# Computational Cost Analysis of the Milagro Implicit Monte Carlo Code

*Scott W. Mosher, Timothy M. Kelley, CCS-2*

Developers of production-level transport codes are motivated (by their user community if not by their own curiosity) to analyze and understand the computational cost of their methods and algorithms. Ideally, one would like to know an algorithm's execution time as a function of various input parameters. The information is useful to users for predicting run times, to developers for evaluating the software implementation, and to both groups for understanding the limits of the algorithm's practical application. To this end, we have completed a systematic survey [1] of the serial (i.e., single processor) performance of the Milagro code [2] for nonlinear, time-dependent, radiative transfer simulations based on the Implicit Monte Carlo method [3]. More recently, we have developed a mathematical model to predict the execution time of Milagro transport simulations.

The computational cost of an implicit Monte Carlo calculation depends on several input variables, such as the number of geometric cells, cell size, material opacity, number of frequency groups, etc. In the survey described in Ref. [1], we measured the computational cost of several sections of the Milagro code over a wide range of input parameters. We analyzed the results to compare the relative cost of different sections, as well as the scaling of the costs with respect to changes in the input parameters. That analysis confirmed that the relative costs and cost scaling of nearly all code sections were as expected. The analysis also identified something unexpected—that a major consumer of CPU time (up to 50% in some cases) was the system-level memory allocation function. Further analysis found that the allocation and deallocation patterns in the Milagro code caused the allocation routine to spend a tremendous amount of time sorting freed blocks of memory. The bottleneck was eliminated by switching to an alternative allocation routine.

The data generated during the survey were catalogued and comprise a performance baseline with which to compare timing results from future versions of the code. A similar exercise was completed as part of this project. That is, a comparison was made between the then-current version of the code and version 4_1_0 from 2004. It was found that the computational efficiency of the code has increased substantially as a result of several improvements. For example, the total computational time as a function of the number of photon frequency groups for both versions running a sample problem is plotted in Fig. 1. It can be seen that the overall cost of the simulation was reduced by about 25 to 50%.

In recent work, we have been developing a mathematical model to predict the computational cost of the transport operations of Milagro. In a Monte Carlo transport algorithm, the execution time per particle scales with the number of events (e.g., scattering collisions, interface crossings, etc.) that occur in the simulation. The number of events per unit simulation time depends on the input parameters in

a complicated way. For an idealized problem, the mean number of events (of various types) can be predicted analytically. From these analytic expressions, a cost model has been constructed to predict the average computational time consumed per transport cycle. The model contains parameters that depend on the implementation of the transport algorithm and the computer system on which the simulation is conducted. For a particular implementation running on a particular system, the parameters can be estimated by numerically fitting the parametric model to actual timing data.

Timing data were collected for a series of Milagro runs in which the independent variables were varied over reasonable sets of values. This led to a total of 144 individual problems (i.e., distinct combinations of the independent variables). The cost model was fit to the Milagro timing data and compared to the actual times. For the full set of data, the average absolute relative error in the predicted transport times was 1.97%. The maximum absolute relative error was 33.02%, which was much higher than expected. There were seven problems for which the predicted times were outside of three estimated standard deviations from the mean. In each of these cases, the model under-predicted the actual time. The actual number of events occurring in these simulations was tallied, and it was found that the model predicted the number of events to within 0.1%. Hence, it currently appears that the additional time was spent performing operations not incorporated into the model. This issue is currently being investigated. Excluding the seven outliers, the average absolute relative error in mean was 0.97% and the maximum was 4.31%.

***For more information contact Scott Mosher at swmosher@lanl.gov.***


[1] T.M. Kelley and S.W. Mosher, "Characterization of Implicit Monte Carlo Projects, Phase 1: Serial Performance of Milagro," Los Alamos National Laboratory memorandum CCS-4:06-03(U) (2006).
[2] T.J. Urbatsch and T.M. Evans, "Milagro Version 2 – An Implicit Monte Carlo Code for Thermal Radiative Transfer: Capabilities, Development, and Usage," Los Alamos National Laboratory report LA-14195-MS (2006).
[3] J.A. Fleck, Jr. and J.D. Cummings, *J. Comp. Phys.* **8**, 313 (1971).



**Funding Acknowledgements**
NNSA's Advanced Simulation and Computing (ASC), Computational Physics and Methods Mission Capability Program, Transport Project.


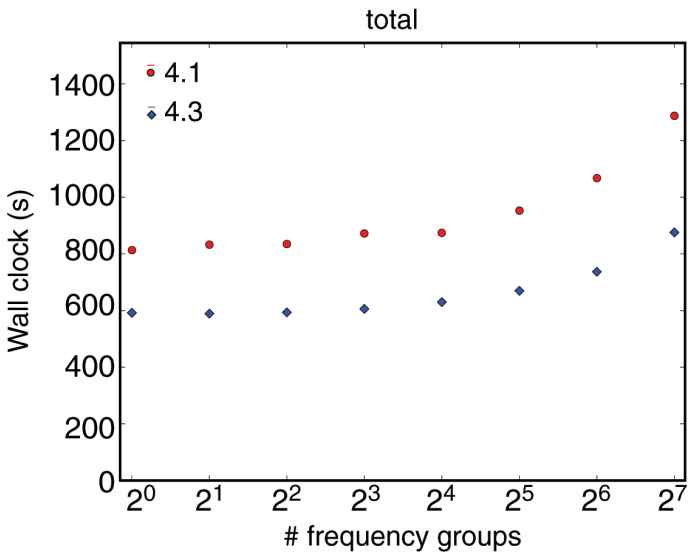


**Fig. 1.**
***Log-log plot of total wall-clock time vs number of groups. Red circles: head version 4_3_0+ (Dec. 2, 2005), blue diamonds: Milagro-4_1_0 (2004).***