A Performance Model of Non-deterministic Particle Transport on Large-Scale Systems

Mark M. Mathis[1,⋆], Darren J. Kerbyson[2], and Adolfy Hoisie[2]

[1] Dept. of Computer Science, Texas A&M University
mmathis@cs.tamu.edu
[2] Performance and Architectures Lab, CCS-3, Los Alamos National Laboratory
{djk,hoisie}@lanl.gov

Abstract. In this work we present a predictive analytical model that encompasses the performance and scaling characteristics of a non-deterministic particle transport application, MCNP. Previous studies on the scalability of parallel Monte Carlo eigenvalue calculations have been rather general in nature [1]. It can be used for the simulation of neutron, photon, electron, or coupled transport, and has found uses in many problem areas. The performance model is validated against measurements on an AlphaServer ES40 system showing high accuracy across many processor / problem combinations. It is parametric with both application characteristics (e.g. problem size), and system characteristics (e.g. communication latency, bandwidth, achieved processing rate) serving as input. The model is used to provide insight into the achievable performance that should be possible on systems containing thousands of processors and to quantify the impact that possible improvements in sub-system performance may have. In addition, the impact on performance of modifying the communication structure of the code is also quantified.

1 Introduction

MCNP is a general purpose Monte-Carlo N-Particle code that represents part of the Accelerated Strategic Computing Initiative (ASCI) workload. It can be used for the simulation of neutron, photon, electron, or coupled transport [2]. Particle transport simulation has found uses in many problem areas including nuclear reactors, radiation shielding, and medical physics. There is great interest in the use of non-deterministic particle simulation on large-scale systems - both those currently in existence as well as future advanced systems being proposed.

A model of MCNP is required to assess the performance that can be obtained on current and future large-scale systems. In particular, a model can provide information to users on what size problem can be processed given a time allocation or what size problem needs to be processed in order to achieve a desired quality of results. The model can also be used in the procurement, and consequent

⋆ Mathis is supported in part by a DOE High-Performance Computer Science Fellowship. Los Alamos National Laboratory is operated by the University of California for the National Nuclear Security Administration of the US Department of Energy.

installation, of future systems by providing information on what performance should be achievable prior to actual system availability (see for example [3]). The model can also be used to identify bottlenecks in the code and to make recommendations for its future development.

In this work we develop a detailed analytical performance model of MCNP. The model consists of two fundamental parts: an application model and a system model. The application model is based on a static analysis of the key portions of the code but is parameterized in terms of the data specific to the problem being simulated. The application model is combined with the system model in order to evaluate performance predictions for a specific system. The system model encapsulates key system characteristics such as communication (e.g. latency and bandwidth), and computational performance (e.g. processor speed). The two parts of the model are kept separate so the model can be re-used without alteration to explore a multitude of performance scenarios. For instance, one may evaluate a different problem by setting the appropriate input parameters to the application model, or evaluate a new machine by changing the input values to the system model. A similar modeling approach has been used to model other large-scale applications including deterministic transport on structured and unstructured meshes [4,5,6], and adaptive mesh refinement [7,8].

Motivation and Previous Work. Criticality safety is a vital part of the storage, transportation, and processing of fissionable materials. Criticality may be defined as that state of a nuclear chain-reacting medium when the nuclear fission chain reaction just becomes self-sustaining (critical). MCNP includes the capability to calculate eigenvalues for critical systems and forms the particular input studied in this work. The example geometry consists of an insulated barrel containing a number of hollow rods of fissionable material. Horizontal and vertical cross-sections of the geometry are shown in Fig. 1. The shading is used

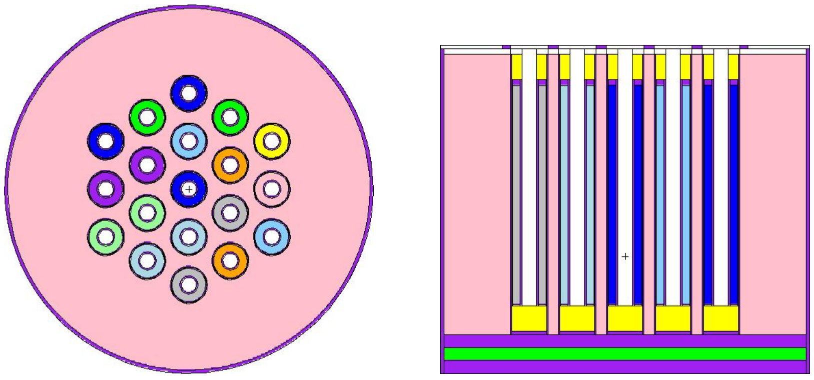

(a) (b)

Fig. 1. Horizontal (a) and vertical (b) cross-sections of example geometry

to indicate the different material properties of each rod and also of the insulated barrel. The hollow portion of the rods are indicated as white. The goal of the simulation is to determine if the arrangement of rods is safe, i.e. non-critical.

2 MCNP

MCNP can trace its roots back to the invention of the Monte Carlo method at Los Alamos during World War II. The Monte Carlo method is generally attributed to Fermi, Metropolis, von Neumann, Richtmyer and Ulam [9]. It was one of the first application programs run on early computers in the 1950's. MCNP is the successor to their work and represents over 450 person-years of development. Version 4C of MCNP was used in this analysis.

Monte Carlo methods in general and MCNP specifically do not solve an explicit equation, but rather obtain answers by simulating the interactions between individual particles and a predefined geometry. The accuracy of the calculation increases in proportion to the number of particles used in the simulation. In general, the error in the calculation reduces as the square root of the number of particles. This is in contrast to deterministic transport methods, the most common of which is the discrete ordinates method, that actually solve the transport equation directly for the average particle behavior [10].

The input geometry for MCNP consists of a collection of cells defined as combinations of primitive shapes such as planes, cylinders and spheres. The material properties are retrieved from an external library . The behavior of each simulated particle and its interaction with the materials travelled through, as defined by the geometry, are recorded to produce a particle *history*. During this process, statistical information about certain events is gathered in histograms or *tallies*. The interaction of particle and geometry can result in several events such as neutron/photon scatter, capture, and leakage.

The current parallelization strategy of MCNP requires the geometry to be copied to all processors and thus the complexity of the geometry is constrained by the memory available in a single processing node [11]. Parallelism can be utilized to either solve the same problem faster by sub-dividing the simulated particles across all processors (strong scaling), or to give a more accurate simulation by simulating more particles in proportion to the number of processors (weak scaling). During each cycle of MCNP, each processor simulates a designated set of particles. At the end of each cycle, a single processor merges the results from all other processors during a *rendezvous*. This communication pattern requires several steps and a fairly high degree of coordination. Note that the achievable performance of MCNP is both input sensitive (the cost to simulate a particle depends on the complexity of the geometry and materials used) and output sensitive (the complexity of the output depends on the requested tallies).

3 Performance Model

MCNP is representative of a general parallel application paradigm known as master-slave. In this paradigm, a master process is responsible for dividing the work to be done across a number of slave processes. Work assignments and state information are distributed from master to slaves during a "scatter" phase at the start of each cycle. Once the slaves receive their assignments they may begin their local computation. Once the slaves have completed their work, they report their results to the master during a "gather" phase. The master then aggregates the results from all the slaves and a new cycle begins. The "scatter" and "gather" phases may actually consist of a sequence of messages. For MCNP the scatter phase consists of 2 communications, and the gather phase consists of 5 communications (Fig. 2).

MCNP is particularly well-suited to the master-slave paradigm due to the independent nature of particle simulation . During the work phase of MCNP there is no communication between slaves (i.e. any particle is simulated independently of any other particle). Unfortunately, this apparent strength reveals a hidden weakness of MCNP and the master-slave approach in general. First, all communication must go through the master and second, the entire geometry must be replicated. The amount of data transmitted from the master to each slave in the scatter phase, and from each slave to the master in the gather phase results in a scaling limitation. The performance of MCNP scales well until communication costs dominate the execution time. The first limitation could be potentially avoided by altering the manner in which data is reported to the master. The second could be overcome by allowing communication between slaves, to relay geometry or particle information.

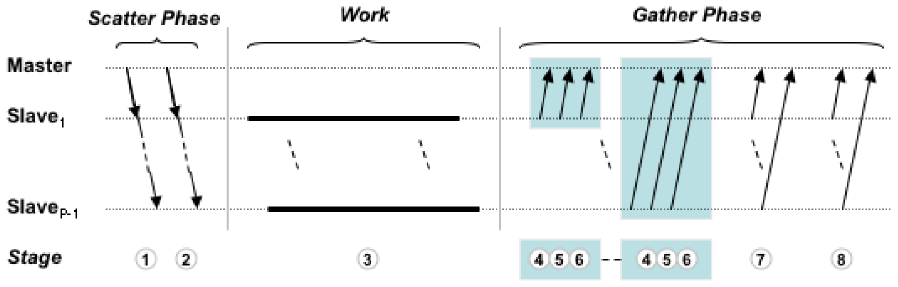


Fig. 2. Communication pattern for a single cycle of MCNP

3.1 Application Model

The application model includes only those portions of the parallel activity that significantly contribute to the overall execution time. The main stages of a cycle of MCNP corresponding to those depicted in Fig. 2 are listed in Table 1. The

Table 1. Summary of parallel activity for one cycle of MCNP. M=Master, S=Slave(s)

Stage	Source	Action	Size (bytes)	Description
1	M	bcast	$P * 8$	particle range to be computed by slaves
2	M	bcast	229240	update to current history
3	S	work	$T_{hist} * \lceil N_{ph}/(P-1) \rceil$	T_{hist} times the number of histories
4	S	pt2pt	5512	task common
5	S	pt2pt	320	tally data
6	S	pt2pt	204920	task array 1
7	S	pt2pt	$48 * \lceil N_{ph}/(P-1) \rceil$	task array 2
8	S	pt2pt	32	timing data

table summarizes the event source (either master or slave), its type (either collective broadcast, point-to-point communication, or computation), and also the weight associated with the event for each stage of the cycle. The weight is in bytes for all communication events (message sizes), and in terms of the number of particle histories for the computation events. The sizes of some of the messages are dependent upon the actual problem being solved. These sizes must be measured prior to the use of the model.

The first task array message (stage 6 in Table 1) is constant for each input geometry and the requested tallies. The tally data message (stage 5) can be calculated simply as the word size (8) times 2 plus the number of requested tallies (38 for this problem). The constant 48 involved in the task array 2 message (stage 7) is obtained by run-time measurement. This is related to the average number of collisions experienced by each particle.

The execution time for a single cycle of MCNP can be modeled as:

$$T_{total} = T_{scatter} + T_{slave} + T_{gather} \tag{1}$$

where the cycle time, T_{total}, is a summation of the scatter, work, and gather phases - $T_{scatter}$, T_{slave}, and T_{gather}, respectively. The form of this model is additive since the gather and scatter stages are in general synchronized by the bottleneck caused by the master, and the serialization of the communication from the slaves to the master. Each cycle begins with a scatter phase:

$$T_{scatter} = T_{bcast}(P * 8, P) + T_{bcast}(229420, P) \tag{2}$$

where the time to perform the collective broadcast operation, $T_{bcast}(S, P)$, is the time taken to broadcast S bytes across P processors on the target system. The scatter phase corresponds to the first two stages in Table 1.

The computation phase, performed on each slave, can be modeled as:

$$T_{slave}(P, N_{ph}, T_{hist}) = \left\lceil \frac{N_{ph}}{(P-1)} \right\rceil * T_{hist} \tag{3}$$

where N_{ph} is the number of particle histories per cycle which are divided amongst the $P-1$ slave processors. In general, it is more accurate to take the computation

Table 2. Summary of system model parameters (S in bytes)

$L_c(S), B_c(S)$	$T_{pack}(S)$	T_{hist}
$\begin{cases} 5.05\mu s, 0.0MB/s & S < 64, \\ 5.47\mu s, 78MB/s & 64 \leq S < 512, \\ 10.3\mu s, 294MB/s & S \geq 512 \end{cases}$	$\begin{cases} 0.12ns & S < 32K, \\ 0.16ns & 32K \leq S \leq 4M, \\ 0.67ns & S > 4M \end{cases}$	$798\mu s$

time for the slowest slave. However, since each slave is responsible for an equal number of particles, we assume that all slaves will take the same time. The time to perform a single particle history, T_{hist}, can be measured on a single processor for the problem being solved. The gather phase can be modeled as:

$$T_{gather}(P, N_{ph}) = \sum_{i=1}^{P-1} \Bigg(T_{pt2pt}(5512, i, 0) + T_{pt2pt}(320, i, 0) + T_{pt2pt}(204920, i, 0) + T_{pt2pt}(48 * \left\lceil \frac{N_{ph}}{(P-1)} \right\rceil, i, 0) + T_{pt2pt}(32, i, 0) \Bigg) \quad (4)$$

where the five point-to-point communications, listed as stages 4-8 in Table 1, are effectively performed in a serialized way due to the master bottleneck. However, an examination of the current messaging within MCNP indicates that some of the data transfered between all slaves and the master (specifically stages 4, 5, possibly part of 6, and 8 in Table 1) can be at least partially implemented as collective reductions. If we assume that all of stages 4, 5, and 8 as well as half of stage 6 can be reduced, equation 4 can be re-written as:

$$T_{gather}(P, N_{ph}) = \sum_{i=1}^{P-1} \Bigg(T_{pt2pt}(102460, i, 0) + T_{pt2pt}(48 * \left\lceil \frac{N_{ph}}{(P-1)} \right\rceil, i, 0) \Bigg) + T_{reduce}(5512, P) + T_{reduce}(320, P) + T_{reduce}(102460, P) + T_{reduce}(32, P) \quad (5)$$

3.2 System Model

For the application model as formulated in Sect. 3.1, the required components of the system model are point-to-point communication times, collective broadcast times, the time required to perform a single particle history , and also the memory performance of a single node (for packing). MCNP actually uses the UPS messaging library [12] for communication between processors. UPS provides a generic interface with a retargetable backend. It allows a message of arbitrary length to be built from many smaller variables using packing functions in a similar way to that of PVM - a feature that is heavily utilized in MCNP. In the analysis that follows a 32 node AlphaServer ES40 cluster is used as the experimental testbed. This machine has four processors per node interconnected using the Quadrics QsNet high-performance network [13]. This network boasts high-performance communication with a typical MPI latency of $5\mu sec$ and a throughput of up to $340MB/s$ in one direction.

Measured MPI latency and bandwidth for inter-node unidirectional communication (point-to-point) were obtained using in-house benchmarks. The collective broadcast and reduction operations for P processors can be assumed to take $log_2(P)$ times that of a single point-to-point communication. The communication costs also include packing operations, implemented in UPS. The point-to-point, broadcast, and reduction communication operations are modeled as:

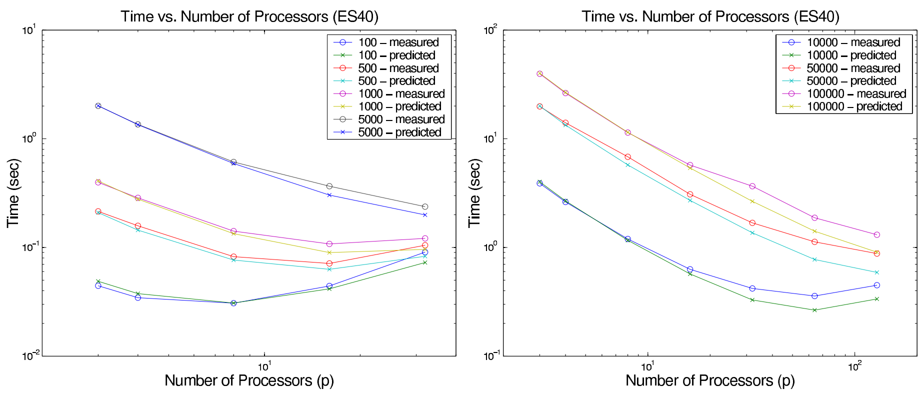
$$T_{pt2pt}(S, src, dest) = T_{pack}(S) + L_c(S) + S/B_c(S) \tag{6}$$

$$T_{bcast}(S, P) = T_{pack}(S) + T_{pt2pt}(S) * log_2(P) \tag{7}$$

$$T_{reduce}(S, P) = T_{pack}(S) + T_{pt2pt}(S) * log_2(P) \tag{8}$$

where S is the size of the message in Bytes, $T_{pack}(S)$ is the time to pack a single byte, $L_c(S)$ and $B_c(S)$ are the latency and bandwidth of a message of size S bytes. The parameters used in the system model are summarized in Table 2.

Model Validation. The model is validated against measurements made on our testbed system showing high accuracy – typically to within a 10% error. Measurements and predictions are shown in Fig. 3 for 7 sets of particle histories per cycle (100, 500, 1000, 5000, 10000, 50000, and 100000) on a range of processor counts (a strong-scaling analysis). The geometry is the same for all runs. Note that for small N_{ph}, the communication costs soon dominate the processing time, resulting in poor scalability. For large N_{ph}, the scalability is better up to a higher processor count.



(a) (b)

Fig. 3. Measured and predicted times for small (a) and large (b) inputs

4 Performance Study of MCNP

In this section the model is used to explore the performance of MCNP on larger configurations of current systems. It is also used to explore the possible improvements resulting from refining the code to make use of reduction operations as suggested in Sect. 3.1. The model is further utilized to investigate the performance of future systems assuming performance improvements in individual sub-system characteristics such as latency, bandwidth, and processing speed.

Scaling Behavior of Larger Systems. The MCNP performance model is used to explore the expected performance on larger AlphaServer ES40 systems in Fig. 4 for both strong and weak scaling models. As the processor count increases in the strong scaling mode, the amount of work per slave decreases and hence communication costs soon become a significant percentage of the overall run-time. In weak-scaling, as the processor count increases the amount of computation per processor is constant and thus the overall run-time increases more gradually due to increased communication costs. Overall it can be seen that in a strong-scaling mode, MCNP scales up to 512 processors on the problem being studied due to communication costs soon becoming significant. In weak-scaling, the performance of MCNP is much better and actually scales up to 8192 processors.

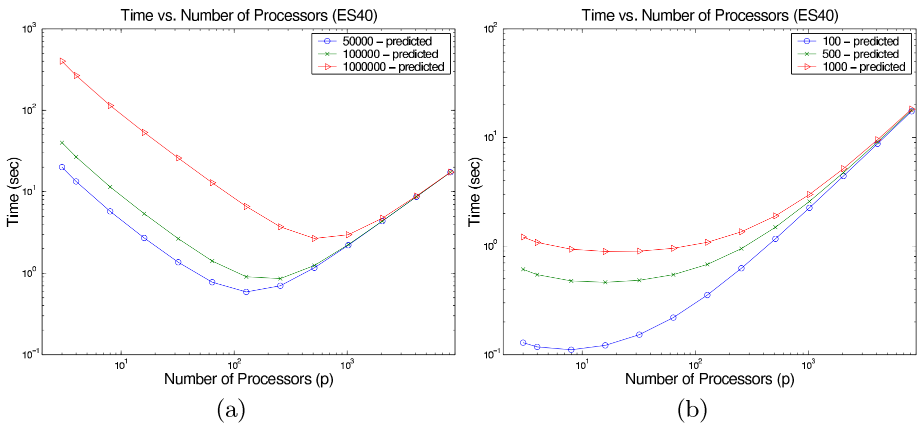


Fig. 4. Expected cycle times on larger ES40 systems for strong (a) and weak (b) scaling

Performance Predictions on Faster Systems. The impact of system performance improvements on the run-time of MCNP can also be quantified in advance of such systems being available. Here we examine a number of what-if scenarios by considering the performance of the communication and computation sub-systems to be improved by a factor of 8 individually. The factor of 8 was chosen to be indicative of what may happen to these sub-system performances

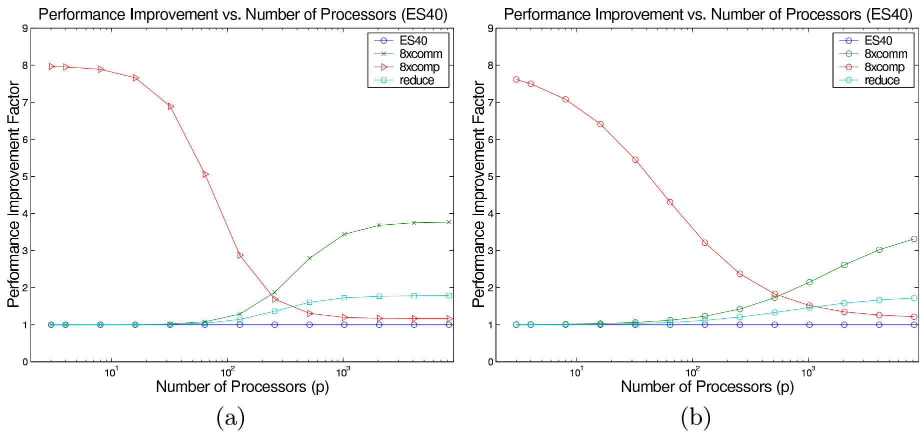


Fig. 5. Relative impact of improved computation and communication capabilities for strong (a) and weak (b) scaling

over the next 5 years.. The relative improvement over the existing ES40 system is shown in Fig. 5. Also included in Fig. 5 is the relative performance improvement that could be obtained if just the modifications to the code as described in Sect. 3.1 above were implemented.

It can be seen from Fig. 5 that an increase in computation capability has a much greater impact on performance for small numbers of processors, but rapidly declines as the number of processors is increased. Similarly on larger processor counts, the increase in communication capability will have a larger impact (due to communication constituting a larger percentage of the execution time as the number of processors increases).

5 Conclusion

In this work we have developed a detailed analytical performance model for MCNP. The model includes the main code characteristics and separates out the application and system characteristics. The model is based on a static analysis of the application but is parameterized in terms of its dynamic behavior. Through a validation process on a 32 node AlphaServer ES40 cluster, we have shown the model to be accurate with a typical error of 10%.

The model has been used to explore a number of performance scenarios. In a scalability analysis, the model was used to give expected performance on larger ES40 systems. This analysis showed that in a weak-scaling mode the application will scale to thousands of processors whereas in a strong-scaling mode the application scales to only hundreds of processors.

The performance of MCNP was also examined for the case of modifying the communication structure in the application to include the use of collective reductions. This analysis indicated that the performance could be improved on large processor counts if such modifications were implemented. In addition, the performance of MCNP was examined on a number of hypothetical systems which included faster computation or communication sub-systems. It was shown that increases in computation speed have the greatest effect on smaller processor counts, and increases in communication speed have greatest effect on larger processor counts.

Through these analyses the benefits of developing a performance model of an application have been illustrated. Once such a model has been validated it can be used to predict performance on systems or configurations that cannot be measured. The model has been used to analyze many scenarios in this work, and will be used to explore the performance on future machines as they become available. The model is part of an ongoing effort to model the ASCI workload and complements existing models for deterministic particle transport on structured and unstructured meshes and for adaptive mesh refinement applications.

References

1. Matsuura, S., Blomquist, R.N., Brown, F.B.: Parallel Monte Carlo Eigenvalue Calculations. Transactions of the American Nuclear Society **71** (1994) 199–202
2. Briesmeister, J.F.: MCNP^{TM}- A General Purpose Monte Carlo N-Particle Transport Code, Version 4C. Los Alamos National Laboratory. (2000)
3. Kerbyson, D.J., Hoisie, A., Wasserman, H.J.: Use of Predictive Performance Modeling During Large-Scale System Installation. In: 1st Int. Workshop on Hardware/Software Support for Parallel and Distributed Scientific and Engineering Computing, SPDSEC02, Charlottesville (2002)
4. Hoisie, A., Lubeck, O., Wasserman, H.J.: Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures Using Multidimensional Wavefront Applications. Int. J. of High Performance Computing Applications **14** (2000) 330–346
5. Kerbyson, D.J., Hoisie, A., Pautz, S.D.: Performance Modeling of Deterministic Transport Computations. In: Performance Analysis and Grid Computing, Kluwer (2003)
6. Mathis, M.M., Amato, N.M., Adams, M.L.: A General Performance Model for Parallel Sweeps on Orthogonal Grids for Particle Transport Calculations. In: ICS, Santa Fe (2000) 255–263
7. Kerbyson, D.J., Alme, H.J., Hoisie, A., Petrini, F., Wasserman, H.J., Gittings, M.L.: Predictive Performance and Scalability Modeling of a Large-scale Application. In: Supercomputing, Denver (2001)
8. Kerbyson, D.J., Wasserman, H.J., Hoisie, A.: Exploring Advanced Architectures using Performance Prediction. In: Innovative Architecture for Future Generation High-Performance Processors and Systems, IEEE CS Press (2002) 27–37
9. Metropolis, N., Ulam, S.: The Monte Carlo Method. J. Amer. Statist. Assoc. **44** (1949) 335–341
10. Koch, K.R., Baker, R.S., Alcouffe, R.E.: Solution of the first-order form of the 3D discrete ordinates equation on a massively parallel processor. Transactions of the American Nuclear Society **65** (1992) 198–199
11. Cox, L.J.: DMMP Upgrade for MCNP4C^{TM}. Los Alamos National Laboratory Research Note (2001)

12. Barrett, R., McKay, M.: UPS: Unified Parallel Software User's Guide and Reference Manual. Los Alamos National Laboratory. (2002)
13. Petrini, F., Feng, W.C., Hoisie, A., Coll, S., Frachtenberg, E.: The Quadrics Network: High-Performance Clustering Technology. IEEE Micro **22** (2002) 46–57