An embarassingly parallel framework for running EGSnrc/BEAMnrc/DOSXYZnrc, FLUKA, MCNP/MCNPX, GEANT4 and PENELOPE on grid and cluster computers

PW Chin*[1], JP Giddy[2], DG Lewis[1], DW Walker[2]

(1) Velindre Cancer Centre, Cardiff, Wales, UK
(2) Welsh e-Science Centre, Cardiff University, Wales, UK

We present a computing framework for Monte Carlo simulation of radiation transport. In conformity with our belief that Monte Carlo studies should be anchored to physics but not to any particular code, we have enabled different Monte Carlo codes in this environment. The framework embraces the EGSnrc family, FLUKA, MCNP/MCNPX, GEANT4 and PENELOPE; dedicated and non-dedicated resources; shared and non-shared filesystems; local clusters and nationsl grids; Unix/Linux family as well as Windows operating systems. The framework works whether or not the executing node has been pre-installed with the Monte Carlo code. It works with resource brokers such as Condor, Nimrod, PBS, SGE, OMII GridSAM and gLite. Validation and quality assurance become very important when using heterogeneous computers beyond local administration -- particularly if simulation results are used for making decisions which potentially save or harm human life e.g. patients seeking medical treatment or staff exposed to radiation hazards. We describe the steps we take before and/or after simulation execution to gain confidence in simulation results. Before execution of the full simulation, smaller test jobs may be run so that results may be validated against known values. This exercise is similar to the test suites available in MCNP/MCNPX, sometimes replaced by or coupled with problem-specific tests designed by the user. After execution of the simulation, results from different batches of radiation histories are compared for statistical agreement. A series of illustrative examples of the use of this computational framework will be presented, involving inter-code comparisons for medical physics applications.

Introduction

In medical physics, commonly used Monte Carlo codes for computer simulation of radiation transport include:
- EGSnrc [1], BEAMnrc [2] and DOSXYZnrc [3] - developed by the National Research Council of Canada.
- FLUKA [4, 5] - developed by the Italian National Institute for Nuclear Physics (INFN) and European Organisation for Nuclear Research (CERN).
- MCNP [6] and MCNPX [7] - developed by Los Alamos National Laboratory (LANL).
- GEANT4 [8] - developed by European Organisation for Nuclear Research (CERN).
- PENELOPE [9] - developed by Universitat de Barcelona.

While the above codes differ by particle types, physics processes, cross sections, variance reduction techniques, algorithms, program design and implementation, etc., two assumptions common to these codes are:
- particles from a radiation source interact with matter but *not* with each other; and
- radiation histories do *not* perturb each other.

Both these assumptions provide very good approximations for most purposes, notable exceptions including astrophysical problems and situations where the beam is extremely intense.

A cluster computer is a group of machines, interconnected by a network, that cooperate as distinct entities, but typically have the same architecture (processor type, operating system), infrastructure (filesystem, networking) and administrative policies (user accounts). A grid is a collection of machines typically spread across multiple administrative domains, and with different heterogeneous architecture and infrastructure. Due to their homogeneity, clusters are typically simpler to use. However, grids can provide access to more resources, and hence better availability than clusters.

Two main approaches exist to reduce turn-around time using clusters and grids:
- the *embarrassingly parallel* approach, which makes use of the above two assumptions. The two assumptions imply that we can execute different batches of histories independently, offering a convenient way to use multiple processors concurrently to solve the problem. When combined, the results will be identical to that of a serial computation of all histories, one after another, on a single processor. In other words, instead of running a long simulation on a single processor, we run many short simulations on many processors.

  Note that the two assumptions above are conditioned intrinsically within the Monte Carlo codes i.e. the assumptions are true whether or not the *embarrassingly parallel* approach is used.

  For validation purposes, this approach provides multiple output files which can be analysed, before being combined, for statistical agreement. Thus, outliers may be used to identify irregularities (e.g. data tampered in transit or incorrect compilation).

- the *true parallel* approach, where processors communicate with each other periodically as they execute the simulation. In this case the simulation is *not* split into batches of histories. This approach would be indispensable if the two assumptions above are not valid, in which case executions of each radiation history would need to talk to each other.

  This approach typically requires design or intervention at the code level, for example, by inserting MPI [14] calls into the source code. Most codes have been developed without having the *true parallel* approach built in. Rewriting a code would require much effort before reaching maturity – we therefore did not consider rewriting codes.

  At the practical level, this approach saves the post-processing step of having to combine multiple output files. Only one output file is produced per simulation. An example is the PVM / MPI capability of MCNP / MCNPX [10].

Based on the two assumptions contained in the code we considered how to employ multiple processors for simultaneous computation of a Monte Carlo problem. We chose the *embarrassingly parallel* approach, and have included EGSnrc/BEAMnrc/DOSXYZnrc, FLUKA, MCNP/MCNPX, GEANT4 and PENELOPE in our

embarrassingly parallel framework. This is consistent with our belief that different Monte Carlo codes should be available to the Monte Carlo physicist, who should be able to switch seamlessly between codes, according to the specific aspect of the problem at hand. For example, BEAMnrc offers a convenient array of 'jigsaw' pieces, named *component modules*, for constructing linear accelerators but it does not simulate neutron transport; FLUKA does not provide similar 'jigsaw' pieces for radiotherapy applications but it does offer full analog transport as an option.

The *embarrassingly parallel* framework

Components in the framework are:
1. Input and output files.
2. Application and library files.
3. Resource broker and grid middleware (e.g. Condor-G[1 5], Nimrod [12], OMII GridSAM[1] and gLite[2]) which provide an abstraction layer for the distributed resource management issues.
4. Remote batch queue scheduler (Condor [11], PBS and SGE)., the resource management tool at the local cluster level, handled transparently by the grid middleware.
5. Execution script, which runs the following tasks, in the given order, at each execution node
   - inflate compressed file into an organised filesystem as expected by the code;
   - set environment variables where necessary;
   - assign random number seed using job index;
   - append input (and consequently, output) filenames with job index, so that the returning output files do not overwrite each other; and
   - execute command line specific to the Monte Carlo code.

Input and output files may be accessible through a shared file system, especially if the computation is run on a cluster. However, in general, we need to specify the files to be transferred to and from the processing nodes. If the simulation has been split into *N* shorter simulations, each executed by a processing node, we will have *N* returning output files. Combining the output files is straight-forward. We have a simple Perl script which can be easily adapted for reading the expected file format, which is different not only when different Monte Carlo codes are used, but also when different problems are simulated. For each of the *N* simulations we usually ask for equal numbers of histories so that the scores/tallies may be combined by taking either the sum or the mean.

Applications are typically pre-installed on the processing nodes. This makes dealing with architectural differences easier. However, it is possible to specify different files dependent on the architecture and operating systems of the processing nodes if necessary.

Within a single cluster, the use of a firewall is often enough to prevent misuse of the resources by untrusted users. A grid, on the other hand, requires the use of more robust security between sites. A grid middleware provides suitable mechanisms for authentication, authorisation, and confidentiality between machines at individual sites. This security typically uses X.509 certificates and the Secure Socket Layer (SSL) protocol, the same mechanism used in secure web requests. A modified version, the Grid Security Infrastructure (GSI) [13], allows delegation, which is the chaining of multiple steps within a secure conversation. This is necessary for computations where, for example, a user makes a secure connection to a remote site to start a computation, and the computation must then make a secure connection to a data repository to retrieve a data file, using the original user's credentials. To provide the security credentials, the UK National Grid Service runs a Certificate Authority.

Code-specific details

	Light-weight filesystem (sample)	Environmental variables	Execution line (sample)

EGSnrc / BEAMnrc / DOSXYZnrc	`data/incoh.data`
	`data/msnew.data`
	`data/nist_brems.data`
	`data/photo_cs.data`
	`data/photo_relax.data`
	`data/eii_ik.data`
	`data/pair_nrc1.data`
	`data/rad_compton1.data`
			`./dosxyznrc.exe -i`
			`${1} -p myinput -e .`
			`-H . > ${1}.egslog`

---

	data/spinms.data pegs4/data/mypegs.pegs4dat dosxyznrc/dosxyznrc.io dosxyznrc/myinput.egsinp dosxyznrc.exe myphantom.egsphant		
FLUKA	flutil/rfluka fluodt.dat random.dat xnloan.dat brems_fin.bin cohff.bin elasct.bin gxsect.bin neuxsc_72.bin nuclear.bin sigmapi.bin myinput.inp	FLUPRO	$FLUPRO/flutil/rfluka -N0 -M1 ${1}
PENELOPE	penslab.exe mycortbone.mat myinput.in		./penslab.exe < ${1}.in > ${1}.out
MCNP / MCNPX	mcnp.exe xsdir xs/crosssectionfiles myinput		./mcnp.exe inp=myinput
GEANT4	libCLHEP-1.9.2.1.so myexecutable.exe run.mac crosssectionfiles	LD_LIBRARY_PATH G4LEVELGAMMADATA G4RADIOACTIVEDATA NeutronHPCrossSection	./myexecutable.exe run.mac ${1} > ${2}.g4out

The job ID is passed as an argument by the resource broker to the script. Since the job ID is unique, we conveniently use it as the random number seed, as well as the serial number in the filename so that the returning output files do not overwrite each other.

Validation of simulation integrity

Validation and quality assurance are particularly crucial when:
- simulation outcome potentially saves or harms human life e.g. radiotherapy patients and staff exposed to radiation hazards;
- simulation outcome is of national or international scales, leading to scientific authority and decision-making;
- we use distributed computing resources, where we do not have administrative control;
- we use non-dedicated computing resources, in competition with a broad spectrum of users and activities; or
- the Monte Carlo code is not pre-installed and/or pre-compiled on the execution node itself.

For example, Cardiff University has a Condor pool consisting approximately 800 open access PCs, whose idle time we make use of for Monte Carlo simulations. As opposed to closed and dedicated clusters, such PCs are open for use by students and staff in the university. While patterns are somewhat predictable as to the time of the day and week when more PCs become free for Monte Carlo simulations, it is usually difficult to predict the specific PC that will be recruited by the resource broker, which operates according to user-defined criteria (e.g. processor MHz, memory, disk space, operating system).

In practice we have 2 optional validation steps:
1. Pre-simulation. Before simulation of the problem proper, short simulations of known results are executed and the results are compared against known results. This step is analogous to the test suite routinely exercised as the final step in a MCNP/MCNPX installation. However, we introduce several improvements to reduce false positives as well as false negatives:
   - we analyse the score/tally numerically for statistically significant deviations, instead of using *diff* (Unix family) or *fc* (Windows), which compare two files lexically. This reduces problems due to different random number sequences or floating point errors. For example, *diff* and *fc* would report

1.0000000000000001 and 1.0000000000000002 as different. Numerical comparison can consider these to be equivalent within a given tolerance.
- we check for the successful completion of a simulation before analysing its results for deviation from the expected values. If a simulation fails and no output is produced, the *diff* between a non-existent output file and the reference output file echoes nothing to the standard output, which would be mistaken as no difference.
2. Post-simulation. After successful completion of the simulations and before the results are combined, results from each processor are analysed for statistical agreement. Outliers indicate irregularities (e.g. data tampered in transit or incorrect compilation).

Note that the above discussion is dedicated to validation of our simulation, not validation of the Monte Carlo code. The purpose is to ensure that our simulations run properly as intended by the code, not to check whether the code runs properly with respect to physics. For the latter, we usually begin a project by running simple simulations under control conditions, and compare the results with known behaviours from databases and textbooks.

Acknowledgement

References

[1] Kawrakow I and Rogers D W O. The EGSnrc Code System, NRC Report PIRS-701. 2000 NRC, Ottawa.
[2] Rogers D W O, Ma C-M, Walters B, Ding G X, Sheikh-Bagheri D, and Zhang G. BEAMnrc Users Manual, NRC Report PIRS-0509A (rev G). 2001 NRC, Ottawa.
[3] Walters B R B and Rogers D W O. DOSXYZnrc Users Manual, NRC Report PIRS-794. 2002 NRCC, Ottawa.
[4] Fasso A, Ferrari A and Sala P R. "Electron–photon transport in FLUKA: status," in Advanced Monte Carlo for Radiation Physics, Particle Transport Simulation and Applications, Proceedings of the Monte Carlo 2000 Conference, Lisbon. Ed. Kling A, Barao F, Nakagawa M, Tavora L, and Vaz P. 2000 Springer-Verlag, Berlin.
[5] Fasso A, Ferrari A, Ranft J and Sala P R, "FLUKA: Status and Prospective for Hadronic Applications," in Advanced Monte Carlo for Radiation Physics, Particle Transport Simulation and Applications, Proceedings of the Monte Carlo 2000 Conference, Lisbon. Ed. Kling A, Barao F, Nakagawa M, Tavora L, and Vaz P. 2000 Springer-Verlag, Berlin.
[6] Briesmeister J F, MCNP—a General Monte Carlo N-Particle Transport Code, Report No. LA-12625-M. 1997 Los Alamos National Laboratory, Los Alamos.
[7] Hendricks J S et al. MCNPX, VERSION 2.5e LA-UR-04-0569. 2004 Los Alamos National Laboratory, Los Alamos.
[8] Agostinelli S et al. GEANT4 – a simulation toolkit. 2003 *Nucl. Instr. and Meth. A 506*.
[9] Salvat F, Fernández-Varea J M, and Sempau J. PENELOPE, a Code System for Monte Carlo Simulation of Electron and Photon Transport. 2003 OECD Nuclear Energy Agency, Issy-les-Moulineaux-France.
[10] McKinney G W. A Practical Guide to Using MCNP with PVM. 1994 Trans. Am. Nucl. Soc. 71.
[11] Thain D, Tannenbaum T, and Livny M, Distributed Computing in Practice: The Condor Experience *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.
[12] Abramson D and Giddy J, Scheduling large parametric modelling experiments on a distributed metacomputer. Proceedings Seventh International Parallel Computing Workshop (PCW'97), Canberra, Australia, September 1997.
[13] Butler R, Engert D, Foster I, Kesselman C, Tuecke S, Volmer J, and Welch V. A National-Scale Authentication Infrastructure. *IEEE Computer*, 33(12):60-66, 2000.
[14] Snir M, Otto SW, Huss-Lederman S, Walker DW, and Dongarra J, "MPI - The Complete Reference: Volume 1, The MPI Core," second edition, pub. MIT press, 1998.
[15] Thain D, Tannenbaum T, and Livny M. "Condor and the Grid." Grid Computing – Making the Global Infrastructure a Reality, ed. Berman F, Hey A, and Fox G. John Wiley & Sons, 2003.