REVIEW

Recent Advances and Future Prospects for Monte Carlo

Forrest B. BROWN*

Los Alamos National Laboratory, Los Alamos, NM, 87545, USA

The history of Monte Carlo methods is closely linked to that of computers: The first known Monte Carlo program was written in 1947 for the ENIAC; a pre-release of the first Fortran compiler was used for Monte Carlo in 1957; Monte Carlo codes were adapted to vector computers in the 1980s, clusters and parallel computers in the 1990s, and teraflop systems in the 2000s. Recent advances include hierarchical parallelism, combining threaded calculations on multicore processors with message-passing among different nodes. With the advances in computing, Monte Carlo codes have evolved with new capabilities and new ways of use. Production codes such as MCNP, MVP, MONK, TRIPOLI, MCU, and KENO are now 20-30 years old (or more) and are very rich in advanced features. The former "method of last resort" has now become the first choice for many applications. Calculations are now routinely performed on office computers, not just on supercomputers. Current research and development efforts are investigating the use of Monte Carlo methods on FPGAs, GPUs, and many-core processors. Other far-reaching research is exploring ways to adapt Monte Carlo methods to future exaflop systems that may have 1M or more concurrent computational processes.

KEYWORDS: parallel, reactors, supercomputing, exaflop systems

I. The First 60 Years of Monte Carlo Codes

The history of Monte Carlo methods is closely linked to that of computers:

The name itself comes from the Manhattan Project in 1945.[1)] The first electronic computer – the ENIAC – was nearing completion and was the subject of much discussion among Stan Ulam, John von Neumann, Nicholas Metropolis, and others. The proposal to use the ENIAC for the statistical method led Metropolis to suggest the name "Monte Carlo."

During 1947, von Neumann[2)] developed the first known computer program for performing Monte Carlo calculations. The description of the program logic is remarkably similar to what is still done today in modern Monte Carlo codes. (In fact, portions of MCNP coding appear to have originated from von Neumann's program, migrated through several generations of assembly language and Fortran.) In describing the program, von Neumann estimated that 100 neutrons with 100 collisions would take 5 hours on the ENIAC (with its 18,000 vacuum tubes). Today that would take milliseconds. The first symposium on the Monte Carlo method was held in 1949.

The first Monte Carlo codes were written in the 1940-50s in very low-level languages (e.g., machine language, assembler, FLOCO, etc.). In 1957, a pre-release of the first Fortran compiler developed at IBM arrived at the Bettis laboratory[3)] (2,000 punched cards for a binary image of the compiler). The very first program tested failed due to a program syntax error, perhaps the first case of user frustration and debugging with Fortran, but then successfully compiled and executed on an IBM 704. Lew Ondis began using the new compiler for Monte Carlo codes.

During the 1960-80s, extensive capabilities were added to Monte Carlo codes running on mainframe computers (e.g., CDC-6600, CDC-7600, Cray-1, etc.), often involving drums, extended core memory devices, disks, and punched cards. In the 1980s, a few codes were developed for vector computers[4,5,6)] and then extended to parallel/vector computers such as the Cray-XMP, -YMP, -C90, -J90 series. During the 1990s, parallel Monte Carlo became commonplace due to the arrival of workstation clusters and modestly priced parallel systems. Early parallel computations typically used 4 or 8 processors at first, and then dozens of processors connected via PVM or MPI message-passing. During the 2000s, Monte Carlo parallelism was extended to 100s and 1000s of processors.

With the arrival of multicore processors in the 2000s, some Monte Carlo codes were adapted to use threaded parallelism, using pthreads, OpenMP, or vendor-specific pragmas. With care, local parallelism using threading on multicore processors can be combined with global parallelism using message-passing among nodes.[7)] At a higher level, independent Monte Carlo jobs can be run concurrently, combining results later (job parallelism). Such hierarchical parallelism – parallel jobs, message-passing, multicore threading – is necessary to fully utilize today's largest teraflop and petaflop computing systems. Monte Carlo calculations have been run using 10,000s of processors.

Perhaps more significant is the tremendous growth in power of office computers and laptops. These have progressed from Kflops in the 1990s to Mflops in the 2000s, to Gflops today, with multicore processors in all current laptops and office computers. Today's laptops outperform supercomputers of the 1990s, so that Monte Carlo codes can now be easily used by anyone. This step change in accessibility has led to routine use of Monte Carlo by students,

* Corresponding author, Email: fbrown@lanl.gov

researchers, and engineers.

The tremendous growth in computer capabilities has facilitated the development of new features and techniques in Monte Carlo codes. Over the past 63 years of methods and code development for Monte Carlo, there has been steady progress in all areas: Geometry modeling capabilities began with a dozen or fewer regions, then grew to a few 1000s in the 1970s, 10-100K in the 1990s, and now to millions or more in some codes. The introduction of hierarchical geometry (i.e., embedded geometry, "holes", "universes", etc.) has enabled large models with reusable geometry parts. Physics interaction data as provided by libraries of nuclear cross-sections has likewise grown from nonexistent to the current ENDF/B-VII libraries that can require gigabytes of storage.

The past 63 years of Monte Carlo history represent a huge investment in the research and development of random number generators, random sampling methods, criticality calculation methods, variance reduction techniques, physics interaction modeling, geometric ray-tracing, parallel computing methods, physics interaction data libraries, and code development. Of even more significance is the very large effort devoted to the verification and validation of Monte Carlo codes. As a result of the huge investment of resources, a number of long-lived and respected Monte Carlo codes are available: MCNP,[7)] MVP,[8)] KENO,[9)] TRIPOLI,[10)] MONK,[11)] MCU,[12)] and many others. These codes are 20-30 years old (or more) and have extensive modeling and analysis capabilities that have undergone many years of use and testing. They represent a formidable level of capabilities that new or future codes must surpass to achieve widespread, general-purpose use.

The combination of realistic and detailed geometry and cross-sections, essentially modeling everything with high precision, has motivated more users to apply Monte Carlo to their problems. It is common today for analysts to run billions of particles for a problem, rather than the millions or thousands of a decade ago.

II. The Next Years of Monte Carlo Codes

In the previous section, the maturity and rich list of features in today's most-used Monte Carlo codes were cited as distinct benefits of the past 63 years of Monte Carlo development. Extending those Monte Carlo codes with ever more features over the years has a cost, however. As software packages become larger, the burden for maintenance, documentation, user support, and verification and validation grows excessively. After 20-30 years of adding new features, new development is impeded by older code structure and logic, obsolete data structures, and coding style that often resembles assembly language. The algorithms and data structures in many mature codes were first developed for problems with only a dozen or so regions and materials. While the codes have been (painfully) modified over the years to now handle many 1000s of materials and regions, extension to millions or billions of materials and regions is difficult. Rewriting the older, mature Monte Carlo codes in a modern form without sacrificing existing capabilities is an expensive endeavor,[13)] seldom accomplished. For effective use on today's large HPC systems, many new Monte Carlo codes are being developed to take full advantage of the new architectures. Such codes as SERPENT, MC21, McCARD, MCP, Monaco, MVP-2, and many others are newer and more readily adapted to large-scale applications. In the near future, these codes will most likely focus on the hierarchical parallelism developed in the 2000s, with threading on multicore nodes and MPI message-passing among nodes. Currently available HPC systems support up to 16 threads per node, with 1,000s of nodes. Within the next few years, the number of threads per node will rise dramatically – initial testing is underway for manycore processors with 32, 48, 80 and more cores.

An alternative approach to clusters of multicore or manycore nodes is heterogeneous computing systems, with some specialized hardware processors included either independently or attached to conventional cpus. Examples include FPGAs, GPUs, GPGPUs, and Cell processors. This type of special purpose processor was originally used solely for graphics support and visualization, and typically utilizes pipelined or vector functional units, with additional parallelism from replicated functional units. The capabilities of these GPGPU processors have grown rapidly in the last few years, leading to significant gains in speed if a calculation can be cast in vector form. Since vectorization of Monte Carlo was proven 30 years ago,[4,5)] it would appear straightforward. In practice, however, existing mature codes would have to be entirely restructured and largely rewritten to take advantage of vectorization; that is prohibitively expensive and would invalidate many years of verification and validation work. New or future Monte Carlo codes show the most promise for utilizing GPGPUs, since algorithms and data structures are flexible when developing new codes. Some of these efforts are showing excellent results.[14)] It should be noted that determining the effectiveness of GPGPUs in speeding up Monte Carlo is difficult. Often the GPGPU timings are compared to single-threaded runs on a conventional processor, when they should more realistically be compared to threaded calculations.

In addition to Monte Carlo development to leverage the advances in computing, there is an abundance of methods development in progress, enabled by the higher speeds and larger memories. These efforts include: (1) New adjoint-weighted tally schemes for continuous-energy Monte Carlo criticality calculations,[15)] permitting correct calculation of reactor kinetics parameters, perturbations in reaction rates, and sensitivity-uncertainty parameters. (2) New iteration methods for criticality calculations, such as Wielandt's method,[16)] that may accelerate convergence and eliminate the underprediction bias in uncertainties. (3) On-the-fly Doppler broadening of neutron cross-sections,[17)] to finally permit a continuous distribution of material temperatures. (4) New types of tallies, such as Kernel Density Estimators (KDE)[18)] and functional expansion tallies (FET),[19)] that permit continuous variation over regions, rather than just simple averages. (5) Depletion analysis of fuel assemblies and reactors,[20,21)] including equilibrium Xenon and control searches. (6) All-particle, all-energy Monte Carlo codes.[22)]

(7) Stochastic geometry[23] for modeling the random locations of fuel particles in newer reactor fuel systems. (8) Improved treatment of the free-gas scattering model at epithermal neutron energies[24] to include important resonance scattering effects. (9) Coupled calculations involving both Monte Carlo and deterministic transport codes[25] to enable more effective variance reduction. There are many more such efforts in progress.

III. Future Prospects for Monte Carlo Codes

High performance computing systems have evolved from Mflops in the 1980s, to Gflops in the 1990s, to Tflops in the 2000s, and now to Pflops in the 2010s. Each jump in performance was enabled by a combination of advances in integrated circuit design and fabrication, changes in system architecture, and new software technology. There are currently intense efforts[26] to continue the trend and provide exaflop performance by 2020. While specific system characteristics and architecture for exaflop systems are still under investigation, several broad predictions can be made:

(1) There will be massive parallelism, with 1-100 M cpu-cores. None of the scientific and engineering software in use today can be scaled to such extreme processor counts. The only way to fully utilize the massive parallelism on exaflop systems with today's software is to run many 1,000s or more parallel jobs. This approach to exascale computing would support parameter studies, where very large numbers of jobs are run with different combinations of code input parameters to span the phase space of a multidimensional problem. It would also support uncertainty quantification analysis, where very many individual code input parameters are varied in separate calculations to assess the sensitivity of problem results to uncertainties in the code input parameters and then estimate the overall uncertainty on calculation of a physical problem. The running of many 1,000s of parallel jobs, however, would not impact the running time of a single job and would do little for the day to day work of most engineers and scientists. To take advantage of exaflop computing power for single jobs, changes in today's software are required.

(2) One of the key considerations in designing future exaflop systems is greatly reducing the power consumption per compute node. (With today's technology, an exaflop system might consume as much as 500 MW, clearly prohibitive.) Large reductions in power requirements imply that future cpu clock cycles may be little different from those of today's cpus, and that the amount of memory per cpu-core may even be smaller than today. The coming limits on or reduction in memory per cpu-core will have a huge impact on software development. (As a current example, consider MCNP5 and the first IBM BlueGene systems. Even though MCNP5 supports hierarchical parallelism at the job, message-passing, and threading levels, and has been run with 10Ks of cpu-cores, it was unable to run on the first BlueGene systems due to the very limited amount of memory per cpu-core.) Changes in software are needed to accommodate the expected reduction in memory per cpu-core. Large datasets required for a calculation will need to be distributed across many nodes, and codes will need to remotely access the distributed data. Latencies imposed by remote data access will need to be masked by executing even more threads than there are cpu-cores (much like the hyperthreading done today on single cores), leading to even more parallel processes.

(3) Distributing large datasets across nodes and remotely accessing the data has even further implications for code developers. While many large datasets are read-only, such as material properties, nuclear cross-sections, and geometry descriptions, other datasets require read-write access. For Monte Carlo codes in particular, the tally data may comprise a very large dataset that needs to be accessed by all individual particle history compute nodes. In today's codes tally data is retrieved from memory, a particle's contribution is added to the data, and then the data is stored back into memory. The particular tally location being modified must be locked (or replicated) to permit only one particle history to modify it at a time. With very large distributed datasets for tallies, a different approach is needed. Tally data for a particle needs to be transmitted to a remote node where a section of the shared tally dataset resides, and a remote operation needs to be performed. That is, while the tally contribution is computed locally, updating the overall tally dataset is done remotely. The remote node holding a section of the tally dataset is then responsible for locking the proper data, performing the arithmetic to update the overall tally, and then unlocking that portion of the remote data. Thus the particle history portion of a Monte Carlo code may be performed locally, but material and geometry properties will be fetched from remote nodes, and results must be tallied by remote add-to-memory operations.

It may be argued that much of the discussion above regarding modified Monte Carlo algorithms could be avoided by using domain decomposition methods, where problem spatial domains are divided among processor nodes, and particles move between nodes. While such approaches have had limited success in the past 10 years, they are doomed to fail on exaflop systems. The inherent parallelism in Monte Carlo calculations is on particles, not spatial domains. The particles travel where they will, based on the physics and geometry of a problem. With millions or billions of spatial domains, a very large fraction of the domains would contain no particles and remain idle. Parallel efficiency would approach zero. Load balancing techniques would likewise not be effective since they require global communications and then global redistribution of the particle workload – a serious problem for a system with millions or billions of cpu-cores.

The future of Monte Carlo for the largest problems on exaflop systems may well involve "Monte Carlo swarms" - a very large number of light-weight Monte Carlo kernels with minimal memory requirements, remotely accessing distributed data via remote memory fetches, and performing tallies using remote add-to-memory operations. In this scheme, the problem data are decomposed and distributed, but rather than assigning particles to compute nodes based on their spatial coordinates, parallelism on particles is achieved by having each compute node retrieve geometry and cross-section data remotely from other nodes only as needed. This approach ensures that each compute node follows the same number of particles and thus performs approximately the same amount of work, so that scaling to very large numbers of processors is effective. There is no overhead in moving particles among processors, and the mapping of the particle and data processes onto the computing nodes is very flexible. While individual particle histories may take longer due to delays from remote access to data, the almost unlimited number of processors available provides massive parallelism to achieve overall large speedups.

This proposed approach to Monte Carlo on future exaflop computing systems represents a new paradigm for parallel Monte Carlo. While the basic coding for Monte Carlo particle histories can remain largely unchanged in existing codes, calls to remote data access routines must be inserted at the proper points to remotely fetch needed data and then perform local operations with that data. In principle, the codes could monitor in real-time the portions of the remote datasets they have been accessing most frequently and learn from that information, to subsequently prefetch data in an anticipatory fashion.

Fortunately, exaflop systems are 8-10 years in the future, so that there is sufficient time to investigate new approaches to massive parallelism for Monte Carlo codes. It must be stressed that the time to investigate such alternate approaches is now, not 8-10 years from now, so that alternate schemes will be available when the exaflop systems arrive. Some work has already begun.[27)]

References

1) N. Metropolis, "The Beginning of the Monte Carlo method," *Los Alamos Science*, Special Issue 1987 (1987).
2) R. D. Richtmyer, J. von Neumann, *Statistical Methods in Neutron Diffusion*, LAMS-551, Los Alamos National Laboratory (LANL) (1947).
3) H. Bright, "FORTRAN Comes to Westinghouse-Bettis, 1957", *IEEE Ann. Hist. Comput.,* **1**[1], 72-74 (1979).
4) F. B. Brown, W. R. Martin, "Monte Carlo Methods for Vector Computers," *J. Prog. Nucl. Energy,* **14**[3], 269-299 (1984).
5) W. R. Martin and F. B. Brown, "Present Status of Vectorized Monte Carlo for Particle Transport Analysis," *Int. J. Supercomput. Appl.*, **1**[2], 11-32 (1987).
6) M. Nakagawa, T. Mori, M. Sasaki, "Monte Carlo Calculations on Vector Supercomputers using GMVP," *Prog. Nucl. Energy*, **24**, 183 (1990).
7) X-5 Monte Carlo Team, *MCNP – A General N-Particle Transport Code, Version 5 – Volume I: Overview and Theory*, LA-UR-03-1987, Los Alamos National Laboratory (LANL) (2003).
8) Y. Nagaya, T. Mori, K. Okumura, M. Nakagawa, *MVP/GMVP II: General Purpose Monte Carlo Codes for Neutron and Photon Transport Calculations*, JAERI-1348, Japan Atomic Energy Research Institute (JAERI) (2005).
9) L.M. Petrie, N.F. Cross, *KENO-IV – An Improved Monte Carlo Criticality Program*, ORNL-4938, Oak Ridge National Laboratory (ORNL) (1975).
10) Tripoli-4 Project Team, *TRIPOLI-4 Version 4 User Guide*, CEA-R-6169 (2008).
11) M. Armishaw, A. J. Cooper, "Current Status and Future Direction of the MONK Software Package," *Proc. ICNC-2007*, St. Petersburg, Russia (2007).
12) E. A. Gomin, L. V. Maiorov, "The MCU Monte Carlo Code for 3-D Depletion Calculations," *Proc. M&C-1999*, Madrid, Spain (1999).
13) F. B. Brown, "MCNP5 Development, Verification, and Performance," *Proc. SNA-2003*, Paris, France, Sept. 22-24 (2003).
14) J. Tickner, "Monte Carlo Simulation of X-ray and Gamma-ray Photon Transport on a Graphics-Processing Unit," *Comput. Phys. Comm.,* doi:10.1016/j.cpc.2010.07.001 (2010).
15) B. C. Kiedrowski, F. B. Brown, P. Wilson, "Calculating Kinetics Parameters and Reactivity Changes with Continuous-Energy Monte Carlo," *Proc. PHYSOR-2010*, Pittsburgh, PA (2009).
16) F. B. Brown, "Wielandt Acceleration for MCNP5 Monte Carlo Eigenvalue Calculations," *Proc. M&C+SNA-2007*, Monterey, CA, April 15-19, 2007 (2007).
17) G. Yesilyurt, W. R. Martin, F. B. Brown, "On-The-Fly Doppler Broadening for Monte Carlo Codes," *Proc. M&C-2009*, Saratoga Springs, NY, May 3-7 (2009).
18) K. Banerjee, W. R. Martin, "Kernel Density Estimation Method for Monte Carlo Tallies with Unbounded Variance," *Trans. Am. Nucl. Soc.,* **101** (2009).
19) D. P. Griesheimer, *Functional Expansion Tallies for Monte Carlo Simulations*, Ph.D dissertation, University of Michigan (2005).
20) J. Leppänen, *PSG2/Serpent – A Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code, User's Manual*, VTT Technical Research Centre of Finland (2009).
21) T. M. Sutton *et al.*, "The MC21 Monte Carlo Transport Code," *Proc. M&C+SNA-2007*, Monterey, CA, April 15-19 (2007).
22) F. B. Brown *et al.*, "Reactor Physics Calculations with MCNP5 and the Status of MCNP6," workshop for *PHYSOR-2010*, Pittsburgh, PA, May (2010) [available at URL: http://mcnp-green.lanl.gov/publication/pdf/la-ur-10-02762_physor2010_workshop.pdf]
23) F. B. Brown, W. R. Martin, "Stochastic Geometry Capability in MCNP5 for the Analysis of Particle Fuel," *Ann. Nucl. Energy*, **31**[17], 2039-2047 (2004).
24) B. Becker, R. Dagan, G. Lohnert, "Proof and Implementation of the Stochastic Formula for Ideal Gas, Energy Dependent Scattering Kernel," *Ann. Nucl. Energy,* **36**, 470-474 (2009).
25) D. E. Peplow, J. C. Wagner, "Automated Variance Reduction for SCALE Shielding Calculations," *Proc. RPSD-2006*, 556-558, Carlsbad, New Mexico (2006).
26) See conference proceedings from the past few years, for example: URL http://www.lanl.gov/orgs/hpc/salishan/
27) P. Romano, B. Forget, F. Brown, "Towards Scalable Parallelism in Monte Carlo Particle Transport Codes Using Remote Memory Access," *Proc. SNA + MC2010,* Tokyo, Japan, October 17-20 (2010).