On the use of tally servers in Monte Carlo simulations of light-water reactors

SNA + MC 2013, 04301 (2014)
DOI: 10.1051/snamc/201404301

On the use of tally servers in Monte Carlo simulations of light-water reactors

Paul K. Romano[1*], Benoit Forget[1], Kord Smith[1], and Andrew Siegel[2]

[1]*Massachusetts Institute of Technology, Department of Nuclear Science and Engineering, 77 Massachusetts Avenue, Cambridge, MA 02139*
[2]*Argonne National Laboratory, Theory and Computing Sciences, 9700 S Cass Ave., Argonne, IL 60439*
[*]Corresponding Author, E-mail: paul.k.romano@gmail.com

An algorithm for decomposing tally data in Monte Carlo simulations using servers has recently been proposed and analyzed. In the present work, we make a number of refinements to a theoretical performance model of the tally server algorithm to better predict the performance of a realistic reactor simulation using Monte Carlo. The impact of subdividing fuel into annular segments on parameters of the performance model is evaluated and shown to result in a predicted overhead of less than 20% for a PWR benchmark on the Mira Blue Gene/Q supercomputer. Additionally, a parameter space study is performed comparing tally server implementations using blocking and non-blocking communication. Non-blocking communication is shown to reduce the communication overhead relative to blocking communication, in some cases resulting in negative overhead.

KEYWORDS: Monte Carlo, data decomposition, tally server, LWR, OpenMC

I. Introduction

Typical parallel implementations of Monte Carlo particle transport rely on full replication of the problem data on each process. This approach has been shown to be highly scalable,[(1)] but does not lend itself to problems where the memory requirements exceed that of a single node. For the realistic analysis of light-water reactors (LWRs), the memory requirements can be quite severe. Neutron interaction cross sections, which need to be stored for over 400 nuclides at various temperatures, may consume up to 100 gigabytes of memory for a practical simulation.[1] For a robust depletion calculation, the required tally memory is likely to exceed 0.5 terabyte.[(4)] Treating realistic tally memory footprints thus requires some form of decomposition across compute nodes. Two decomposition methods have been proposed previously for addressing this problem: *domain decomposition*[(4,5)] and *data decomposition*.[(6)]

In a recent paper,[(7)] Romano et al. demonstrate an implementation of data decomposition via a tally server algorithm and show that it offers a viable means of performing full core light-water reactor simulations via Monte Carlo. A theoretical model was developed to predict the performance of a simulation using the tally server algorithm relative to a simulation based on full memory replication. The model depends on a number of machine-, code-, and problem-specific parameters. In the present work, we revisit the derivation of the expected performance and make refinements to a number of assumptions and parameters. The goal is to develop a more realistic expectation for the performance of the tally server algorithm specifically when applied to simulation of LWR problems.

[1] A number of novel algorithms may ultimately enable simulations involving continuous temperature distributions to be performed using cross sections at 0 K.[(2,3)]

II. Tally Server Model

During a Monte Carlo simulation, estimates of integral physical parameters, referred to as *tallies*, are made by keeping running sums of scores from events such as collisions or particle tracks. Normally, tallies are stored in local memory. Synchronization between processors is typically performed only after simulating a predetermined number of particles, referred to as a *batch*. However, since tally data is not needed for determining the random walk of a particle, it can be stored remotely.

In the tally server algorithm, the tally data is stored in the address space of a process whose sole purpose is to receive scores from other processes (which we call *tally servers*) and increment the tallies accordingly. Thus, a total of p processes are divided into c compute processes and s tally servers. Each of the compute processes is assigned a set of particles that it will track. As particle tracking is simulated, an array of scores is sent to a tally server at each event that results in a contribution to a tally. Since all tally accumulation is performed on the server, the compute processes do not need to store the tallies in memory (other than meta-data describing the tally).

The goal of the analysis here is to develop a model for the expected time to simulate N particles using the tally server algorithm relative to a classic simulation with no data decomposition. To that end, we first define a number of parameters:

μ = particle tracking rate [1/second],
f = number of tallying events per particle,
d = tally data sent per event [bytes],
α = application-level latency [seconds],
β = application-level inverse bandwidth [seconds/byte].

The latency and inverse bandwidth are determined by the network interconnect; f, μ, and d will depend on the machine

hardware as well as the code being used and the model being simulated. Thus, while these parameters may be hard to predict, they can easily be measured from an actual simulation. Once these parameters are known, we can develop a rough estimate for the time-to-solution with and without tally servers. In a normal simulation without tally servers, the expected time to simulate N particles is, assuming perfect parallel scaling,

$$t_0 = \frac{N\mu}{p}. \tag{1}$$

When the tally server algorithm is used, there are two sources that lead to overhead: 1) availability of fewer processors to simulate particles and 2) network communication for tally data from compute processes to the servers. The expected simulation time when using tally servers is identical to the expression in Equation 1 but with p replaced by c:

$$t_c = \frac{N\mu}{c}. \tag{2}$$

Since $f(\alpha + d\beta)$ is the expected tally server communication time for one particle and N/c is the number of particles per processor, the total expected communication time is

$$t_s = \frac{fN}{c}(\alpha + d\beta). \tag{3}$$

We then define $t = t_c + t_s$ as the total simulation time using tally servers. Combining Equation 1, Equation 2, and Equation 3, we obtain an expression relating the simulation time with and without tally servers:

$$\frac{t}{t_0} = \frac{p}{c}\left[1 + \frac{f}{\mu}(\alpha + d\beta)\right]. \tag{4}$$

The first factor on the right-hand side of Equation 4, p/c represents the loss in efficiency due to having fewer processes tracking particles. The remaining term within the square brackets represents the loss in efficiency due to the necessary network communication. In this work, we will primarily be concerned with the communication overhead,

$$\Delta_s = \frac{f}{\mu}(\alpha + d\beta). \tag{5}$$

1. Model Refinements

In the previous work by Romano et al.,[7] estimates of the tally server parameters were made by analyzing a hypothetical depletion simulation of the Monte Carlo Performance Benchmark[8] on two target supercomputers: the Titan Cray XK7 at Oak Ridge National Laboratory and Intrepid Blue Gene/P at Argonne National Laboratory. For the sake of simplicity, some of the assumptions made in estimating these parameters were not conservative. We now revisit those assumptions to develop more realistic estimates to determine what effect, if any, they have on the expected performance of the tally server algorithm on a modern supercomputer.

1.1. Target Model

Rather than look at the Monte Carlo Performance Benchmark, which contains many unrealistic simplifications (e.g., no fuel enrichment zoning and no control rods), we have chosen as our target problem the BEAVRS PWR benchmark model.[9] This model includes accurate enrichment loadings, burnable absorber patterns, and control bank positions as well as faithfully-modeled axial grid spacers, core baffle structures, neutron shield panel structures, and relevant core internals. The use of a different model will have an impact on μ and f. For Mira Blue Gene/Q, the particle tracking rate for the benchmark is about $1/\mu = 69$ particle/s. This is very similar to the particle tracking rate for the Monte Carlo performance benchmark on Blue Gene/P, and the number of tracks in fuel is virtually the same at $f = 21$. Assuming the same physical quantities need to be tallied, d will not change. As in our previous work,[7] a range of d will be investigated.

1.2. Annular Regions in Fuel

In a depletion simulation, six reaction rates for each nuclide must be tallied each time a particle track crosses fuel. Furthermore, it is necessary to subdivide fuel regions into annular segments since spatial self-shielding will result in the outer part of a fuel pin depleting faster than the inner part. The impact of this subdivision of the fuel on f and μ has not previously been accounted for. With an increasing number of subdivisions, the number of events that will result in contributions to tallies will increase. At the same time, the time to simulate a single particle will increase since there will be more surface crossings, re-evaluation of cross sections, and tallying events.

To explicitly determine the effect of fuel subdivision on f and μ, a series of simulations were run using the OpenMC Monte Carlo code[1] on the BEAVRS PWR benchmark model varying the number of annular regions in the fuel from 1 to 10. Figure 1 shows the dependence of f on the number of annular regions. While not intuitively obvious *a priori*, this figure demonstrates that the number of tracks in fuel is directly proportional to the number of annular regions.

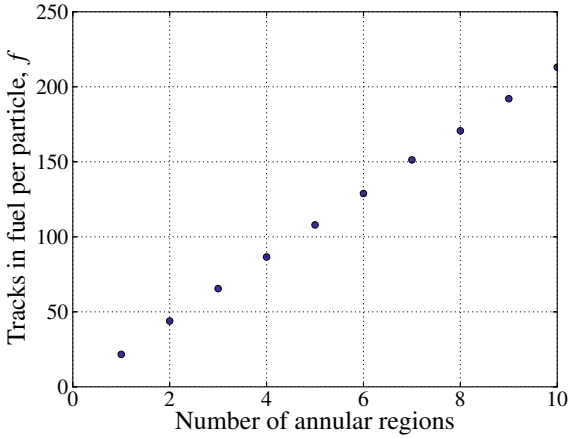

Figure 1: Number of tracks in fuel as a function of the number of annular regions.

In OpenMC, each time a particle enters a new material, the macroscopic cross sections must be calculated. This is true

even if the particle hasn't changed energy. Thus, as the number of annular segments in fuel increases, the calculation time will increase due predominantly to the extra cross section evaluations. Figure 2 shows the dependence of μ on the number of annular regions as measured by OpenMC running on an Intel Core i5 Processor. While the relative simulation time also increases linearly with the number of annular regions, unlike the number of tracks the two are not directly proportional since only a fraction of the simulation time is spent tracking particles in fuel.

1.3. Network Interconnect

For the present analysis, rather than looking at the Titan Cray XK7 or Intrepid Blue Gene/P supercomputers, our target architecture is the Mira Blue Gene/Q supercomputer at Argonne National Laboratory. Mira has 48 racks, each with 1024 nodes containing a 16-core PowerPC A2 processor for a total of 768,432 processor cores. More importantly, the Blue Gene/Q network interconnect utilizes a 5D torus and has lower latency and high bandwidth than the interconnect used for Blue Gene/P. The nearest-neighbor MPI latency has been observed to be about 2.0 µs[10] and the maximum-hop latency is about 3.0 µs.[11] In our analyses we assume an average latency of $\alpha = 2.5\,\mu s$. The internode single link bandwidth is about 1.8 GB/s.[11] Consequently, we will use $\beta = 5.55 \cdot 10^{-10}$ s/byte. Table 1 gives a summary of the parameters used in the model predictions for the tally server overhead as well as those used in our previous work.[7]

Table 1: Parameters used for tally server overhead model

Parameter	Description	Intrepid	Mira
α	Latency (s)	$3.53 \cdot 10^{-6}$	$2.5 \cdot 10^{-6}$
β	Bandwidth (s/byte)	$2.60 \cdot 10^{-9}$	$5.55 \cdot 10^{-10}$
$1/\mu$	Particles/second	76	69
d	Data/event (bytes)	0 – 15,360	0 – 15,360
f	Events/particle	21	21–213

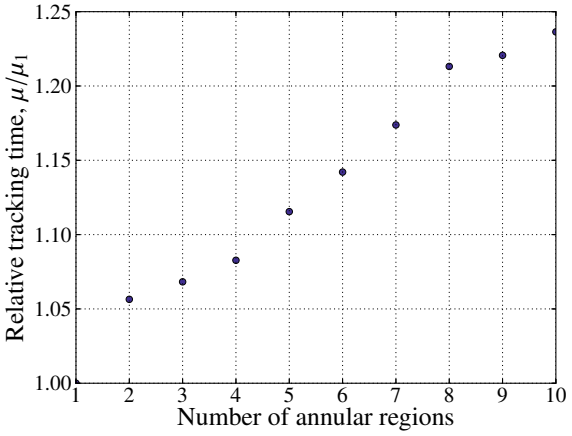

Figure 2: Relative simulation time per particle as a function of the number of annular regions.

1.4. Predicted Overhead

As discussed earlier, the increase in simulation time when using tally servers can be attributed to 1) having fewer processors tracking particles and 2) network communication. The first factor that increases the simulation time is known and is simply determined by the user's choice of p, c, and s. Thus we will evaluate only the overhead from network communication as given in Equation 5.

In the previous section, we demonstrated that when subdividing the fuel pins into annular regions, the number of tallying events per particle f is directly proportional to the number of annular regions, whereas μ increases only slightly. Thus, the communication overhead based on (5) will increase almost in direct proportion to the number of annular regions. Figure 3 shows the predicted overhead on the Mira supercomputer as a function of d for varying numbers of annular regions based on the results in Figure 1 and Figure 2. The upper limit on d is 15,360 bytes, the amount of tally data for six reaction rates in each of 320 nuclides within a material. The latency and bandwidth of the interconnect were taken from Table 1. Even when 10 annular regions in the fuel are modeled, the maximum predicted communication overhead is still under 20%.

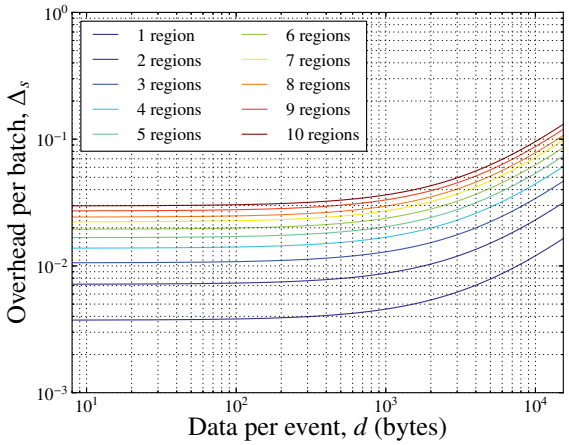

Figure 3: Estimated tally server overhead on the Mira Blue Gene/Q supercomputer as a function of the number of annular regions.

1.5. End-of-batch Accumulation

One aspect of the algorithm that was not previously accounted for in the model of overhead is the accumulation of tallies at the end of a batch. For statistical purposes, after a set of N neutrons are simulated, the accumulated score for each tally random variable is added to a running sum, and the square of the accumulated value is added to a sum of squares. These sums enable the sample variance to be calculated at the end of the simulation. When a tally server algorithm is used, the task of incrementing these two sums is shifted from the compute processes to the servers. Said another way, the total amount of work the compute processes must perform is reduced slightly. As a result, the reduced work may partially or completely negate the network communication overhead.

To model this effect, we break up the average time to simulate

N particles into two components, $N\mu = N\mu_t + \mu_b$, where μ_t is the average time to transport a particle and μ_b is the average time to calculate sums and sums-of-squares. Since μ_b is directly proportional to the total number of tally scores, which in turn is typically proportional to d, we can express it as $\mu_b = \mu_b' d$. Without tally servers, the total time to simulate N particles on p processors becomes

$$t_0 = \frac{N\mu_t + \mu_b' d}{p}. \tag{6}$$

When tally servers are used, the time spent incrementing the sums is offloaded to the servers. Thus, the total tracking time on c compute processes is

$$t_c = \frac{N\mu_t}{c} \tag{7}$$

As before, when we combine Equation 6, Equation 7, and Equation 3, we obtain an expression relating the simulation time with and without tally servers:

$$\frac{t}{t_0} = \frac{p}{c}\left[\frac{\mu_t + f\,(\alpha + d\beta)}{\mu_t + \frac{\mu_b' d}{N}}\right]. \tag{8}$$

The communication overhead, defined earlier as the bracketed term minus unity, now includes a term in the denominator that will increase with d:

$$\Delta_s = \frac{\mu_t + f\,(\alpha + d\beta)}{\mu_t + \frac{\mu_b' d}{N}} - 1. \tag{9}$$

According to Equation 9, it is possible for the communication overhead to be negative if $Nf(\alpha + d\beta) < \mu_b' d$. If d is sufficiently large that the latency is negligible ($\alpha \approx 0$), then the condition for negative overhead becomes $Nf\beta < \mu_b'$. While this condition no longer depends on d, μ_b' can still increase if the total number of tally score bins is increased (e.g., by refining a mesh over which scores are being tallied). Figure 4 shows the predicted overhead on the Mira supercomputer as a function of d for the original model in Equation 5 and the modified model in Equation 9. The parameters μ_t, α, and β are all from Table 1 and it was assumed that $\mu_b'/N = 50$ ns/byte. This value was chosen merely to demonstrate that negative overhead is possible and that μ_b' need not be exceedingly large. For small values of d, the overhead is dominated by the latency term. For larger values of d, Equation 5 results in an increasing overhead due to the bandwidth term whereas Equation 9 results in decreasing overhead since $Nf(\alpha + d\beta) < \mu_b' d$.

To summarize, there are two key takeaways:

1. Negative overhead is possible due to offloading the incrementing of tally sums and sums-of-squares to the tally servers and is more likely to occur when a large number of quantities are being tallied.

2. In practice, the beneficial effect of offloading this operation may be masked by large N. Particularly in reactor simulations where it is expected that a single batch of neutrons may exceed one billion neutrons, it is unlikely that negative overhead could be achieved.

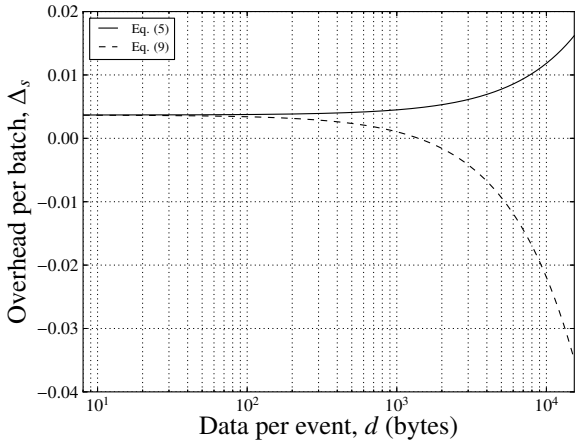

Figure 4: Estimated tally server overhead accounting for accumulation.

The foregoing analysis has thus far assumed that network communication is blocking. However, if non-blocking communication is used, the communication operations may overlap with computation. In the best case scenario, the non-blocking sends from compute processes would return instantaneously implying that $t_s = 0$. This in turn would imply that

$$\Delta_s = \frac{\mu_t}{\mu_t + \frac{\mu_b' d}{N}} - 1 = -\frac{\mu_b' d}{N\mu_t + \mu_b' d}. \tag{10}$$

We see here that with non-blocking communication, negative overhead is possible regardless of the size of N.

While non-blocking communication may reduce the network communication overhead at the sender to a level that is negligible or even negative, it's important to keep in mind that the time to complete a batch of neutrons is still limited by the lesser of the time the compute processes require to transport the particles and the time the tally servers require to accumulate tallies. The latter time is constrained in the sense that an excessively large support ratio, c/s, would result in network contention at the tally servers. For the tally server, there is no computation to be performed and thus no opportunity to overlap communication and computation—handling communication is the sole purpose of the server. In light of this, the latency and bandwidth of the network are still crucial parameters that have a bearing on the feasibility of the tally server algorithm.

III. Results

A complete implementation of the tally server algorithm in the OpenMC Monte Carlo code was previously described by Romano et al.[7] The initial implementation, which was based on blocking communication, was tested over a wide range of parameters on two supercomputers: the Titan Cray XK7 supercomputer at ORNL and the Intrepid Blue Gene/P supercomputer at ANL. It was argued based on the performance model that in the limit of an optimal support ratio, the use of non-blocking communication could reduce the total overhead by a factor of two, but such an implementation was never tested. Since then, a tally server algorithm based on non-blocking communication has been implemented in a branch of OpenMC.

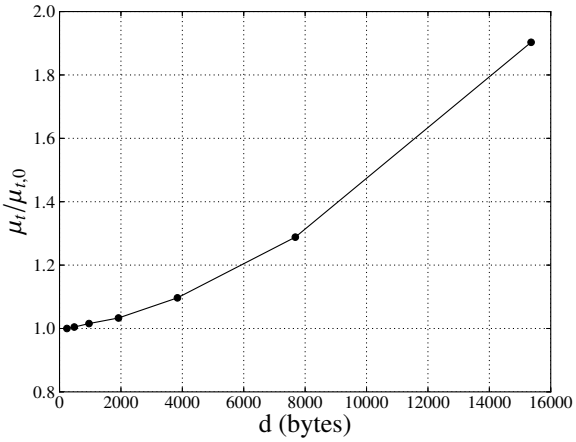

Figure 5: Observed dependence of μ_t on the amount of data tallied, d, on Mira.

The performance model developed in section II depends on a variety of parameters. For our target system, the Mira Blue Gene/Q supercomputer, α, β, and μ are constant and can be determined based on measured data as previously discussed. The remaining parameters are manipulated by varying the definition of the tallies and the job parameters. To fully test the performance of the non-blocking tally server implementation, a parameter study was performed that covers a range of the parameters p, s, and d. For the present work, we have focused specifically on the dependence of the communication overhead on d for varying support ratios c/s, total number of processors p, and a fixed f. As expressed in Equation 9, we do not expect the overhead to vary with either the support ratio or the total number of processors—nevertheless we have chosen to include them as parameters since any limitation to the scalability of the algorithm is likely to show up as a trend with p or c/s.

To begin, a number of baseline simulations of the BEAVRS benchmark were run without tally servers to determine the dependence of μ_t on d. These simulations were run on Mira with 16 processors and a total of 32,000 particles per batch. Ten batches were run both without tallies (referred to as *inactive batches*) and with tallies (*active batches*). For each case, a tally was set up with a mesh filter and a second filter to match only events within the fuel volume. Six reaction rates were tallied for varying numbers of nuclides, starting with 5 nuclides and doubling the number of nuclides up to 320. Thus, the amount of data sent at each event varied from 240 bytes up to 15.36 kilobytes. Figure 5 shows the observed dependence of μ_t on d normalized to the $d = 5$ case.

The parameter study using tally servers on the Mira supercomputer consisted of two sets of 168 simulations with each combination of the following parameters: $p = 16, 32, 64, 128, 256, 512$, $c/s = 1, 3, 7, 15$, and $d = 240, 480, 960, 1920, 3840, 7680, 15360$. The first set was performed with blocking communication between the compute processes and the servers and the second set with non-blocking communication. Like the baseline cases, the runs with tally servers had 10 inactive batches, 10 active batches, and $N/p = 500$. The effective overhead from tally servers was determined in the following manner. First, the expected overhead due to looking up cross sections during tallying was subtracted from the active batch time based on the results from the baseline cases. Then, the adjusted simulation time in active batches was divided by the inactive batch time to determine the overhead in active batches. The result is a quantity that is a proxy for the communication overhead, Δ_s. One should take note that it does not account for the fact that we have fewer compute processes. The overhead calculated in this manner for $c/s = 1$, $c/s = 3$, $c/s = 7$, and $c/s = 15$ is shown in Figure 6, Figure 7, Figure 8, and Figure 9, respectively.

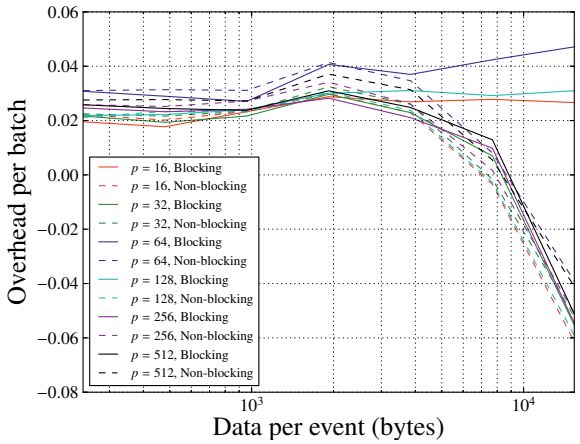

Figure 6: Observed tally server overhead on ANL Mira with 1 compute process per server.

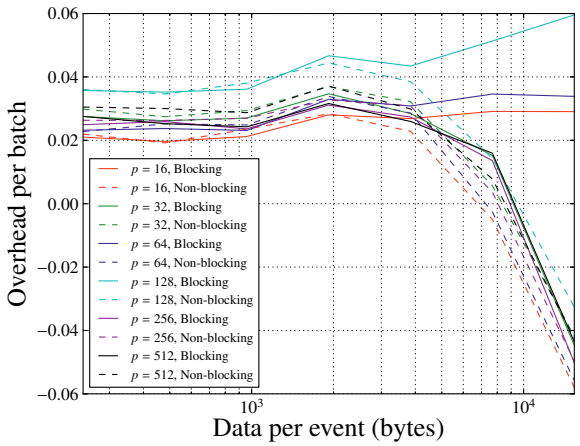

Figure 7: Observed tally server overhead on ANL Mira with 3 compute process per server.

Compared to our previous study, the observed communication overhead is lower for large d primarily due to the higher bandwidth on Mira compared to Titan or Intrepid. In all cases, the communication overhead is less than 6%, whereas for Intrepid and Titan it had exceeded 30% in some cases. A more striking feature in all the results is the fact that all non-blocking cases exhibit a clear trend of increasingly negative overhead for large d. Based on the previous discussion, this is a direct consequence of the fact that the incrementing of tally sums and sums-of-squares has been offloaded to the tally servers. Had the choice of N been larger, this effect would have been mitigated. That negative overhead could be observed at all is a

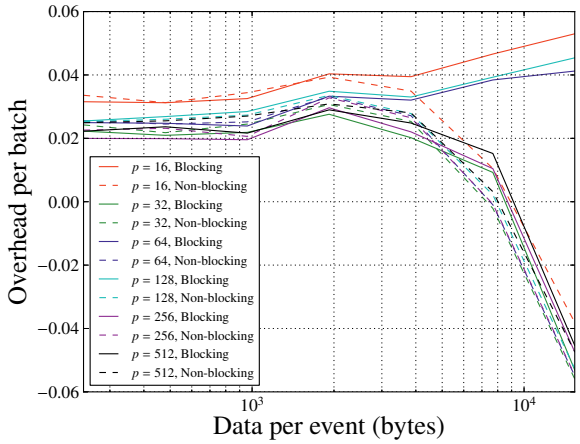

Figure 8: Observed tally server overhead on ANL Mira with 7 compute process per server.

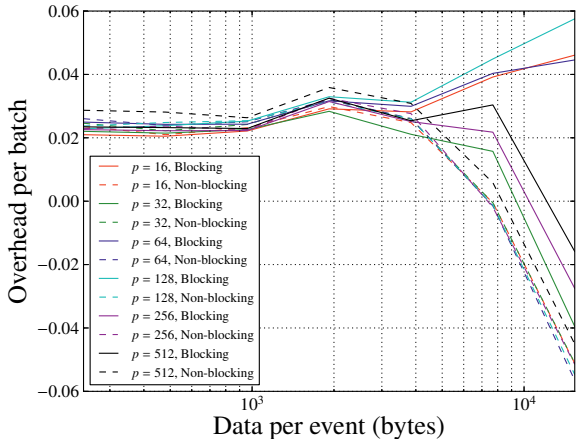

Figure 9: Observed tally server overhead on ANL Mira with 15 compute process per server.

testament to the inherently fast network interconnect on Mira which results in little overhead, especially when non-blocking semantics are used.

It is also of interest to observe the behavior of the tally server overhead with increasing numbers of total processors. According to the performance model, the overhead should not depend on the number of processors used. Figure 10 shows the overhead plotted as a function of p for cases with $d = 15360$. For the simulations where blocking communication was used, there is no clear trend with p. The overhead when using 16, 64, and 128 total processor cores was consistently positive whereas the overhead turned negative for 32, 256, and 512 total processors. Despite the odd behavior with changes in p, there was little variation as a function of the support ratio, c/s. When non-blocking communication was used, the overhead was consistently negative for all cases.

IV. Conclusions

In the present work, we have made further inroads towards evaluating the potential for the tally server data decomposition algorithm to be applied to Monte Carlo simulations of light-water

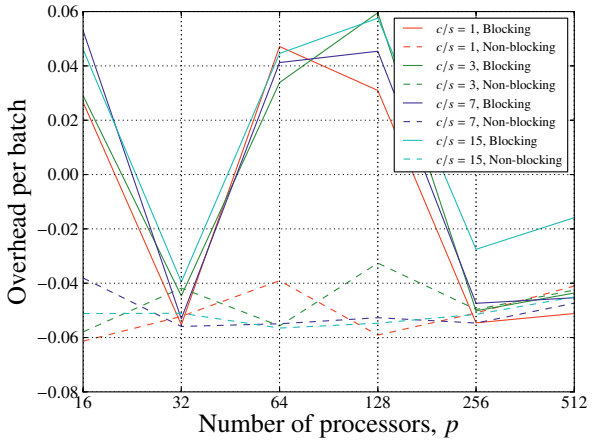

Figure 10: Observed tally server overhead on ANL Mira as a function of p with $d = 15360$.

reactors. The two major contributions are 1) improvements in the theoretical performance model, and 2) a thorough parameter space study looking at the impact of blocking vs. non-blocking communication in a real tally server implementation in the OpenMC Monte Carlo code.

In our previous work, the Monte Carlo performance benchmark, otherwise known as the Hoogenboom-Martin benchmark, was used as the basis for evaluating tally server performance model parameters. That benchmark model was overly simplified, and the recent introduction of a more realistic PWR benchmark, BEAVRS, has allowed us to re-evaluate the model parameters. The change of benchmark models did not have a significant effect on any model parameters. The simplified theoretical model that was developed previously has also been refined to better predict the performance of a realistic reactor simulation. Most importantly, the effect of fuel subdivision on the number of particle tracks and calculation rate for the BEAVRS benchmark was quantified using OpenMC. It was shown that the predicted overhead due to tally servers increases linearly with the number of annular regions in fuel. Nevertheless, even with 10 regions, the predicted overhead of using tally servers is less than 20% on the Mira supercomputer over a wide parameter regime. Thus, the subdivision of fuel pins into unique depletion regions should not be a major impediment towards achieving high-fidelity simulations that rely on tally servers.

A modified implementation of the tally server algorithm in OpenMC using non-blocking communication was tested on the Mira supercomputer along with the original implementation based on blocking communication. The observed communication overhead was reduced when using non-blocking communication as previously predicted. Furthermore, the communication overhead decreased to the point that it was negative as the amount of data being sent at each tally event increased. This was attributed to the accumulation of tally scores at the end of a statistical batch being offloaded to the tally servers rather than being performed by the compute processes. It is important to recognize that the negative overhead observed is a consequence of the particular choice of run parameters and would be unlikely to occur in a hypothetical reactor depletion

simulation where the total number of particles per statistical batch is necessarily very large, thus reducing the importance of any end-of-batch operations.

The basic conclusions of our previous work, i.e., that the tally server algorithm is a successful approach to circumventing on-node memory constraints associated with detailed Monte Carlo reactor simulations, in unchanged in light of the evidence presented in this work. While the tally server algorithm could already be employed on the world's fastest supercomputers today, the need for an extremely fast network interconnect means that it may not be amenable for use on commodity computer architectures that would more likely be used by scientists for day-to-day work.

Acknowledgments

This research was performed under appointment of the first author to the Rickover Fellowship Program in Nuclear Engineering sponsored by Naval Reactors Division of the U.S. Department of Energy. This work was also supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

References

1) P. K. Romano and B. Forget, "The OpenMC Monte Carlo Particle Transport Code," *Ann. Nucl. Energy*, **51**, 274–281 (2013), doi:10.1016/j.anucene.2012.06.040.
2) G. Yesilyurt, W. R. Martin, and F. B. Brown, "On-the-Fly Doppler Broadening for Monte Carlo codes," *Nucl. Sci. Eng*, **171**, 239–257 (2012).
3) T. Viitanen and J. Leppänen, "Explicit Treatment of Thermal Motion in Continuous-Energy Monte Carlo Tracking Routines," *Nucl. Sci. Eng.*, **171**, 165–173 (2012).
4) A. R. Siegel, K. Smith, P. K. Romano, B. Forget, and K. Felker, "The effect of load imbalances on the performance of Monte Carlo codes in LWR analysis," *J. Comput. Phys.*, **235**, 901–911 (2013), doi:10.1016/j.jcp.2012.06.012.
5) A. Siegel, K. Smith, P. Fischer, and V. Mahadevan, "Analysis of communication costs for domain decomposed Monte Carlo methods in nuclear reactor analysis," *J. Comput. Phys.*, **231**, 3119–3125 (2012), doi:10.1016/j.jcp.2011.12.014.
6) F. B. Brown and W. R. Martin, "High Performance Computing and Monte Carlo," *Trans. Am. Nucl. Soc.*, **91**, *1*, 279–280 (2004).
7) P. K. Romano, A. R. Siegel, B. Forget, and K. Smith, "Data decomposition of Monte Carlo particle transport simulations via tally servers," *J. Comput. Phys.*, **252**, 20–36 (2013), doi:10.1016/j.jcp.2013.06.011.
8) J. E. Hoogenboom, W. R. Martin, and B. Petrovic, "The Monte Carlo Performance Benchmark Test - Aims, Specifications and First Results," *Int. Conf. Math. Comput. Methods Applied to Nucl. Sci. Eng.*, Rio de Janeiro, Brazil, May 8–12, 2011.
9) N. Horelik, B. Herman, B. Forget, and K. Smith, "Benchmark for Evaluation and Validation of Reactor Simulations (BEAVRS)," *Int. Conf. Mathematics and Computational Methods Applied to Nuclear Science and Engineering*, Sun Valley, Idaho, May 5–9, 2013.
10) J. Hammond, "Mysteries of the Deep: What happens inside of MPI on Blue Gene/Q and why it matters," *Leap to Petscale Workshop*, Argonne, Illinois, May 22–25, 2012, `http://www.alcf.anl.gov/sites/www.alcf.anl.gov/files/JeffsL2Ptalk_0.pdf`.
11) K. Kumaran, "Introduction to Mira," *Code for Q Workshop*, Argonne, Illinois, April 30–May 2, 2012, `http://www.alcf.anl.gov/sites/www.alcf.anl.gov/files/bgq-perfengr.pdf`.