MCNP6 Optimization and Testing for Criticality Safety Calculations

Forrest B. Brown

Los Alamos National Laboratory, PO Box 1663, Los Alamos, NM, 87544, fbrown@lanl.gov

INTRODUCTION

The MCNP6.1.1 [1] Monte Carlo code was released in 2014. It is an update to the MCNP6.1 [2] code released in 2013. The MCNP6.1.1 coding and algorithms were optimized to provide significant improvements in performance for criticality safety applications. This paper reports the performance testing results obtained for MCNP5-1.60 [3], MCNP6.1, and MCNP6.1.1 on several modern computer platforms for a suite of ICSBEP criticality benchmark calculations. The comparisons provide useful guidance to criticality safety analysts on the benefits of using the latest version MCNP6.1.1 on newer computers.

BACKGROUND

The MCNP6.1 Monte Carlo code, released in 2013, provides all of the standard methodology for criticality calculations available in the previous MCNP5-1.60 code and was thoroughly verified against MCNP5-1.60 [4]. During 2013, it was evident that MCNP6.1 was slower than MCNP5-1.60, typically by 20-30%, but sometimes by factors of 2-5x. Assessment of the code led to a plan for improving the performance and structure of MCNP6.1. The initial performance improvements, described in [5], were incorporated into the 2014 update, MCNP6.1.1. MCNP6.1.1 was also thoroughly verified to ensure correct results for criticality safety applications [6].

On the particular computer system used for the code optimization, MCNP6.1.1 demonstrated speedups by factors of 1.2x - 4x compared to MCNP6.1, depending on the type of problem. For criticality problems, MCNP6.1.1 is typically 1.5x - 1.7x faster than MCNP6.1 and 1.2x - 1.3x faster than MCNP5-1.60.

MCNP6 CODE OPTIMIZATION

Reference [5] provides details on the coding and algorithm optimizations that were incorporated into MCNP6.1.1, so only a brief summary is provided here. The optimization effort included both “classic code optimizations” and improvements to algorithms. The classic code optimizations involve well-known practices for speeding up short, localized sections of coding, such as: compiler options, eliminating vector operations on noncontiguous data, inlining heavily used functions such as binary searches, using guarding if-statements to avoid unnecessary function calls, using thread-private common blocks instead of declaring individual variables thread-private, etc. Classic coding optimizations typically provide small speedups of 5-30%. Larger speedups can be obtained from algorithm improvements. A new hash-based energy lookup algorithm [7] developed for MCNP6.1.1 provided speedups of 15x – 20x for the portion of MCNP6.1.1 that calculates macroscopic cross-sections (which can consume 1/3 – 2/3 of the overall runtime). Other minor algorithm improvements were also included, such as buffering fission bank entries and improved rejection schemes in collision physics. Table I provides a summary of test problems used in the code optimization, and Table II provides the resulting speedups for MCNP6.1.1 vs. MCNP6.1. The speedups for criticality problems are 1.14x – 2.20x. These results were obtained on one particular computer system (2010 Mac Pro with 2 quad-core 3 GHz Xeon processors), for problems that are not necessarily representative of day-to-day criticality safety calculations.

CRITICALITY SUITE TESTING

To provide performance results specifically for routine criticality safety applications, one of the standard criticality benchmark suites from the MCNP distribution package was used, the “Criticality Validation Suite” [8] consisting of 31 problems from the ICSBEP Handbook [9], using the ENDF/B-VII.0 nuclear data libraries. This suite was also used as part of the verification testing reported in [4] and [6]. Collectively for all 31 problems, the suite involves 38.25 M neutron histories, 1.1 GB of cross-section data I/O, 1.1 GB of dumpfile I/O, and 125 K lines of printed output.

Table III presents the wall-clock time required to run the 31 benchmark problems on various computer platforms. For all systems, it can be seen that MCNP6.1.1 is significantly faster than MCNP6.1, completing the benchmark problems in about 2/3 of the time (i.e., running about 50% more neutron histories per minute). Running this suite on an office computer took 4-8 hours a decade ago, an hour or less 5 years ago, and 15 minutes or less today on the newest systems. As can be seen in Table III, the overall speedups are due to more processor cores, not faster processors. Because a Monte Carlo code like MCNP makes extreme demands on memory, with repeated random access to retrieve small amounts of data, the faster memory bandwidth in newer systems also significantly improves performance. However, the timings for the latest Mac Pro 2014 indicate an emergent issue for future work – with 24 hyperthreads sharing the same L3 cache, memory access contention limits the performance for more than about 15 hyperthreads.

Table I. Test Problems for Performance Improvements

Criticality Problems	
ks1	3D PWR, OECD perf. bench., Kord Smith, 60 isos, no tallies
ks2	ks1, with 10 isotopes, no tallies
ks3	ks1, with 10 isotopes, fmesh tallies
ks4	ks1, 60 isotopes, fmesh tallies
baw1	BAWXI2 ICSBEP problem, 31 isotopes, no tallies
baw2	BAWXI2 ICSBEP problem, 31 isotopes, fmesh tally
fvf	fuel storage vault, from OECD source convergence
g1	Godiva problem, 3 isotopes
g2	Godiva problem, 423 isotopes
pin	AECL pin cell, with FPs, 147 isotopes
Fixed-source Problems	
void1	ks1, with VOID card & no tallies
void2	baw1, with VOID card & no tallies
void3	fvf, with VOID card & no tallies
det1	3D well-log, neutrons, weight windows, F4 tallies
med1	medical physics, modified 3D Zubal head, photons
pht1	pulse-height tally test, cylindrical problem, photons

Problems run on Mac Pro (3 GHz Xeon), 8 threads, Intel 12.0 Fortran

Table II. Speedups for MCNP6.1.1 vs. MCNP6.1

Criticality Problems	
ks1	1.76
ks2	2.13
ks3	1.35
ks4	1.36
baw1	2.19
baw2	1.59
fvf	2.04
g1	1.14
g2	2.20
pin	1.73
Fixed-source Problems	
void1	3.03
void2	4.11
void3	2.72
det1	1.67
med1	1.15
pht1	1.22

Table III. Overall Run Times for Criticality Validation Suite on Various Computer Systems

Computer	CPU Speed (GHz)	Mem. Speed (GHz)	processors	cores per processor	hyperthreads per core	MCNP threads used	MCNP Version	Total Time (min)
MacBook 2010	2.7	1.1	1 - i7	2	2	4	mcnp6.1.1	88.0
MacBook 2013	3.0	1.6	1 - i7	2	2	4	mcnp5-1.60	39.9
						4	mcnp6.1	62.0
						4	mcnp6.1.1	41.7
Mac Pro 2010	3.0	0.67	2 - Xeon	4	-	8	mcnp5-1.60	30.1
						8	mcnp6.1	43.6
						8	mcnp6.1.1	28.3
Windows 2012	2.7	1.3	2 - Xeon	6	-	10	mcnp6.1.1	19.2
						12	mcnp6.1.1	27.1
Mac Pro 2012	2.4	1.07	2 - Xeon	4	2	16	mcnp5-1.60	24.5
						16	mcnp6.1	32.4
						4	mcnp6.1.1	41.8
						8	mcnp6.1.1	24.2
						16	mcnp6.1.1	22.3
Mac Pro 2014	2.7	1.6	1 - Xeon	12	2	12	mcnp5-1.60	13.9
						12	mcnp6.1	19.9
						4	mcnp6.1.1	27.8
						8	mcnp6.1.1	15.9
						12	mcnp6.1.1	13.5
						14	mcnp6.1.1	11.7
						16	mcnp6.1.1	13.0

SUMMARY

The performance gains reported in this work demonstrate that the initial efforts to improve the performance and structural foundation of MCNP6 have succeeded. Many more such improvements are planned over the next few years to address parallel threading efficiency; cache and memory access improvements; new techniques for performing and storing tally information; improved coding clarity, robustness, and compliance with standards; parallel MPI improvements for clusters; etc.

A faster Monte Carlo code has direct benefits to the overall quality of criticality safety analyses, by enabling analysts to run more problems and reduce the Monte Carlo statistical uncertainty. The highest priority for development, however, goes to maintaining and improving the physics accuracy of the code so that accurate and reliable results are produced.

ACKNOWLEDGMENTS

This work was supported by the US DOE/NNSA Nuclear Criticality Safety Program.

REFERENCES

1. J.T. Goorley, "MCNP6.1.1-Beta Release Notes," LA-UR-14-24680 (2014).
2. J.T. Goorley, et al., "Initial MCNP6 Release Overview - MCNP6 version 1.0," LA-UR-13-22934 (2013).
3. F.B. Brown, B.C. Kiedrowski, J.S. Bull, "MCNP5-1.60 Release Notes," LA-UR-10-06235 (2010).
4. F.B. Brown, B.C. Kiedrowski, J.S. Bull, "Verification of MCNP5-1.60 and MCNP6.1 for Criticality Safety Applications," LA-UR-13-22196 (2013).
5. F.B. Brown, "MCNP6 Monte Carlo Code Optimization", *Proc. ANS MC2015*, Nashville, TN, April 19-23 (2015).
6. F.B. Brown, B.C. Kiedrowski, J.S. Bull, "Verification of MCNP6.1 and MCNP6.1.1 for Criticality Safety Applications," LA-UR-13-22196 (2014).
7. F.B. Brown, "New Hash-based Energy Lookup Algorithm for Monte Carlo Codes," *Trans.Am. Nucl. Soc*. **111** (2014).
8. Russell D. Mosteller, "Validation Suites for MCNP," *Proc. of ANS RPSD-12*, April 14-17, Santa Fe, NM (2002).
9. International Handbook of Evaluated Criticality Safety Benchmark Experiments, NEA/NSC/DOC(95)03, OECD Nuclear Energy Agency (2007).