1 => /lib64/libibverbs.so.1 (0x00002ad397d64000)
libudev.so.1 => /lib64/libudev.so.1 (0x00002ad397f7d000)
libpciaccess.so.0 => /lib64/libpciaccess.so.0 (0x00002ad398193000)
libpmi2.so.0 => /lib64/libpmi2.so.0 (0x00002ad39839d000)
libpmi.so.0 => /lib64/libpmi.so.0 (0x00002ad3985b7000)
libutil.so.1 => /lib64/libutil.so.1 (0x00002ad3987be000)
libintlc.so.5 => .../lib/../lib/libintlc.so.5 (0x00002ad3989c1000)
libXau.so.6 => /lib64/libXau.so.6 (0x00002ad398c3c000)
liblnetconfig.so.4 => /lib64/liblnetconfig.so.4 (0x00002ad398e40000)
libyaml-0.so.2 => /lib64/libyaml-0.so.2 (0x00002ad399073000)
libreadline.so.6 => /lib64/libreadline.so.6 (0x00002ad399293000)
libnl-genl-3.so.200 => /lib64/libnl-genl-3.so.200 (0x00002ad3994d9000)
libnl-3.so.200 => /lib64/libnl-3.so.200 (0x00002ad3996df000)
libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00002ad399900000)
libnuma.so.1 => /lib64/libnuma.so.1 (0x00002ad399b04000)
libinfinipath.so.4 => /lib64/libinfinipath.so.4 (0x00002ad399d10000)
libnl-route-3.so.200 => /lib64/libnl-route-3.so.200 (0x00002ad399f1f000)
libcap.so.2 => /lib64/libcap.so.2 (0x00002ad39a18c000)
libdw.so.1 => /lib64/libdw.so.1 (0x00002ad39a391000)
libresolv.so.2 => /lib64/libresolv.so.2 (0x00002ad39a5e2000)
libslurm_pmi.so => /usr/lib64/slurm/libslurm_pmi.so (0x00002ad39a7fc000)
libtinfo.so.5 => /lib64/libtinfo.so.5 (0x00002ad39ac57000)
libattr.so.1 => /lib64/libattr.so.1 (0x00002ad39ae81000)
libelf.so.1 => /lib64/libelf.so.1 (0x00002ad39b086000)
liblzma.so.5 => /lib64/liblzma.so.5 (0x00002ad39b29e000)
libbz2.so.1 => /lib64/libbz2.so.1 (0x00002ad39b4c4000)
```

Listing 6: `ldd mcnp6.mpi` (Rocinante)

```
linux-vdso.so.1 (0x00007ffe37d7e000)
libSM.so.6 => /usr/lib64/libSM.so.6 (0x000014f54ffb0000)
libICE.so.6 => /usr/lib64/libICE.so.6 (0x000014f54fd93000)
libX11.so.6 => /usr/lib64/libX11.so.6 (0x000014f54fa52000)
libXext.so.6 => /usr/lib64/libXext.so.6 (0x000014f54f840000)
libsci_intel_mpi_mp.so.5 => /opt/cray/pe/lib64/libsci_intel_mpi_mp.so.5 (0
  x000014f54f001000)
libsci_intel_mp.so.5 => /opt/cray/pe/lib64/libsci_intel_mp.so.5 (0x000014f54b717000)
```

```
libiomp5.so => /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64/
  libiomp5.so (0x000014f54b2d1000)
libstdc++.so.6 => /usr/lib64/libstdc++.so.6 (0x000014f54aeae000)
libirng.so => /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64/
  libirng.so (0x000014f54ab97000)
libcilkrts.so.5 => /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64/
  libcilkrts.so.5 (0x000014f54a95a000)
libhdf5_parallel_intel.so.200 => /opt/cray/pe/lib64/libhdf5_parallel_intel.so.200
  (0x000014f54a3e9000)
libmpi_intel.so.12 => /opt/cray/pe/lib64/libmpi_intel.so.12 (0x000014f547859000)
libdl.so.2 => /lib64/libdl.so.2 (0x000014f547655000)
libxpmem.so.0 => /opt/cray/xpmem/default/lib64/libxpmem.so.0 (0x000014f5503bd000)
libimf.so => /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64/libimf.
  so (0x000014f54726b000)
libm.so.6 => /lib64/libm.so.6 (0x000014f546f20000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x000014f546cfd000)
libc.so.6 => /lib64/libc.so.6 (0x000014f546908000)
libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x000014f5466e9000)
libuuid.so.1 => /usr/lib64/libuuid.so.1 (0x000014f5503b2000)
libxcb.so.1 => /usr/lib64/libxcb.so.1 (0x000014f5464c0000)
libifcoremt.so.5 => /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64/
  libifcoremt.so.5 (0x000014f550236000)
libsvml.so => /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64/
  libsvml.so (0x000014f544e97000)
libintlc.so.5 => /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64/
  libintlc.so.5 (0x000014f544e20000)
/lib64/ld-linux-x86-64.so.2 (0x000014f5501b8000)
librt.so.1 => /lib64/librt.so.1 (0x000014f544c17000)
libz.so.1 => /lib64/libz.so.1 (0x000014f544a00000)
libfabric.so.1 => /opt/cray/libfabric/1.15.2.0/lib64/libfabric.so.1 (0
  x000014f54470d000)
libatomic.so.1 => /usr/lib64/libatomic.so.1 (0x000014f544504000)
libpmi.so.0 => /opt/cray/pe/lib64/libpmi.so.0 (0x000014f544302000)
libpmi2.so.0 => /opt/cray/pe/lib64/libpmi2.so.0 (0x000014f5440e0000)
libifport.so.5 => /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64/
  libifport.so.5 (0x000014f55020b000)
libXau.so.6 => /usr/lib64/libXau.so.6 (0x000014f543edc000)
```

```
libcxi.so.1 => /usr/lib64/libcxi.so.1 (0x000014f543cb7000)
libcurl.so.4 => /usr/lib64/libcurl.so.4 (0x000014f543c18000)
libjson-c.so.3 => /usr/lib64/libjson-c.so.3 (0x000014f543a08000)
libpals.so.0 => /opt/cray/pe/lib64/libpals.so.0 (0x000014f543800000)
libnghttp2.so.14 => /usr/lib64/libnghttp2.so.14 (0x000014f5435d8000)
libidn2.so.0 => /usr/lib64/libidn2.so.0 (0x000014f5433bb000)
libssh.so.4 => /usr/lib64/libssh.so.4 (0x000014f54314d000)
libpsl.so.5 => /usr/lib64/libpsl.so.5 (0x000014f542f3b000)
libssl.so.1.1 => /usr/lib64/libssl.so.1.1 (0x000014f542e9c000)
libcrypto.so.1.1 => /usr/lib64/libcrypto.so.1.1 (0x000014f542b5c000)
libgssapi_krb5.so.2 => /usr/lib64/libgssapi_krb5.so.2 (0x000014f54290a000)
libldap_r-2.4.so.2 => /usr/lib64/libldap_r-2.4.so.2 (0x000014f5426b6000)
liblber-2.4.so.2 => /usr/lib64/liblber-2.4.so.2 (0x000014f5424a7000)
libzstd.so.1 => /usr/lib64/libzstd.so.1 (0x000014f542177000)
libbrotlidec.so.1 => /usr/lib64/libbrotlidec.so.1 (0x000014f541f6b000)
libjansson.so.4 => /usr/lib64/libjansson.so.4 (0x000014f541d5d000)
libunistring.so.2 => /usr/lib64/libunistring.so.2 (0x000014f5419da000)
libjitterentropy.so.3 => /usr/lib64/libjitterentropy.so.3 (0x000014f5417d3000)
libkrb5.so.3 => /usr/lib64/libkrb5.so.3 (0x000014f5414fa000)
libk5crypto.so.3 => /usr/lib64/libk5crypto.so.3 (0x000014f5412e2000)
libcom_err.so.2 => /lib64/libcom_err.so.2 (0x000014f5410de000)
libkrb5support.so.0 => /usr/lib64/libkrb5support.so.0 (0x000014f540ecf000)
libresolv.so.2 => /lib64/libresolv.so.2 (0x000014f540cb7000)
libsasl2.so.3 => /usr/lib64/libsasl2.so.3 (0x000014f540a9a000)
libbrotlicommon.so.1 => /usr/lib64/libbrotlicommon.so.1 (0x000014f540879000)
libkeyutils.so.1 => /usr/lib64/libkeyutils.so.1 (0x000014f540674000)
libselinux.so.1 => /lib64/libselinux.so.1 (0x000014f54044b000)
libpcre.so.1 => /usr/lib64/libpcre.so.1 (0x000014f5401c2000)
```