The effect of load imbalances on the performance of Monte Carlo codes in LWR analysis

A. R. Siegel[a], K. Smith[c], P. K. Romano[c], B. Forget[c], K. Felker[b]

[a]*Argonne National Laboratory, Theory and Computing Sciences and Nuclear Engineering Division*
[b]*Argonne National Laboratory, Theory and Computing Sciences*
[c]*Massachusetts Institute of Technology, Department of Nuclear Science and Engineering*

Abstract

A model is developed to predict the impact of particle load imbalances on the performance of domain-decomposed Monte Carlo neutron transport algorithms. Expressions for upper bound performance "penalties" are derived in terms of simple machine characteristics, material characterizations and initial particle distributions. The hope is that these relations can be used to evaluate tradeoffs among different memory decomposition strategies in next generation Monte Carlo codes, and perhaps as a metric for triggering particle redistribution in production codes.

Keywords: Monte Carlo neutron transport reactor analysis load balancing

1. Background

Particle based Monte Carlo (MC) methods for neutron transport are becoming an increasingly active area of research in the reactor physics community. Traditional reactor core analysis codes are based on various discretizations of the deterministic transport equation, viz. so-called PN, SN, and Method of Characteristics (MOC) formulations. Monte Carlo methods, however, offer several potential advantages to these traditional formulations,

Email addresses: siegela@mcs.anl.gov (A. R. Siegel), kord@mit.edu (K. Smith), romano7@mit.edu (P. K. Romano), bforget@mit.edu (B. Forget), felker@mcs.anl.gov (K. Felker)

particularly in areas that tend to encumber the practical adoption of transport tools to new classes of problems – viz. the avoidance of complex meshing for complicated geometries, the simplification of the cumbersome multigroup cross section generation process, and, perhaps most importantly, the potentially easier adaptability to the extreme levels of concurrency that are likely to characterize beyond-petascale HPC architectures [1]. Reactor MC simulations, though, are notoriously expensive computationally [2, 3, 4], and a number of algorithmic and implementation challenges remain before they can be robustly applied to practical reactor analysis.

One of these challenges involves developing parallel methods with dramatically reduced per-node memory footprints (e.g. [5]). Existing production MC codes for reactor analysis are either implemented serially or carry out parallelization by simple replication – i.e. distributing batches of particles to processors while replicating and synchronizing the key data structures across nodes. This approach is relatively easy to implement and has excellent scalability properties. However, for robust reactor calculations, the required memory footprint in general far exceeds node-level memory resources. Thus, even the largest reactor MC calculations to date have been forced to employ a number of simplifications and approximations in order to fit standard reactor physics calculation in memory [4].

In reactor applications the majority of memory is consumed by two types of data – 1) interaction cross section libraries and 2) interaction tallies for pre-defined geometric regions [2]. The cross section data libraries are read into main memory and accessed randomly during the tracking of an individual neutron. This data needs to be accessible to each process and can be more or less significant depending on the specific application. Standard Light Water Reactor (LWR) analyses, for example, require data for several reaction types for each of 200–400 isotopes at approximately 50 temperature intervals and 200,000 energy points. At eight bytes per data point this memory can exceed 100GB. Existing research into compact functional representations has the potential to reduce this footprint significantly, but this is currently a topic of research and developing algorithmically creative ways of handling cross section data remains an important technique for reducing the total memory cost [6].

For LWR applications in particular, the tallies are an even more significant consumer of memory. We conservatively estimate that a robust LWR analysis would require approximately a factor of fifty greater than the 8GB reported in the landmark calculations of Kelly et al. [4] – specifically, an additional 5 axial levels and and a factor of 10 in radial rings per fuel pin (to treat radial temperature profiles and self-shielded absorbers) – for a total of 500GB of memory. Unlike the cross section libraries, tally data is write only and requires a relatively small amount of data to be transferred for each update.

In this paper we focus on algorithmic issues related to the reduction of the local tally memory. Two classes of strategies have been proposed to address this problem – what we loosely refer to as *data decomposition* and *domain decomposition*. Data decomposition could take many forms, but the basic approach would involve the same naturally parallel by-particle parallelization strategy as is currently employed together with a set of dedicated *tally servers* that continuously receive tally updates from the tracking processors (e.g. with one-side operations or by running a continuous receive loop). The spatial decomposition of the tallies is in general arbitrary and the data sent from the tracking processors could be carried out with non-blocking MPI send operations to maximize overlap in communication/computation. Of course, this idea could be extended to non-disjoint processor sets, but for illustrative purposes it is perhaps easiest to imagine them as distinct.

Domain decomposition on the other hand associates a contiguous region of physical space with each processing element (or node). Each processor then owns the subset of tallies for the corresponding region of physical space. This approach is potentially made efficient with a more complex parallelization strategy where each processor only tracks the particles that are passing through its own portion of the domain. However, new complexities emerge – fast neutrons travel long distances before absorption, and thus the local leakage rates (probability of a neutron leaving a partition before being absorbed) are relatively large. This requires intermediate data exchange *stages* where particles are moved to adjacent processors, potentially leading to significant communication costs and non-trivial load imbalances.

In a recent work by Siegel et al. [7], the authors attempt to quantify this cost, exploring the feasibility of carrying out efficient domain decomposition in a parameter regime relevant to Light Water Reactor (LWR) analysis. Key scaling regimes are identified and performance estimates are carried out over a range of characteristic parameters. The results demonstrate that for the model problem good performance is expected for reasonable effective latency and bandwidth values for partition particle densities as low as 10^4 per node (e.g. on the BG/P platform). This analysis takes an important step toward quantifying the key theoretical issues and establishing the feasibility of carrying out robust LWR calculations with MC methods.

However, the analysis in [7] does not address the impact of initial particle imbalances and small variations in local leakage rates on evolving particle distributions. If particle communication costs are potentially manageable in light of typical machine bandwidth and latencies, to what extent will load imbalances negatively impact performance to the point where such an approach is no longer a practical alternative? Establishing a quantitative foundation for this performance penalty will greatly facilitate weighing tradeoffs in deciding the best path forward among various approaches for next-generation MC codes. In this paper we address this issue directly by attempting to quantify the effect of particle load imbalances and typical material inhomogeneities on the performance of domain decomposed MC algorithms.

2. Analysis

2.1. Problem definition

To fully define the problem, begin with a domain X segmented into N non-overlapping partitions $X = \cup x_j, j = 1...N$. If we denote the initial number of particles on partition x_j as $p_{0,j}$, then the initial global particle count P_0 is given by:

$$P_0 = \sum_{j=1}^{N} p_{0,j}. \tag{1}$$

For simplicity, assume that each partition is mapped to a single processing element on a three-dimensional virtual Cartesian topology of N processors. In this work we thus use the terms *processor*, *partition*, and *process* interchangeably.

On a given partition, particles owned by that partition are advanced through a sequence of interactions until they are either 1) absorbed or 2) reach a processor boundary. For optimal performance particles that reach the boundary are buffered locally until all particle trajectories are computed, and are subsequently exchanged with neighbor processors. We refer to the particle exchange phase of this process as a *stage*, and to a complete set of stages (i.e. until all particles are absorbed) as a *cycle*. At the end of each cycle, the fission source is updated and the next cycle begins.

Define the particle *local leakage* λ on each x_j for a given stage i as

$$\lambda_{i,j} = \frac{\text{number of particles leaving } x_j \text{ at stage } i}{\text{number of particles starting in } x_j \text{ at stage } i}.$$

The goal of this analysis is to estimate the impact of initial particle load imbalances and spatially varying local leakage values on previous performance estimates of domain-decomposed LWR simulations. We refer to the perfectly load balanced, constant leakage scenario as the *ideal* case. This was explored in depth in [7], where it was shown that communication costs imposed a reasonable penalty on overall simulation time for a broad range of relevant parameter values.

An important followup question, though, is to what extent load imbalances will affect overall performance. In [7] the analysis was carried out in the ideal case, and it was assumed that load imbalances could be handled efficiently by standard re-partitioning algorithms. Here we aim to quantify the cost of load balance, both to further gauge the feasibility of domain-decomposed MC codes, as well as to provide a decision-making metric for carrying out load balancing in production codes.

Following [8] we first define the *load balance* Γ_i at each stage i as

$$\Gamma_i := \frac{\overline{P}_i}{p_i^{max}}, \tag{2}$$

where

$$\overline{P}_i := \frac{1}{N}\sum_{j=1}^{N} p_{i,j}$$

and p_i^{max} denotes the maximum particle count at stage i. The amount the load balance Γ_i differs from the ideal case $\Gamma = 1$ measures the relative difference between the highest particle and average particle counts:

$$1 - \Gamma_i = \frac{p_i^{max} - \overline{P}_i}{p_i^{max}} \tag{3}$$

Note that unlike [8] we express the load imbalance in terms of particle densities rather than execution time. Their equivalence will be argued in the following section.

We seek to estimate a related but distinct quantity – an upper bound for the difference between the total per-cycle simulation time of the non-ideal case and the ideal case. We define this normalized difference as the *load imbalance penalty* Δ:

$$\Delta := \frac{\tau' - \tau}{\tau} = \frac{\tau'}{\tau} - 1, \tag{4}$$

where τ denotes the total per-cycle simulation time in the ideal case, and τ' for the non-ideal case. Note that an expression for Δ must include both the particle tracking as well as the communication costs across all stages of a given cycle. While related to the per-stage particle load balance, the load imbalance penalty is a distinct quantity.

2.2. Basic properties

Given this simple problem definition above, it is clear that, for purely reflective boundary conditions (a good approximation for reflectors in power reactors) the *global* number of particles at any *stage* is given by

$$P_{i+1} = \sum_{j=1}^{N} p_{i+1,j} = \sum_{j=1}^{N} \lambda_{i,j} p_{i,j} \qquad i = 0, 1, \ldots M, \tag{5}$$

where M denotes the final stage in the cycle – i.e. when all particles are absorbed and none remain. We emphasize again that P_i denotes the global particle count at stage i while $p_{i,j}$ denotes the local particle count on partition j at stage i.

To estimate τ', the per cycle simulation time in the non-ideal case, we seek an expression for the *local* particle distribution – i.e. the number of particles on each partition x_j at stage i. Assuming isotropic leakage, the neutron count on a partition is given by:

$$p_{i+i,j} = \frac{1}{6} \left(\lambda_{i,j1} p_{i,j1} + \lambda_{i,j2} p_{i,j2} + \cdots + \lambda_{i,j6} p_{i,j6} \right) \tag{6}$$

where $j1, j2 \cdots j6$ denote the six immediate neighbors of partition j on a Cartesian lattice. Equation 6 simply states that, in advancing stages, a given partition receives $1/6th$ of the leaked particles from each of its six neighbors on a Cartesian grid.

Equation 6 shows that the evolution of the local particle distributions depends on the detailed alignment of the local particle counts and the local leakage rates. To reduce it further requires additional assumptions. We start by noting that, in an LWR, the neutron spectrum and material heterogeneities are roughly constant from partition to partition (this property is demonstrated using simulation data in Section 3). This observation makes the case of approximately spatially uniform local leakages of great practical interest in the present context, and thus we initially consider the case of $\lambda_{i,j} = \lambda_i$. Note that we still expect non-trivial per-stage variation in leakages as neutron energies will shift toward the thermal range with increasing stages. Important model corrections for small spatial variations in λ will be considered in Section 3.

For spatially constant λ, (5) then becomes

$$\begin{aligned}
P_i &= \sum_{j=1}^{N} \lambda_{i-1} p_{i-1,j} = \lambda_{i-1} \sum_{j=1}^{N} p_{i-1,j} = \lambda_{i-1} P_{i-1} \\
&= \lambda_{i-1}\lambda_{i-2}\ldots\lambda_0 P_0 \\
&\sim \langle\lambda\rangle^i P_0 \qquad\qquad i = 1, 2, \ldots M
\end{aligned} \tag{7}$$

where $\langle\lambda\rangle$ is defined as the geometric mean of the leakage rate,

$$\langle\lambda\rangle := \sqrt[M]{\lambda_{M-1}\lambda_{M-2}\ldots\lambda_0}. \tag{8}$$

Equation 6 then simplifies to

$$p_{i+i,j} = \frac{\lambda_i}{6}\left(p_{i,j1} + p_{i,j2} + \cdots + p_{i,j6}\right) \tag{9}$$

Equation 9 shows that, for spatially constant λ, at each subsequent stage each partition has the average value of the total leaked particles from the neighboring partitions in the previous stage. Equation 7 shows that, in the case of constant λ, the particle load imbalance does not affect the number of stages required to complete a cycle, which can be estimated by setting $P_i = 1$ in (7):

$$\langle\lambda\rangle^M P_0 = 1 \implies M \sim -\frac{\log P_0}{\log\langle\lambda\rangle}. \tag{10}$$

Both properties are used in the analysis that follows.

2.3. Expression for τ

Given these basic properties, we first seek an estimate of the total per-cycle cost in the idealized case of an initial even distribution of particles and spatially constant λ. This is accomplished by decomposing τ into a local work τ_l and inter-processor communication τ_c component as

$$\tau := \tau_l + \tau_c. \tag{11}$$

In the case of perfect load balancing and assuming a roughly equal distribution of track length and neutron spectra across partitions (i.e. our earlier assumption of constant λ), the local work τ_l should be roughly proportional to the total number of particles tracked on a partition in a given cycle,

$$\tau_l = \mu \sum_{i=0}^{M} \overline{P}_i, \tag{12}$$

where the constant μ is a measure of the tracking time per particle, and where it is understood that the particle count $p_{i,j}$ is the same on any given partition, since all partitions are equivalent in the ideal case.

The communication time τ_c can be further decomposed into a latency and bandwidth component [7]. For each cycle, a total of M messages need to be sent to each processor's six neighbors, so in general the total time per cycle due to message latency can be modeled as $\sim 6\alpha M$, where α is some measure of the effective application-level latency for a single send.

If we ignore the dependence of λ on stage, by definition $\lambda\overline{P}_i$ particles are sent from each processor at stage i, the total number of particles sent in a cycle from any processor is $\lambda \sum_i \overline{P}_i$. Thus, the bandwidth term can be roughly modeled as $\beta\lambda \sum_i \overline{P}_i$, where β denotes the effective inverse bandwidth for nearest-neighbor exchanges (expressed in $\frac{time}{particle}$).

When we account for stage dependence λ is replaced by $||\lambda||$, defined as the solution to the M-th order polynomial:

$$\sum_{i=1}^{M} ||\lambda||^i = \lambda_0 + \lambda_1\lambda_0 + \cdots + \lambda_M\lambda_{M-1}...\lambda_0. \tag{13}$$

The latency term can then be written as:

$$\begin{aligned}
\beta \sum_{i=0}^{M} \lambda_i \overline{P}_i &= \beta(\lambda_0\overline{P}_0 + \lambda_1\overline{P}_1 + \ldots \lambda_M\overline{P}_M) \\
&= \beta(\lambda_0\overline{P}_0 + \lambda_1\lambda_0\overline{P}_0 + \cdots + \lambda_M\lambda_{M-1}...\lambda_0\overline{P}_0) \\
&= \beta\overline{P}_0(\lambda_0 + \lambda_1\lambda_0 + \ldots) \\
&= \beta\overline{P}_0 \sum_{i=1}^{M} ||\lambda||^i \\
&\sim \beta\overline{P}_0 \frac{||\lambda||}{1 - ||\lambda||}
\end{aligned} \tag{14}$$

Using these relations together with (10) then yields the following expression for the total communication time:

$$\tau_c = -6\alpha \frac{\log P_0}{\log\langle\lambda\rangle} + \beta \overline{P}_0 \frac{||\lambda||}{1 - ||\lambda||} \tag{15}$$

where it is understood that the last term in the summation is zero – i.e. no particles are sent after the final stage – and that it is included as a notational convenience. Using the same approach on (12) and combining with (15) gives the final expression for total simulation time in the idealized case:

$$\tau = -6\alpha \frac{\log P_0}{\log\langle\lambda\rangle} + (\mu + \beta\|\lambda\|) \frac{\overline{P}_0}{1 - \|\lambda\|}, \tag{16}$$

2.4. Expression for τ'

We now seek an estimate for the total cycle time τ' in the presence of an initial load imbalance. For convenience we first express $p_{i,j}$ as a combination of partition mean and fluctuating parts. That is,

$$\delta p_{i,j} \quad := \quad p_{i,j} - \overline{P}_i \tag{17}$$

When a load imbalance is present the partition with the largest particle count controls the total performance cost. If we denote the particle count on this process as $p_i^{max} = \overline{P}_i + \delta p_i^{max}$, then by analogy with (16) it is clear that the load-imbalanced performance cost is:

$$\begin{aligned} \tau' &= -6\alpha \frac{\log P_0}{\log\langle\lambda\rangle} + (\mu + \beta\|\lambda\|) \sum_{i=0}^{M} (\overline{P}_i + \delta p_i^{max}) \\ &= \tau + (\mu + \beta\|\lambda\|) \sum_{i=0}^{M} \delta p_i^{max} \end{aligned} \tag{18}$$

where the equivalence of the latency terms (i.e. (10)) has been used.

While we cannot directly evaluate the sum of δp_i^{max}, a simple upper bound can be derived by recognizing that, assuming roughly equal leakage to each neighbor, the largest possible value of p at stage $i+1$ occurs if all six neighbors of a partition contain δp_i^{max} particles. Thus,

$$\delta p_{i+1}^{max} \leq \lambda_i \delta p_i^{max}. \tag{19}$$

Substituting the right hand side of (19) for each element of the sum (22) yields:

$$\begin{aligned}\sum_{i=0}^{M} \delta p_i^{max} &\leq \delta p_0^{max} + \lambda_0 \delta p_0^{max} + \cdots + \lambda_M \delta p_0^M \\ &= \delta p_0^{max}(1 + \lambda_0 + \lambda_0\lambda_1 + \dots) \\ &\sim \frac{\delta p_0^{max}}{1 - ||\lambda||} \end{aligned} \tag{20}$$

Equation 18 then becomes

$$\tau' = \tau + (\mu + \beta\|\lambda\|)\frac{\delta p_0^{max}}{1 - ||\lambda||}. \tag{21}$$

2.5. Expression for Δ

Using (18) and (21) we get the following expression for Δ:

$$\Delta := \frac{\tau' - \tau}{\tau} \leq \frac{\delta p_0^{max}}{(1 - \|\lambda\|)\epsilon + \overline{P_0}} \tag{22}$$

where ϵ measures the relative importance of latency relative to bandwidth and tracking timescales, i.e.

$$\epsilon := \frac{6\alpha}{\mu + \beta\|\lambda\|} \frac{\log P_0}{\log\langle\lambda\rangle} \tag{23}$$

which is presumed to be small for typical problem sizes and parameter regimes (e.g. see [7]) but which is retained here for the sake of generality. Note that (22) implies that

$$\Delta \leq \frac{\delta p_0^{max}}{\overline{P_0}} = \frac{1}{\Gamma_0} - 1, \tag{24}$$

which should be a good approximation for applications where the latency term is much smaller than the bandwidth and tracking terms.

Assuming spatially constant λ, then, and given isotropic neutron local leakage and constant mean tracking rates per partition, we can establish an upper bound for Δ entirely in terms of the initial particle configuration.

3. Variable leakage rates

Equation 22 should give a reasonable estimate for reactor applications across a range of parameter values, where material inhomogeneities are roughly equally distributed and thus local leakage rates show very little variation. However, it fails to capture a critical effect that emerges as we move to smaller partition sizes, and which sets an important limit on the utility of the domain-decomposed approach. To see this we explicitly account for spatially variant, non-constant leakage in the formulation of the model.

Consider a distribution of leakage rates across partitions at a given stage with a maximum value defined as:

$$\lambda_i^{max} = \max \{\lambda_{i,j} : 1 \le j \le N\} \tag{25}$$

In estimating the computation time for this scenario compared to the ideal case (i.e. calculating Δ), the question arises of what corresponding spatially constant value of λ should be used for the ideal case. Several options are reasonable, but here we choose a mean value that preserves stages. Specifically, if we define a particle-weighted mean leakage as:

$$\overline{\lambda}_i := \frac{\sum_{j=1}^{N} p_{i,j}\lambda_{i,j}}{P_i} \tag{26}$$

then the ideal and non-ideal cases are guaranteed to have the same number of global particles at successive stages:

$$\frac{P_{i+1}}{P_i} = \frac{1}{P_i}\sum_{j=1}^{N} \lambda_{i,j}p_{i,j} = \overline{\lambda}_i$$

Following (19), then, the largest possible particle count at stage i occurs on partition x_j if on its six neighbors p_i^{max} coincides with λ_i^{max}. That is,

$$p_{i+1}^{max} \le \lambda_i^{max} p_i^{max}. \tag{27}$$

This relation then allows us to derive an upper bound for τ' in the case of variable λ:

$$\begin{aligned} \tau' &\le 6\alpha\frac{\log P_0}{\log\langle\overline{\lambda}\rangle} + \beta\sum_{i=0}^{M} \lambda_i^{max}p_i^{max} + \mu\sum_{i=1}^{M} p_i^{max} \\ &= 6\alpha\frac{\log P_0}{\log\langle\overline{\lambda}\rangle} + (\mu + \beta||\lambda^{max}||)\sum_{i=0}^{M} p_i^{max} \end{aligned} \tag{28}$$

where $||\lambda^{max}||$ is defined analogous to (13):

$$\sum_{i=1}^{M} ||\lambda^{max}||^i = \lambda_0^{max} + \lambda_1^{max}\lambda_0^{max} + \cdots + \lambda_M^{max}\lambda_{M-1}^{max}...\lambda_0^{max}.$$

This then yields the following upper bound for the load imbalance:

$$\Delta \leq \frac{(\mu + \beta||\lambda^{max}||)\sum_{i=0}^{M}\delta p_i^{max}}{6\alpha\frac{\log P_0}{\log\langle\overline{\lambda}\rangle} + (\mu + \beta||\overline{\lambda}||)\frac{\overline{P_0}}{1-||\overline{\lambda}||}} \tag{29}$$

Note that the term $\sum_{i=0}^{M}\delta p_i^{max}$ can be written in terms of p_0^{max} as:

$$\begin{aligned}\sum_{i=0}^{M}\delta p_i^{max} &= \delta p_0^{max} + \lambda_0^{max}\delta p_0^{max} + \lambda_0^{max}\lambda_1^{max}\delta p_0^{max} + \ldots \\ &= \delta p_0^{max}(1 + \lambda_0^{max}|\lambda_0^{max}\lambda_1^{max} + \ldots \\ &= \delta p_0^{max}\sum_{i=1}^{M}||\lambda^{max}||^i \end{aligned} \tag{30}$$

If for convenience we replace $\sum_{i=1}^{M}||\lambda^{max}||^i$ by its continuous representation $\frac{1}{1-||\lambda||}$, then (29) implies that:

$$\Delta \leq C\frac{p_0^{max}}{\overline{P}_0} = \frac{C}{\Gamma_0} - 1. \tag{31}$$

where the quantity C, defined as,

$$C := \frac{(\beta||\lambda^{max}|| + \mu)}{(\beta||\overline{\lambda}|| + \mu)}\frac{(1 - ||\overline{\lambda}||)}{(1 - ||\lambda^{max}||)},$$

can be seen as a "correction factor" relative to (24) – clearly, when $||\overline{\lambda}|| = ||\lambda^{max}||$, (29) reduces to (24).

Note however that the Taylor Series expansion used in the derivation of (31),

$$\frac{1}{1-x} = 1 + x + x^2 + \cdots + x^M,$$

even for large M is a poor approximation for x extremely close to 1. While this will not be a problem for average values of λ in reactor applications, we must allow for the possibility that $\sum_{i=0}^{M}||\lambda^{max}||^i$ approaches its true upper bound of M (i.e. the *maximum* leakage in each stage is 1). Thus, a potentially more accurate version of C is:

$$C := \frac{(\beta||\lambda^{max}|| + \mu)}{(\beta||\overline{\lambda}|| + \mu)} \frac{\sum_{i=0}^{M} ||\lambda^{max}||^i}{\sum_{i=0}^{M} ||\overline{\lambda}||^i}$$

The greater the spatial variation in leakage rates the more the system bandwidth and neutron tracking rates factor into the performance. The implications of these formulas are explained in the simple tests below, where we aim to estimate C as a function of the main problem parameters.

4. Evaluation of model

For a given initial particle configuration, evaluation of the model equation (29) requires estimates for particle tracking rate μ, application-level inverse bandwidth β and latency α, and local leakage rate estimates from which to compute $||\lambda^{max}||$ and $||\overline{\lambda}||$. These parameters vary widely by both machine and specific code application. Here we evaluate these terms in a parameter regime relevant to LWR physics on modern supercomputers. Other applications and machine architectures can be evaluated with appropriate values of these parameters.

4.1. Leakage rates

While best-estimate application-level bandwidth and latency terms can be estimated from standard benchmarks (e.g. [9]), the leakage rate terms in (29) are more difficult to approximate. Simplified models, such as the Wigner rational approximation [10] provide rough estimates of expected domain-dependent leakage rates, but do a poor job at estimating stage dependence. Thus, to be as precise as possible we measure these values directly using an existing Monte Carlo simulation code – OpenMC [11]. Note that, though parallel, OpenMC does not employ domain decomposition. To model partitions and stages, we overlay within OpenMC an imaginary grid decomposition and effectively measure the behavior of particles between fictional stages during a cycle.

The specific test executed was based on the Monte Carlo Performance benchmark [12] using .5 billion active particle histories. Leakage rates at each stage and within each partition were measured for three cases: a single assembly, quarter-assembly, and ninth-assembly partition overlay using 20, 40, and 60 axial levels, respectively. In each case, the benchmark model was simulated with 1 million neutrons per cycle for 150 inactive and 500 active cycles. For neutron cross-sections, data from ENDF/B-VII.0 was used.

Figure 1 illustrates the scale and level of variation in local leakage rates for the full, quarter, and one-ninth assembly cases. The plots shown are for axial level nearest the middle of the core at stage zero. They represent "typical" leakage rate distributions and are intended to graphically illustrate their relative lack of spatial coherency and small range of values. Also, it is evident from the figure that, as expected, leakage rates increase non-trivially with decreasing partition size.

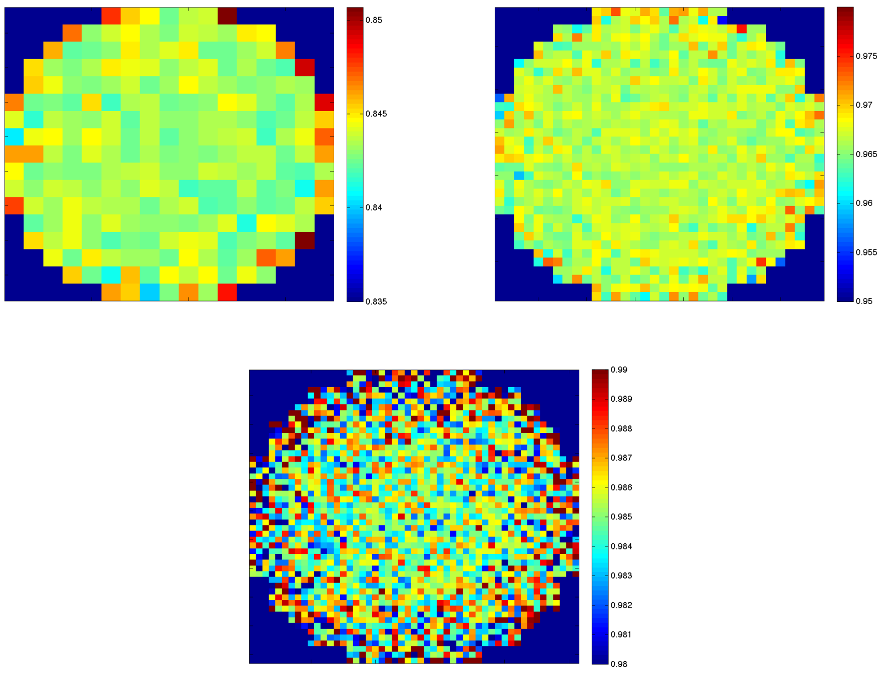

Figure 1: Sample leakage rate distributions for the full, quarter, and ninth assembly experiments. Each image shows the local leakage for stage 0 at the midplane. The figure depicts a "typical" case, showing the very small degree of spatial fluctuation and lack of spatial coherency. Also, the expected trend toward higher leakage rates with decreasing partition size is evident.

To see this more clearly, Figure 2 shows the stage-dependent mean and standard deviation (error bar overlay) of λ for each simulation. In each case, we see the clear trend toward lower leakages as neutrons have a higher probability of thermalization in later stages. Furthermore, the superposed standard deviations indicate extremely small spatial variation in the first several stages, which accounts for the majority of data movement and performance cost. When particle counts are small in later stages statistical variations result in larger standard deviation values, but their impact on total performance is expected to be small. We note that this very small spatial variation hints that the correction factor for variable λ in equation (31) may be small. This is evaluated in the next section.

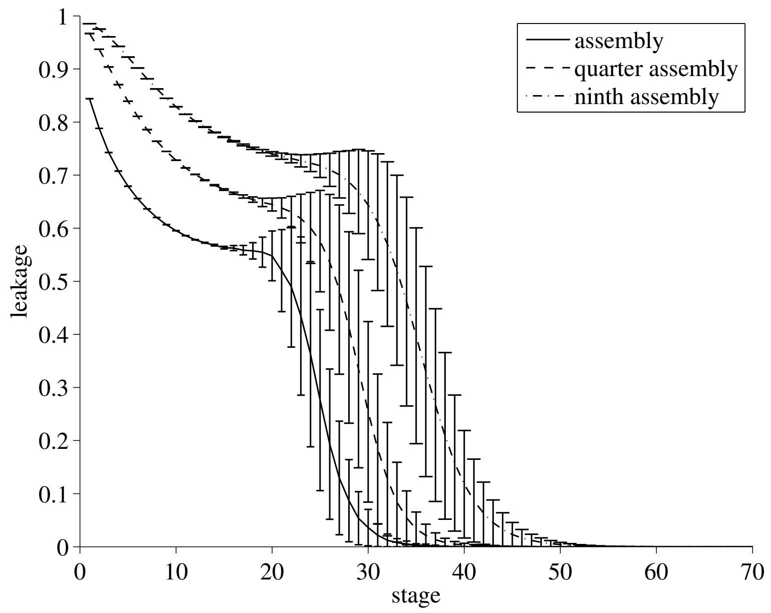

Figure 2: *Average values of leakage rate λ at each stage for the full, quarter, and ninth assembly experiments. Superposed as error bars are the standard deviation, showing very little spatial variation in the early stages. The leakage rate trend clearly indicates the higher probability of thermalization in later stages of the calculation*

We next test the predictions for total number of cycles M per stage given by (10), i.e.

$$M = -\frac{\log P_0}{\log \left\langle \overline{\lambda} \right\rangle}$$

The values of the $\left\langle \overline{\lambda} \right\rangle$ were calculated for each of the three experiments and plugged into the formula for M. Table 1 compares these results with

experiment	M (data)	M (model)
full assembly	41	39
quarter assembly	52	51
ninth assembly	71	70

Table 1: *The number of stages M for the three numerical experiments vs. the value predicted by (10)*

those obtained directly from the simulation. The model formula behaves as expected, differing by only several percent from the measured data. Exact correspondence is not expected – statistical fluctuations, slight anisotropies and other minor effects are likely to yield small variations. For practical purposes though the current estimate is more than adequate.

It is furthermore instructive to test the fidelity of (27) to the true measure maximum particle counts at each stage. While (27) is a true statement, in a practical sense it is of questionable value if it over-predicts p_i^{max} by too significant a margin. Figure 3 shows the computed value of p^{max} for each stage versus the value predicted by (27). Given that leakage rate variation is very small spatially, it is not surprising to see that (27) works extremely well as an upper bound, over-predicting the measured value by less than 1.0% for the initial stages (which account for the bulk of the particle transfers).

4.2. Evaluation of correction factor C and penalty Δ

Given an initial particle configuration and reasonable estimates for leakage, it remains to evaluate C in (31). To reiterate, under the assumption of spatially constant leakage C is identically 1 and the load balance can be upper bounded by the initial particle configuration as $\frac{p_0^{max}}{P_0}$. When leakage rates vary spatially C measures the amplification of the performance penalty. We estimate C in two steps, first evaluating the contribution of the bandwidth and tracking times, given by the ratio

$$\frac{\beta||\lambda^{max}|| + \mu}{\beta||\bar{\lambda}|| + \mu} = \frac{\frac{\beta}{\mu}||\lambda^{max}|| + 1}{\frac{\beta}{\mu}||\bar{\lambda}|| + 1}.$$

Note that for any conventional machine $\beta << \mu$, reflecting that tracking rates are much slower than inter-processor communication, and since $||\bar{\lambda}|| \sim ||\lambda^{max}||$, this term remains very close to unity and contributes negligibly to the overall load imbalance.

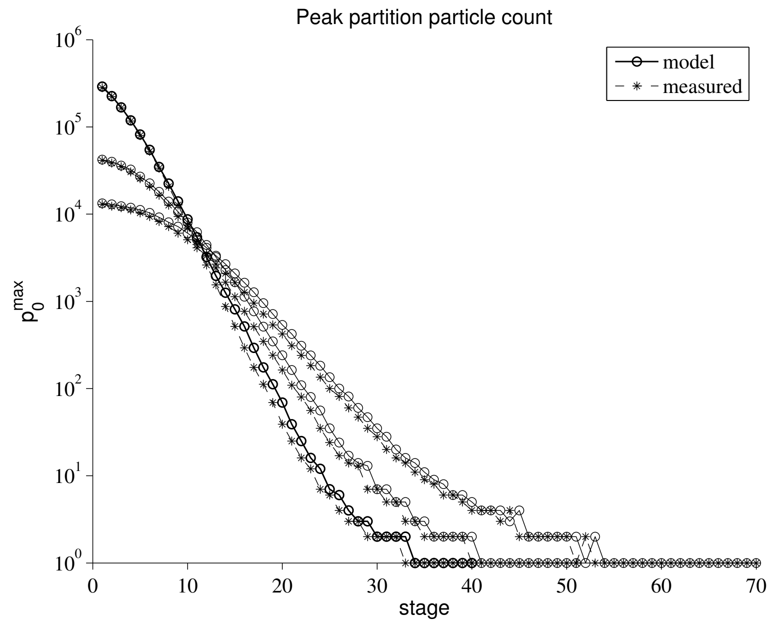

Figure 3: *Average values of leakage rate λ at each stage for the full, quarter, and ninth assembly experiments. Superposed as error bars are the standard deviation, showing very little spatial variation in the early stages. The leakage rate trend clearly indicates the higher probability of thermalization in later stages of the calculation*

The second contribution of C is the ratio

$$\frac{\sum_{i=0}^{M} ||\lambda^{max}||^i}{\sum_{i=0}^{M} ||\bar{\lambda}||^i}.$$

Note that this term becomes problematic as the processor grid is refined, since we intuitively expect maximum leakage rates of unity for sufficiently small domains. Assuming that this is the case, the numerator is upper bounded by $1 + M$, where again M is the number of stages in a cycle. Assuming that the average value does not approach unity, the Taylor series approximation should be reasonable and we can rewrite this expression as:

$$\frac{\sum_{i=0}^{M} ||\lambda^{max}||^i}{\sum_{i=0}^{M} ||\bar{\lambda}||^i} \leq M(1 - ||\bar{\lambda}||).$$

While this term is likely a modest fraction of the total number of stages,

experiment	$\frac{\beta\|\|\lambda^{max}\|\|+\mu}{\beta\|\|\bar{\lambda}\|\|+\mu}$	$\frac{\sum_{i=0}^{M}\|\|\lambda^{max}\|\|^i}{\sum_{i=0}^{M}\|\|\bar{\lambda}\|\|^i}$	C	$\frac{p_0^{max}}{P_0}$	Δ
full assembly	1.00	1.13	1.13	3.24	3.67
quarter assembly	1.00	2.58	2.58	3.28	8.47
ninth assembly	1.00	6.98	6.98	3.45	24.15

Table 2: Values of Δ and the various terms which contribute to it for each of the three numerical experiments. The tables used values of inverse latency $\beta = 10^{-8}\frac{sec}{particle}$ and tracking time $\mu = 5 \times 10^{-4}\frac{sec}{particle}$. Note that within the precision presented the bandwidth term (second column) is identical in all cases, a manifestation of the fact that bandwidths are much higher than tracking rates. Notice also that the load imbalance penalty is magnified significantly on the finest partitions grid.

we must recall that M increases with decreasing partition size, and even a small fraction could easily significantly amplify the load imbalance.

To explore this in greater depth requires use of the OpenMC simulation results. Table 2 shows the results, including the model predictions for the load imbalance penalty for the full, quarter, and ninth assembly experiments. Note that in all cases the initial particle configuration and thus $\frac{p_0^{max}}{p_0}$ is expected to be roughly independent of partition size and be roughly approximated by the one-group solution on a cylindrical geometry:

$$S(x,y,z) = J_0\left(\frac{2r_0x}{L}\right) J_0\left(\frac{2r_0y}{L}\right) \cos\left(\frac{\pi z}{L}\right), \quad \frac{L}{2} \geq x,y,z \leq \frac{L}{2}$$

where $r_0 = \pm 2.4048$ is the root of the zeroth order Bessel Function J_0. Note that S has a peak to mean value $\frac{S^{max}}{\bar{S}} = 4.33$, which is reasonably close the values of $\frac{p_0^{max}}{\bar{P}}$ shown in Table 2.

Equation 31 states that, with no load rebalancing, a simulation with this initial particle distribution is expected to take at most $C\frac{p_0^{max}}{P_0}$ as long as a perfectly load balanced simulation. For the full assembly simulation $C = 1.13$ and the total penalty is 3.67, which could in many contexts be considered reasonable compared to e.g. the cost and complexity of implementing repartitioning algorithms. However, it is clear that the situation rapidly deteriorates for decreasing partition size, with a value of $C = 6.98$ for the ninth partition experiments. This corresponds to load imbalance penalty of 24.15, which for most contexts is likely unacceptably high. It is clear what has happened both in the model and the physics – In the one-ninth assembly case the peak leakage is unity for all but the final stage, and thus the summation in the numerator of C approaches M. Note that we expect the performance to degenerate even further with decreasing partition size since M is expected to increase according to (10).

5. Conclusion

We have developed simple relationships to quantitatively analyze the impact of load imbalances on the performance of domain decomposed MC methods in the context of reactor analysis. These techniques provide a quantitative framework to estimate the additional performance costs incurred by typical load imbalances in reactor applications. Preliminary numbers were presented for a classic reactor benchmark, indicating that load imbalances were not that significant for assembly-size partitions, but increased dramatically as partition sizes were decreased beyond that point. This indicates that domain decomposition is likely a reasonable strategy for modest-size parallelism but that it is inherently limited when we consider the massive levels of concurrency on the path to exascale computing (at least without significant repartitioning).

Our main goal however is not to judge here whether these penalties are large or small in an absolute sense. Rather, the techniques presented allow one to weigh tradeoffs between domain decomposition and more sophisticated data decomposition strategies for their specific needs, or perhaps to estimate the cost of carrying out load re-balancing or other re-tracking techniques within an operational production code. When processing power is cheap and memory is at a premium, factors of several in performance time are not necessarily large, and performance models that go beyond purely speculative are a critical component of assessing the best path forward.

References

[1] Jonhhwa Chang and Nam Zin Cho. Some outstanding problems in neutron transport computation. *J. Korean Nuc. Soc.*, 41(4):381–390, 2009.

[2] J. Eduard Hoogenboom and William R. Martin. A proposal for a benchmark to monitor the performance of detailed Monte Carlo calculation of power densities in a full size reactor core. In *Proc. 2009 Int. Conf. on Math., Comp. Meth., & Reactor Phys.*, Saratoga Springs, NY, 2009. American Nuclear Society.

[3] J.C. Wagner, D.E. Peplow, S.W. Mosher, and T.M. Evans. Review of hybrid (deterministic/Monte Carlo) radiation transport methods, codes, and applications at Oak Ridge National Laboratory. In *Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo*, Tokyo, 2010.

[4] Daniel J. Kelly, Thomas M. Sutton, Timothy Trumbull, and Peter S. Dobreff. MC21 Monte Carlo analysis of the Hoogenboom–Martin full-core PWR benchmark problem. In *PHYSOR – Advances in Reactor Physics to Power the Nuclear Renaissance*, 2010.

[5] Paul K. Romano, Benoit Forget, and Forrest Brown. Towards scalable parallelism in monte carlo particle transport codes using remote memory access. *Progress in NUCLEAR SCIENCE and TECHNOLOGY*, 2:670–675, 2011.

[6] F. Brown. Recent advances and future prospects for monte carlo. In *Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo (SNA + MC2010*, Tokyo, 2010.

[7] A. Siegel, K. Smith, V. Mahadevan, and P. Fischer. Analysis of communication costs for domain decomposed monte carlo methods in nuclear reactor anlaysis. *J. Comput. Phys.*, 231:3119–3125, 2012.

[8] L. Ridgway Scott, Terry Clark, and Babak Bagheri. Scientific Parallel Computing. Princeton University Press, Princeton, NJ, 2005.

[9] Alan J. Wallcraft. SPMD OpenMP versus MPI for ocean models. *Concurrency - Practice and Experience*, 12(12):1155–1164, 2000.

[10] Yu. G. Pashkin. Accuracy of the Wigner approximation. *Atomic Energy*, 28:184–185, 1970. 10.1007/BF01162626.

[11] Paul K. Romano and Benoit Forget. The OpenMC Monte Carlo particle transport code. *Ann. Nucl. Energy*, 2012. Submitted.

[12] J. Eduard Hoogenboom, William R. Martin, and Bojan Petrovic. The monte carlo performance benchmark test - aims, specifications and first results. In *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering*, Rio de Janeiro, Brazil, May 2011.