This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

ANS M&C 2021 - The International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering · Raleigh, North Carolina · April 11–15, 2021

SAND2020-13765C

CONDITIONAL POINT SAMPLING IMPLEMENTATION FOR THE GPU

Luke J. Kersting[1], Aaron Olson[1], and Kerry Bossler[1]

[1]Sandia National Laboratories, Radiation Effects Theory Department
P.O. Box 5800, MS1179, Albuquerque, New Mexico 87185

ljkerst@sandia.gov, aolson@sandia.gov, kbossle@sandia.gov

ABSTRACT

Conditional Point Sampling (CoPS) is a recently developed stochastic media transport algorithm that has demonstrated a high degree of accuracy in 1D and 3D simulations implemented for the CPU in Python. However, it is increasingly important that modern, production-level transport codes like CoPS be adapted for use on next-generation computing architectures. In this project, we describe the creation of a fast and accurate variant of CoPS implemented for the GPU in C++. As an initial test, we performed a code-to-code verification using single-history cohorts, which showed that the GPU implementation matched the original CPU implementation to within statistical uncertainty, while improving the speed by over a factor of 4000. We then tested the GPU implementation for cohorts up to size 64 and compared three variants of CoPS based on how the particle histories are grouped into cohorts: successive, simultaneous, and a successive-simultaneous hybrid. We examined the accuracy-efficiency tradeoff of each variant for 9 different benchmarks, measuring the reflectance and transmittance in a cubic geometry with reflecting boundary conditions on the four non-transmissive or reflective faces. Successive cohorts were found to be far more accurate than simultaneous cohorts for both reflectance (4.3 times) and transmittance (5.9 times), although simultaneous cohorts run more than twice as fast as successive cohorts, especially for larger cohorts. The hybrid cohorts demonstrated speed and accuracy behavior most similar to that of simultaneous cohorts. Overall, successive cohorts were found to be more suitable for the GPU due to their greater accuracy and reproducibility, although simultaneous and hybrid cohorts present an enticing prospect for future research.

KEYWORDS: Conditional Point Sampling; CoPS; stochastic media; GPU; Monte Carlo transport

1. INTRODUCTION

Conditional Point Sampling (CoPS) is a recently developed Monte Carlo stochastic media transport algorithm [1-3] that has a high degree of accuracy in 1D [1] and 3D [3] simulations when benchmarked against, for example, the simplistic atomic mix approximation (AM) [4] and the celebrated Chord Length Sampling (CLS) [5] method [3]. The improved Poisson-Box Sampling method (PBS-2) [5] outperformed all tested versions of CoPS for reflectance [3] but nearly all tested variations of CoPS proved to be more accurate than all non-CoPS methods for transmittance.

Use of Woodcock tracking [6] (sometimes called delta tracking) enables CoPS to achieve efficiency gains compared to the "brute-force" approach of full transport simulations on many realizations by only sampling the parts of the geometry needed on-the-fly and accuracy gains compared to many approximate methods by making it tractable to remember and employ all (or most) previously sampled geometry features when sampling new geometry features. In CoPS, simulated particles stream to new pseudo-collision sites without requiring knowledge of what materials the particle has streamed through. At each

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

pseudo-collision site, the probability of sampling each material type is approximated, and a material is sampled using a pseudo-random number. Histories in the same "cohort" compute material probabilities at new pseudo-collision sites by considering points already sampled by all histories in the same cohort, and thus work together to partially define, or reveal, a realization of the stochastic media [2]. CoPS uses a conditional probability function (CPF) to compute the probability of sampling each material at each pseudo-collision site conditionally on previously sampled points in the same cohort. The notation "CoPS*N*" denotes how many points are used in a CPF evaluation, e.g., CoPS3 denotes use of two previously sampled points to inform the sampling probability of the new point. Colocation of more than one history in the same cohort enables computation of the variance of transport quantities caused by the stochastic nature of the materials [2]. While cohorts of as few as two histories can be used to compute output variance [2], ongoing work [7] has demonstrated that larger cohorts can be used to solve for full probability density functions (PDFs) of transport outputs.

However, in spite of its advantages in both speed and accuracy, CoPS was originally implemented in a Python-based serial-CPU research code, which inherently limits the algorithm's potential speed and applicability to problems with real physics. The first challenge addressed in this paper is our adaptation of CoPS to production-level C++ code that runs on the GPU. For the sake of simplicity, we only adapted CoPS2 (CoPS using a two-point CPF). The transition presented challenges, in part because C++ significantly predates the development of GPU (and therefore many modern C++ practices cannot be directly used for GPU), and in part because Monte Carlo codes are particularly problematic for architectures like the GPU (because they rely on highly divergent algorithms and can require huge memory capacity) [8, 9]. As such, our work is part of a trend in which production-level transport codes, such as Mercury [10], Shift [11], and MCNP [12, 13], are being made more compatible with GPUs, and the gains represented by this trend are robust. The original CPU implementation of CoPS was limited to using single-history cohorts in 3D, so our code-to-code comparison was limited to a cohort size of one history; however, the GPU implementation proved to be over three orders of magnitude faster while reproducing transport results within statistical uncertainty.

Our second challenge was to extend testing for the GPU implementation to cohorts of any size, with the eventual goal of computing output PDFs. In an attempt to maximize efficiency, we propose and weigh three different cohort implementations differentiated by whether the histories in the cohort are simulated one after another or at the same time: successive, simultaneous, and successive-simultaneous hybrid cohorts (Fig. 1). To prevent a race condition, in which a newly sampled point from one history could overwrite sampled points from another history, we used atomic operations. We examined the accuracy and efficiency of CoPS2 as a function of cohort type and the number of histories in a cohort and found that, although (as expected) the simultaneous and hybrid cohorts were far faster than the successive cohorts, the successive cohorts have a significantly higher degree of accuracy and uniquely yield reproducible results.

2. IMPLEMENTATION METHODOLOGY

In its CPU iterations, CoPS was designed with several variants. One of the simplest, CoPS2, uses a simple two-point CPF, in which only the point nearest to the new point is considered in CPF evaluations. More complex multi-D schemes have been outlined and benchmarked in Ref. [3] for a Markovian-media-based CPF that uses three-point relationships (CoPS3), four-point relationships (CoPS4), and higher point relationships (CoPS*N*). As demonstrated for Markovian-mixed media in Ref. [3], higher-point relationships have the capacity to yield greater accuracy, but also require more complex implementation and yield increased runtime. Nonetheless, we judged achieving accurate and efficient implementation of large cohorts on the GPU to be a more challenging problem than adapting higher-order CPFs and thus

chose to implement only the simple CoPS2 for this initial GPU investigation. All references in this document to "the GPU iteration of CoPS" should be understood as referring to CoPS2.

For both the CPU and GPU iterations of CoPS, CPF predictions for a new point are based on the geometrically closest previously sampled point and assume Markovian material mixing behaviors. This relationship, shown in Eq. (1), has been derived in 1D in Ref. [1] and in 3D in Ref. [3], and since the relationship is only along a line, the 1D and 3D Markovian-mixing-based CPFs are the same. The probability of sampling material type α a distance of r away from an existing point of either material type α (Eq. (1a)) or β (Eq. (1b)) is computed as

$$\pi(\alpha \mid \alpha,r) = 1 - (1 - P_\alpha)\left(1 - e^{-\frac{r}{\Lambda_c}}\right), \text{ and} \tag{1a}$$

$$\pi(\alpha \mid \beta,r) = P_\alpha\left(1 - e^{-\frac{r}{\Lambda_c}}\right), \tag{1b}$$

where P_α is the material abundance of material type α, and Λ_c is the correlation length of the material computed for binary, Markovian-mixed media as a function of the average chord lengths of the two materials

$$\Lambda_c = \frac{\Lambda_\alpha \Lambda_\beta}{\Lambda_\alpha + \Lambda_\beta}. \tag{2}$$

To verify that our GPU version of CoPS was correctly implemented, we tested it against the CPU implementation. However, because the CPU implementation was limited to single-history cohorts for 3D simulations, our code-to-code comparison was also limited to single-history cohorts.

Once a code-to-code verification was completed, we extended our testing of the GPU implementation for cohorts of up to 64 histories. Previous 1D examination of CoPS cohorts [2, 7] simulated particles in a cohort successively, which provided a significant degree of accuracy. In our implementation, we introduced an option that allows the user to define the number of histories in a cohort and the number of threads to use to simulate the histories on that cohort, here illustrated in Fig. 1. Successive cohorts employ only one thread per cohort, such that all the histories run successively, one after the other. (Our single-history cohort employed the same methodology, although it is not, technically, successive, since it concludes after the single history.) Simultaneous cohorts run the same number of histories and threads per cohort, such that all histories run at the same time on different threads. Hybrid cohorts run at least two threads per cohort simultaneously, such that histories run on at least two threads at the same time, and at least one thread runs at least one history after the first history on that thread.

One of the biggest challenges when writing a multithreaded or parallel code that needs to share resources is race conditions, which occur when multiple threads write to the same memory location at the same time, overwriting each other and causing data to be lost. To prevent race conditions, we used atomic operations to write new material points on the GPU, ensuring that two threads cannot write to the same point index at the same time. However, even though the implementation avoids race conditions that could lead to incorrect results, simultaneous and hybrid cohorts do not yield reproducible results because the CoPS algorithm is dependent on the order in which material points are sampled. Consequently, the order in which threads sample additional material points can change when the threads are running in parallel, rendering the results non-reproducible. The exact magnitude of the difference between two runs appears to be small, but this preliminary observation warrants further investigation. It could also be possible to modify the simultaneous/hybrid cohorts CoPS algorithm, sacrificing some efficiency to gain reproducibility.

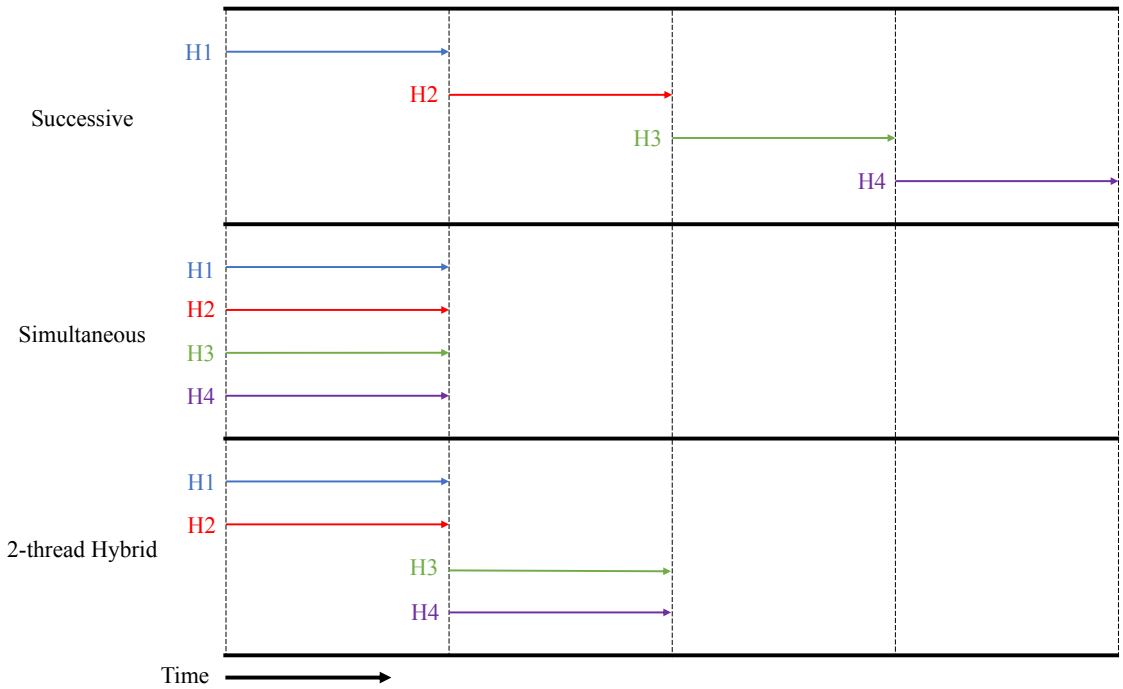

Figure 1. Simulation of Histories in Cohorts: Diagram of how histories (H1, H2, H3, and H4) are simulated over the runtime of a simulation for a cohort of size four using a successive, simultaneous, and 2-thread hybrid cohort.

3. NUMERICAL RESULTS

We performed a code-to-code verification of the CoPS2 GPU implementation using the 3D benchmark problem set proposed in Ref. [4] and used in previous work to assess the accuracy of CoPS [3]. Here, we compared the accuracy of the GPU implementation with selected non-CoPS algorithms using values from Refs. [4] and [5]. The problems in this benchmark set involve a cubic geometry of side length 10 with a normally incident plane source and reflecting boundary conditions on the four non-transmissive or reflective faces. Particles are either absorbed or scatter isotropically. Two constituent materials are mixed according to Markovian mixing and two quantities are computed: reflectance and transmittance. The nine problems in the benchmark set are described using the unique permutations of case number and case letter outlined in Table I.

Table I. Benchmark Set Parameters: The total cross section ($\Sigma_{t,j}$), average chord length (Λ_j), and scattering ratio (c_j) for materials j ∈ {0,1} in each benchmark problem.

Case Number	$\Sigma_{t,0}$	$\Sigma_{t,1}$	Λ_0	Λ_1	Case Letter	c_0	c_1
1	10/99	100/11	99/100	11/100	a	0.0	1.0
2	10/99	100/11	99/10	11/10	b	1.0	0.0
3	2/101	200/101	101/20	101/20	c	0.9	0.9

An NVIDIA V100 GPU was used to compare the GPU implementation of CoPS2 with the original CPU implementation of CoPS2 for single-history cohorts. The results (Table II) show that the GPU implementation agrees with the CPU implementation to within 2-sigma for benchmark 2b, and within 1-

sigma for the other benchmarks. The CPU and GPU results used 10^6 histories. In parentheses are 1-sigma statistical uncertainties on the last digit. The new C++ GPU implementation is, on average, more than 4000 times faster than the original Python CPU implementation.

Table II. Benchmark Comparison: Comparison of the GPU implementation with the original CPU implementation and benchmark values of a published 3D benchmark suite.

		Reflectance			Transmittance			Runtime (s)	
		Bench. [4]	CPU	GPU	Bench. [4]	CPU	GPU	CPU	GPU
1	a	0.4065(4)	0.4077(5)	0.4073(5)	0.0162(1)	0.0165(1)	0.0165(1)	8484.6	1.9
	b	0.0376(2)	0.0371(2)	0.0373(2)	0.00085(3)	0.00085(3)	0.00085(3)	1855.8	0.3
	c	0.4036(4)	0.4070(5)	0.4075(5)	0.0164(1)	0.0169(1)	0.0168(1)	8352.6	1.9
2	a	0.223(2)	0.2172(4)	0.2172(4)	0.0935(8)	0.0980(3)	0.0976(3)	9795.0	2.1
	b	0.161(2)	0.1440(4)	0.1431(4)	0.119(2)	0.1106(3)	0.1101(3)	9497.4	1.9
	c	0.3438(6)	0.3135(5)	0.3133(5)	0.1650(2)	0.1550(4)	0.1550(4)	14539.8	3.3
3	a	0.670(4)	0.6559(5)	0.6562(5)	0.169(3)	0.1835(4)	0.1828(4)	4092.6	1.1
	b	0.0167(6)	0.0129(1)	0.0129(1)	0.045(3)	0.0409(2)	0.0411(2)	646.8	0.2
	c	0.395(1)	0.3745(5)	0.3740(5)	0.085(3)	0.0810(3)	0.0811(3)	1648.8	0.4

The results for the simultaneous, successive, and hybrid cohorts with a variety of different numbers of threads were compared for cohort sizes ranging from 1 to 64. NVIDIA GPUs execute instructions in groups of 32 threads, also known as warps, in the SIMT (Single Instruction, Multiple Thread) fashion. To take full advantage of the warp and ensure that the particles are executed simultaneously (one particle per thread), simultaneous cohorts were only tested up to a cohort size of 32. Each simulation was run for 10^6 histories, independent of the cohort size. The root mean squared relative error, mean relative error, and max relative error across the benchmark set are shown in Fig. 2. The results for several approximate methods in the literature are also plotted, including AM [4], CLS [5], and PBS-2 [5]. AM is significantly less accurate than all the other methods described here, followed by our simultaneous and hybrid cohorts. CLS, PBS-2, and successive cohorts all yielded similar levels of accuracy, with PBS-2 being the most accurate for reflectance. CoPS with a small successive cohort was more accurate than PBS-2 but it became less accurate than PBS-2 with large or non-successive cohorts. It should be noted that only the CoPS2, and not the more accurate CoPSN, $N \geqslant 3$, algorithm was used for this analysis.

As the cohort size increases, the error also increases until it reaches a plateau. Since subsequent CPF evaluations depend on previously sampled points, error will generally compound as more CPF evaluations are made. The accuracy of computed mean and variance values as a function of cohort size was examined in Ref. [2], and it was observed that error in each of these quantities increases as cohort size increases, rapidly at first, but then appearing to approach an asymptotic error limit.

In summary, successive cohorts were more accurate than simultaneous cohorts by a mean relative error 4.3 times lower for reflectance and 5.9 times lower for transmittance, averaged across all benchmark problems and cohort sizes. Hybrid cohorts behaved similarly to simultaneous cohorts in terms of runtime and accuracy except when using 2-thread hybrid cohorts in which the behavior of results was between that of simultaneous and successive cohorts. The discrepancy in accuracy between simultaneous and hybrid cohorts and successive cohorts is likely due to the fact that in simultaneous and hybrid cohorts new CPF evaluations use only some of the points sampled by other histories whereas CPF evaluations in

successive cohorts use all of the points sampled by previous histories. Simultaneity may introduce more error in transmittance than reflectance since transmitting particles generally require more CPF evaluations involving points sampled by other histories than reflecting particles, thus exaggerating the aforementioned error.

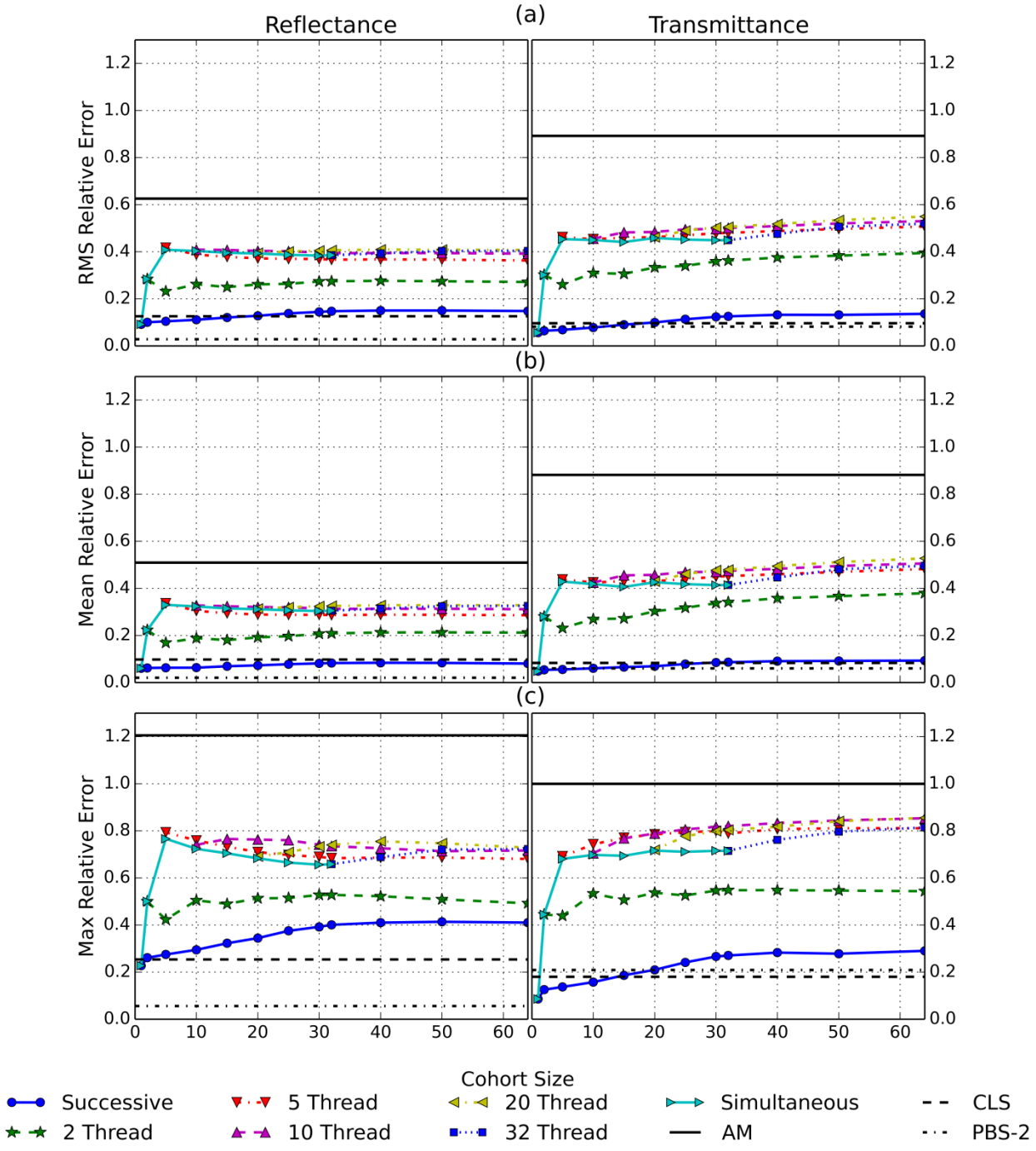

Figure 2. Benchmark results: (a) Root mean squared relative error, (b) Mean absolute relative error, and (c) Max absolute relative error across benchmark set

The mean runtimes across the benchmark sets are shown in Fig. 3. Unsurprisingly, simultaneous cohorts tend to be fastest, then hybrid, and then successive cohorts. It can be seen that the runtime per history

tends to increase in a linear fashion as the cohort size increases. This is expected, since the current CoPS2 implementation uses a linear search algorithm to find the point in a cohort nearest to the newly sampled point. As the cohort size increases, the number of points in a cohort and thus the time that the algorithm takes to find a new point's nearest neighbor also increases. Overall, all three cohort types are significantly faster than the original Python CPU implementation. Although simultaneous and hybrid cohorts currently introduce a large amount of error, they also represent a potential path forward towards achieving even greater computing efficiency.

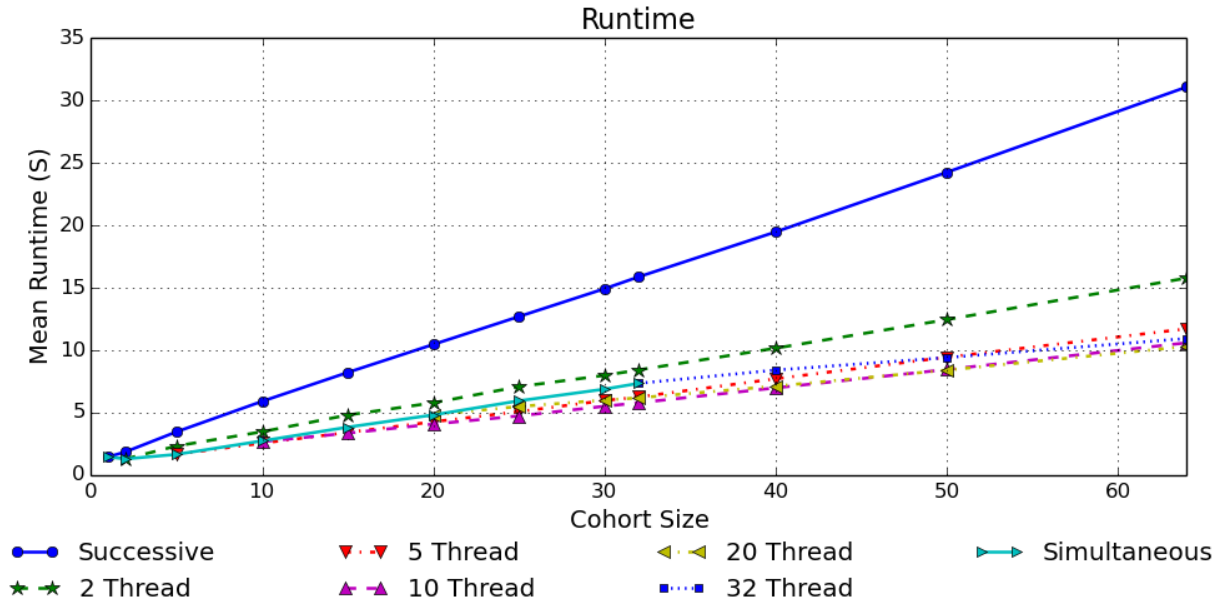

Figure 3. Benchmark 3a Runtimes

4. CONCLUSIONS

The GPU implementation of CoPS2 has been verified against the original CPU implementation and has been shown to match within statistical uncertainty. The C++ GPU implementation was significantly faster than the original Python CPU implementation, running an average of over 4000 times faster on benchmark problems. Testing for the GPU implementation was then extended for cohorts up to size 64 using successive, simultaneous, and hybrid cohorts. Simultaneous and hybrid cohorts were found to introduce a significant amount of error compared to successive cohorts, such that the mean relative error for simultaneous cohorts was, on average, higher by a factor of 4.3 for reflectance and 5.9 for transmittance. All three cohort versions ran significantly faster than the original CPU implementation, with simultaneous and hybrid cohorts giving an additional speed increase, achieving the greatest gains with the largest cohort sizes. On average, simultaneous cohorts were over 2 times faster than successive cohorts. Overall, successive cohorts were found to have the best accuracy-efficiency balance, though simultaneous and hybrid cohorts show the potential for greater speeds if their accuracy can be improved.

One of the main objectives of our future work is to implement more complex versions of CoPS for the GPU, including CoPS3 and CoPS4, which have both been shown to increase the accuracy of the algorithm [3]. However, there are several challenges yet to be resolved. One major limitation of the GPU implementation is the limited memory of the GPU. We found that we were limited to a maximum of about 10^6 total histories before the GPU ran out of memory. One possible solution to this constraint is to

train the algorithm to not keep all points from all of the histories that make up each cohort, which should result in a dramatic improvement in speed, if reasonable accuracy can be maintained. Work is currently being done to develop and test various "limited-memory methods" in CoPS, using an algorithm to choose which points in a cohort to keep and which ones to eliminate [7, 14]. It is possible that, with the introduction of limited memory techniques or more complex algorithms, like CoPS3 and CoPS4, simultaneous or hybrid cohorts could achieve competitive accuracy and even greater speedup. Though the GPU implementation is thread safe, our current results are only reproducible for successive cohorts because the CoPS algorithm is dependent on the order in which material points are sampled in a realization. Further work would be required to investigate making simultaneous and hybrid cohorts reproducible.

ACKNOWLEDGMENTS

Supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. The authors thank Dr. Jacquilyn Weeks for writing consultation in the preparation of this document.

REFERENCES

1. E. H. Vu and A. J. Olson. "Conditional Point Sampling: A novel Monte Carlo method for radiation transport in stochastic media." *Trans. Am. Nucl. Soc.,* **120**, 471-474 (2019).
2. E. H. Vu and A. J. Olson. "An extension of Conditional Point Sampling to quantify uncertainty due to material mixing randomness." In *M&C2019*. American Nuclear Society, Portland, OR (2019).
3. A. J. Olson and E. H. Vu. "An extension of Conditional Point Sampling to multi-dimensional transport." In *M&C2019*. American Nuclear Society, Portland, OR (2019).
4. C. Larmier, F. Hugot, F. Malvagi, A. Mazzolo, and A. Zoia. "Benchmark solutions for transport in d-dimensional Markov binary mixtures." *J Quant Spectrosc and Rad Transfer,* **189**, 133-148 (2017).
5. C. Larmier, A. Zoia, F. Malvagi, E. Dumonteil, and A. Mazzolo. "Poisson-Box Sampling algorithms for three-dimensional Markov binary mixtures." *J Quant Spectrosc and Rad Transfer,* **206**, 70-82 (2018).
6. E. Woodcock, T. Murphy, P. Hemmings, and T. Longworth, "Techniques used in GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry." In *Proceedings of the Conference on the Application of Computing Methods to Reactor Problems, ANL-7050*, Argonne National Laboratory, Chicago, IL (1965).
7. E. H. Vu and A. J. Olson. "Amnesia radius versions of Conditional Point Sampling for radiation transport in 1D stochastic media." In *M&C2021*. American Nuclear Society, Raleigh, NC (2021, accepted).
8. R. M. Bergmann and J. L. Vujic, "Algorithmic choices in WARP – A framework for continuous energy Monte Carlo neutron transport in general 3D geometries on GPUs," *Annals of Nuclear Energy*, **77**, 176-193 (2015)
9. K. L. Bossler, "Coupled electron-photon Monte Carlo radiation transport for next-generation computing systems." SAND2018-10539 (2018).
10. M. S. McKinley, R. Bleile, P. S. Brantley, S. Dawson, M. O'Brien, M. Pozulp, and D. Richards, "Status of LLNL Monte Carlo transport codes on Sierra GPUs." LLNL-CONF-773124 (2019).
11. S. P. Hamilton and T. M. Evans, "Continuous-energy Monte Carlo neutron transport on GPUs in the Shift code." *Annals of Nuclear Energy*, **128**, 236-247 (2019).
12. J. E. Sweezy, "A Monte Carlo volumetric-ray-casting estimator for global fluence tallies on GPUs." *Journal of Computational Physics*, **372**, 426-445 (2018).

13. T. P. Burke and F. B. Brown, “Development of a library for conducting Monte Carlo tallies on heterogeneous systems,” *Trans. Am. Nucl. Soc.,* **119** (2018).
14. E. H. Vu and A. J. Olson. “Recent memory versions of Conditional Point Sampling for radiation transport in 1D stochastic media.” *Trans. Am. Nucl. Soc.,* **123** (2020).