Performance and Accuracy of Criticality Calculations Performed Using WARP - A Framework for Continuous Energy Monte Carlo Neutron Transport in General 3D Geometries on GPUs

Ryan M. Bergmann*, Kelly L. Rowland, Nikola Radnović, Rachel N. Slaybaugh, Jasmina L. Vujić

Department of Nuclear Engineering, 4155 Etcheverry Hall, University of California - Berkeley, Berkeley, CA 94720-1730

Abstract

In this companion paper to "Algorithmic Choices in WARP - A Framework for Continuous Energy Monte Carlo Neutron Transport in General 3D Geometries on GPUs" (doi:10.1016/j.anucene.2014.10.039), the WARP Monte Carlo neutron transport framework for graphics processing units (GPUs) is benchmarked against production-level central processing unit (CPU) Monte Carlo neutron transport codes for both performance and accuracy. Neutron flux spectra, multiplication factors, runtimes, speedup factors, and costs of various GPU and CPU platforms running either WARP, Serpent 2.1.24, or MCNP 6.1 are compared. WARP compares well with the results of the production-level codes, and it is shown that on the newest hardware considered, GPU platforms running WARP are between 0.8 to 7.6 times as fast as CPU platforms running production codes. The GPU platforms running WARP are also shown to be between 15% and 50% as expensive to purchase and between 80% to 90% as expensive to operate as equivalent CPU platforms performing at an equal simulation rate.

Keywords: Monte Carlo, Neutron Transport, GPU, CUDA, OptiX, Supercomputing

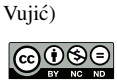
1. Introduction

WARP Bergmann and Vujić (2015), which can stand for "Weaving All the Random Particles," is a three-dimensional (3D) continuous energy Monte Carlo neutron transport code developed at UC Berkeley to efficiently execute on NVIDIA graphics processing unit (GPU) platforms. WARP accelerates Monte Carlo simulations while preserving the benefits of using the Monte Carlo method, namely, that very few physical and geometrical simplifications are applied. WARP is able to calculate multiplication factors, neutron flux distributions (in both space and energy), and fission source distributions for time-independent neutron transport problems. It can run in both criticality or fixed source modes, but fixed source mode is currently not robust or optimized. WARP can transport neutrons in unrestricted arrangements of parallelepipeds, hexagonal prisms, cylinders, and spheres.

The goal of developing WARP is to investigate algorithms that can grow into a full-featured, continuous energy, Monte Carlo neutron transport code that is *accelerated* by running on GPUs. The crux of the effort is to make Monte Carlo calculations faster while producing accurate results. Modern supercomputers are commonly being built with GPU coprocessor cards in their nodes to increase their computational efficiency and performance. Compared to more common central processing units (CPUs), GPUs have a larger aggregate memory bandwidth, much higher rate of floating-point operations per second (FLOPS), and lower energy consumption per FLOP. GPUs execute efficiently on data-parallel problems, but most CPU codes, including those for Monte Carlo neutral particle transport, are predominantly task-parallel. Data-parallelism is parallelism that arises from a single task operating on many different pieces of data at one time, whereas task-parallelism is parallelism that arises from running many concurrent tasks that each

*Corresponding author. Tel.: +41.76.687.53.09.

Email addresses: ryanmbergmann@gmail.com (Ryan M. Bergmann), krowland@berkeley.edu (Kelly L. Rowland), radnovicn@gmail.com (Nikola Radnović), slaybaugh@berkeley.edu (Rachel N. Slaybaugh), vujic@nuc.berkeley.edu (Jasmina L. Vujić)



FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

act on a single piece of data. Figure 1 shows an illustration of the difference between a data-parallel and a task-parallel neutron transport loop. Therefore, to create Monte Carlo software that works well on GPUs, the transport algorithms had to be reconsidered.

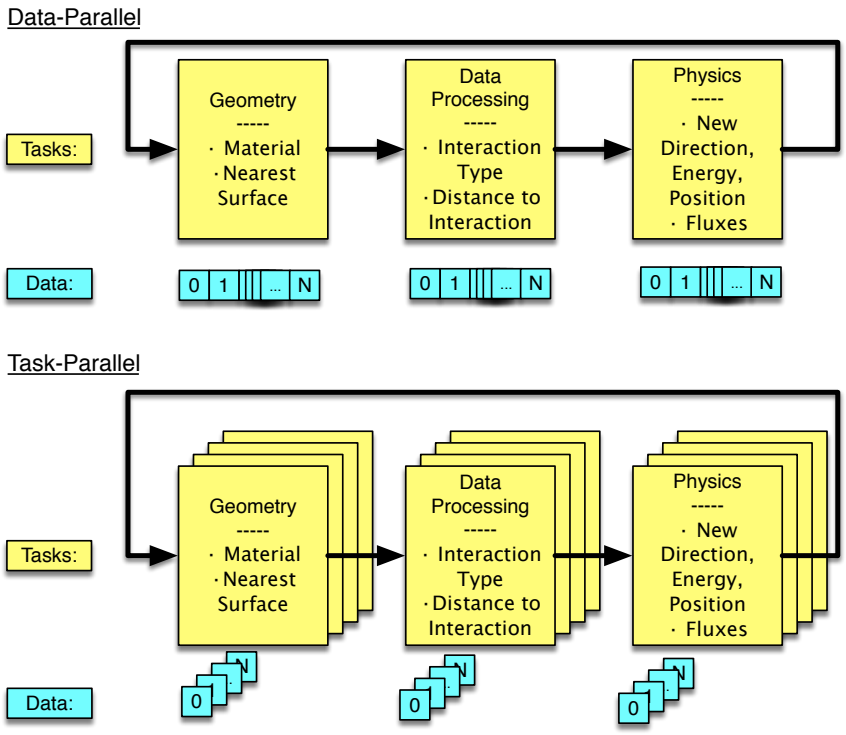


Figure 1: A data-parallel neutron transport loop vs. a task-parallel transport loop for transporting N neutrons in parallel.

In this paper, we start in Section 2 discussing background information about GPUs and CPUs in the context of Monte Carlo neutral particle transport. Then, in Section 3, the current features of WARP are summarized while highlighting the newest developments. The results of criticality calculations performed by WARP are compared against those from Serpent 2.1.24 Leppänen (2007); ser (2014) and MCNP 6.1 Pelowitz et al. (2013), two widely-used production-level Monte Carlo neutron transport codes. This comparison is done to ensure the accuracy of WARP and to highlight performance differences. The test case models used to conduct the comparisons are outlined in detail in Section 4, then the run times, multiplication factors, and neutron flux spectra calculated for the test cases are compared in detail in Section 5, after which a discussion is given summarizing comparisons and addressing any discrepancies shown in Section 6. Finally, conclusions about the initial development of WARP are drawn and future work is roughly outlined in Section 7.

2. Background

Typical Monte Carlo neutron transport algorithms are "task-based." A thread will transport the neutron until it is terminated through absorption or by leaking out of the system. The sequence of events the neutron undergoes from start to finish is called a neutron "history," and is the basic unit of work for a thread in a task-based algorithm. Neutron histories are independent of each other and the algorithm is parallelized by running many histories in independent, parallel threads. Thus, having as many parallel threads as possible should provide the greatest performance. This parallelization method requires that the threads can execute completely independently of each other, i.e. the actions of one thread do not affect the other parallel threads in any way. This is a MIMD (multiple instruction multiple data) way of executing threads. In MIMD execution model, different threads can execute different instructions on their data

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

at a point in time. Threads appear to completely independent, and having threads at different points in the transport algorithm is not a problem since they are allowed to execute their own instructions.

At first glance, it seems that GPU cards would be good for running Monte Carlo neutron transport because they can run large numbers of concurrent threads to complete the large number of independent tasks required in traditional Monte Carlo neutron transport. However, it soon becomes apprent that the situation is complicated by how instructions are carried out on these threads. The reported concurrent thread number is based on the width of the processor's SIMD (Single Instruction Multiple Data) units. SIMD is an execution model some processors use to lower the number of instructions needed per amount of computation done, which increases both power and computational efficiency Lee et al. (2011). SIMD requires the same instructions to be carried out over every element in a concurrently-processed data vector.

An NVIDIA GPU's basic processing unit is a streaming multiprocessor (SM). These multiprocessors house many individual computational cores and can process jobs independently of each other. One instruction is carried out over the entire SM, however, so it is the processing level above which SIMD requirements are no longer present. The GPU programming model abstracts SIMD execution by using threads, which can be thought of in the traditional sense. Single-instruction multiple-thread (SIMT) and SIMD are similar in the sense that identical instructions are carried out across different pieces of data, but SIMT allows threads to act independently, albeit with a performance penalty. From a thread standpoint, SIMT requires threads to execute the same instruction at one point in time. If a set of threads does not execute the same instruction at the same time (the data they act on can be different), the GPU will serialize the operations. In this case, the subset of *divergent* threads executing the first instruction will all execute together, then the subset of divergent threads executing the second instruction will all execute together after the first subset has completed. In the limit of each thread executing a different instruction, the operations become completely serial. The magnitude of the performance penalty depends on the rate at which data is being delivered to the multiprocessor since work can only be done if data is present. If the multiprocessor can complete all jobs, even if it needs some serialization, before the next set of data arrives, then the penalty will be invisible NVI (2013). To most efficiently use SIMT and avoid serialization, instruction regularity[1] is required. Unfortunately, Monte Carlo typically breaks instruction regularity because of its conditional statements based on random numbers. Therefore, if Monte Carlo algorithms are to be used on GPUs, they must be implemented in a manner that carefully takes into account the limitations of the GPU.

A further consideration is that the memory subsystems of GPUs also function in a SIMD-like way. In order to use the full memory bandwidth of the device, more than one piece of data must be loaded and used per transaction, and the only way that multiple pieces of data can be loaded simultaneously is if they are adjacent in memory. In other words, if a program requests a single piece of data at location i and a piece at location $i + 10$, these requests will be split into two separate transactions that each yield one data element. Two data elements in two transactions produces an effective bandwidth of one element per transaction time. On the other hand, if the program requests data at i, $i + 1$, $i + 2$, and $i + 3$, the entirety of the requested data can be retrieved in a single transaction. Four data elements in a single transaction yields an effective memory bandwidth of four elements per transaction time, four times higher than the previous scenario. Having a memory subsystem that handles requests in this way is not ideal for Monte Carlo methods, however. With Monte Carlo, data is accessed in a very random way because of the random nature of the simulation, and requested data is unlikely to be adjacent. This means the full memory bandwidth of the GPU will not be used unless this problem is mitigated in some way.

Another challenge of using GPUs for Monte Carlo is GPUs have very high global memory latency compared to a CPU. Memory latency is the number of clock cycles, or amount of time, it takes for a data request to be fulfilled. As Table 1 shows, the GPU's global memory latency is about an order of magnitude higher than the CPU's Ruggiero (2008); NVI (2013). GPUs try to eliminate the effect of large global latency by pipelining memory access. Pipelining means threads that have received their data can execute as other threads are waiting for their data to load. If many requests are known, the data can be continually loaded as threads start to execute their jobs. The hope is that the jobs take longer than the memory loads, eventually all data arrives, and the later threads appear to have zero latency for their memory access. This is why it is important for GPUs to have such a large number of concurrent threads. It allows them to pipeline data access and minimize the impact of memory latency. A CPU system is a like a fast, strong

[1] "Instruction regularity" means some systematic behavior for the instructions that adjacent threads need to execute. Ideally, they should all be executing the same instructions, but operating on different pieces of data.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

horse that can move some goods from A to B very quickly by itself. If only one cart of goods is allowed on the path from A to B, then a GPU system is like a large team of slower ponies that can move a larger load of goods in a larger time, but only if they work together in moving the same, large load. Averaged over time, the team of ponies can move more goods than the horse if there are enough of them working together.

Table 1: A comparison of an NVIDIA GPU and an Intel CPU Catanzaro (2011); Ruggiero (2008); opt (2011).

Processor	Intel i7 (Westmere-EP)	NVIDIA Tesla C2075 (Fermi)
Processing Elements	6 cores, 2 issue, 4-way SIMD	14 cores, 2 issue, 16-way SIMD
Frequency	3.46 GHz	1.15 GHz
Resident Strands / Threads (max)	48	21,504
Single Precision GFLOPS	166	1030
Mem. Bandwidth	32 GB/s	144 GB/s
Global Latency	~50 clocks	200-800 clocks
FLOPS / byte	5.2	7.2
Register File	6 kB	2 MB
Local Storage / L1 Cache	192 kB	896 kB
L2 Cache	1536 kB	0.75 MB
L3 Cache	12 MB	-

Another notable feature is that the GPU has a greater FLOPS/byte of memory bandwidth ratio than the CPU. This implies that GPUs could be used to turn a compute-bound problem into a bandwidth-bound problem. This may seem like a deficit, but the GPU has a higher maximum memory bandwidth, so even though a problem is bandwidth-bound on a GPU, it may still require less execution time than on a CPU.

The impetus for developing WARP was the research done by Martin and Brown in 1984 Brown and Martin (1984) for Monte Carlo and by Vujić and Martin in 1991 Vujic and Martin (1991) for the collision probability method. In the 1984 paper, a method for mapping the Monte Carlo problem onto SIMD vector computers is described. The essential idea in the 1984 paper was to bank the neutrons into vectors based on their required operation. If a neutron was sampled to scatter, its data were placed into a buffer containing the data of other neutrons that have also been sampled to undergo scattering interactions. If a neutron crosses a surface, it was placed into the surface crossing buffer, and so on. Once a buffer became full, it was processed in a SIMD fashion by the vector computer. Processing the neutron data further after the vector operation caused the buffers to contain non-uniform reactions, however, and a "shuffle" operation was done that actually moved data back into contiguous blocks based on the reaction type.

This new approach was named "event-based" Monte Carlo, since the neutron events are tracked and processed as a group. This was a very different way of performing a Monte Carlo simulation at the time. Almost all computers were strictly serial, and SIMD lanes were only available in supercomputers. Therefore, the pervasive method was the "task-based" method in which neutrons are tracked for their entire lifetime in series. Since GPUs are massively parallel and rely on SIMD, an event-based algorithm seemed to be the appropriate approach to GPU-accelerated neutron transport.

Instead of trying to port an existing code to the GPU, the simplest way to accommodate requirements for efficient GPU execution was to write a new code from scratch. This project ultimately resulted in WARP. WARP uses an event-based algorithm, but with some important differences. Vectorizing the Monte Carlo algorithm should allow for efficient GPU execution, but a paper by Zhang Zhang et al. (2011) also shows that by remapping data references, the thread divergence in GPU warps (groups of threads) can be minimized. Moving data is expensive, especially when the data set is large, so WARP uses a remapping vector of pointer/index pairs to direct GPU threads to the data they need to access. The remapping vector is sorted by reaction type after every transport iteration using a high-efficiency parallel radix sort (via the CUDPP library Harris et al. (2007)), which serves to keep the reaction types as contiguous as possible and removes completed histories from the transport cycle. Sorting reduces the amount of divergence in GPU "thread blocks," keeps the SIMD units as full as possible, and eliminates using memory bandwidth to check if a neutron in the batch has been terminated or not. Using a remapping vector means the data access pattern is irregular, but this is mitigated by using large batch sizes where the GPU can effectively reduce the high cost of irregular global

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

memory access. The details about the existing algorithms used in WARP are discussed in Bergmann and Vujić (2015) and will not be discussed in detail here. New algorithmic changes added since the last publication are discussed in Section 3.

To use NVIDIA GPUs, parts of software must be written in a language that facilitates interaction with the GPU, and CUDA NVI (2013), a set of extensions for C/C++, was chosen. CUDA was first released in 2006 as NVIDIA's proprietary GPU programming platform. It makes minimal additions to C/C++, and any C programmer would be very comfortable programming in CUDA NVI (2013). It was chosen over OpenCL, the open source GPU programming platform, because of CUDA's greater feature support, stability, ease of programming, wider community usage, and ability to use new, cutting-edge features in NVIDIA GPUs that OpenCL does not have.

3. Features of WARP

WARP has only existed since 2013, and it is not as fully-featured as more mature Monte Carlo codes. It has the functionality necessary to compare its performance against Serpent 2.1.24 and MCNP 6.1, however. This section outlines the current set of features available in WARP. WARP is mainly written in C/C++ and is compiled to a shared library. With the shared library compiled, a user must write a `main()` function that calls the library routines and directs program flow. In other words, there currently are no input processing routines included with WARP to interpret text file inputs. Instead, the simulations are configured by providing arguments to the library routines themselves. Therefore, when one wants to change the simulation, the arguments to the WARP routines must be changed, and this requires the `main()` function to be recompiled. This is why WARP is termed a "framework" rather than a "program." Of course, there is flexibility in the setup as well, and a user could write a main function that could handle all conceivable input cases and would never have to recompile the `main()` function.

Pythonic wrapping of the shared library has been done via SWIG Pyt (2016); Beazley (2003), which automatically wraps compiled languages like C/C++ in high-level scripting languages. With the C++ classes exposed in Python, the `main()` function can be replaced with a Python script, eliminating the need to recompile a main function and link it to the WARP library when different geometries or different run parameters are desired. Using a Python script to call the WARP library is effectively equivalent to having a text input file, but also allows users to use programming structures like loops and regular expressions directly in their "input files." Eliminating the text input file removes a level of developer programming effort for input parsing, and allows direct access to the results' data arrays for easy post processing and plotting. Also, eliminating files as the sole program output removes user error and effort in parsing them for post processing and plotting.

The Python wrapping approach deviates from the standard flat text input file structure that many Monte Carlo codes use. Flat text input/output relies on keywords and/or regular formatting, which adds a layer between simulation and analysis. Using Python to directly access the classes and their data removes this layer and allows a user to build complex, custom applications as desired. With Python, the results of a calculation are also resident in a Python session and are therefore easily available to the user for plotting with scripts or processing with other analysis tools. To process data in the same way from a text file output, the output would need to be parsed with a user-written function or processed by hand, which is time-consuming and can lead to human error.

3.1. Physics

Only neutrons are transported by WARP. Any other particles are not considered in any way. Cross section data compiled by the United States is distributed by the Department of Energy in *ENDF* files. ENDF stands for "Evaluated Nuclear Data File" and can contain data for nuclear decay, photons, atomic relaxation, fission yields, thermal neutron scattering, and charged particle reactions as well as neutron reactions Chadwick et al. (2011); Laboratory (1998). WARP loads ACE-formatted nuclear data libraries via the "ace" module in PyNE (Python for Nuclear Engineering) Scopatz et al. (2014). This module has been separated from PyNE and included as a standalone Python module with WARP. Including it as a separate module with WARP means the entire PyNE package does not need to be installed in order to compile and run WARP. ACE stands for "A Compact ENDF" and strips out much of the extra information unnecessary for neutron transport and formats the data into the specified tabulation Laboratory (1998). Since many Monte Carlo codes read ACE-formatted data rather than the original ENDF file, WARP can eliminate one potential cause for discrepancies by loading the same data as other codes. The cross section data are then reformatted to use a

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

unionized energy grid[2] via NumPy van der Walt et al. (2011); the reformatted data are then passed to the C++ routines via the Python C API.

In its current state, WARP does not use S(α,β) thermal scattering tables or unresolved resonance parameters. It only uses the free-gas approximation to account for the motion of material nuclei. The free-gas approximation treats the target as part of a gas at a certain temperature and does not consider other modes of energy transfer that rise from inter- and intra-molecular phenomena. Thermal scattering and unresolved resonance data improve the physical fidelity of the simulation, but these features were turned off in production codes so that WARP could be directly compared to them.

WARP currently has an incomplete set of the ENDF sampling laws implemented: laws 3 (level scattering), 4 (continuous tabular distribution), 7 (simple Maxwell fission spectrum), 9 (evaporation spectrum), 11 (energy-dependent Watt spectrum), 44 (Kalbach-87 formalism), 61 (LAW=44 but tabular angular distribution), and 66 (N-body phase space distribution) Team (2003). These laws cover all the reactions present in the test problems presented here and most nuclides in the ENDF/B-VII.1 cross section data set, but some nuclei may have interactions that require the remaining neutron sampling laws: 1 (tabular equiprobable energy bins), 5 (general evaporation spectrum), 22 (tabular linear functions), 24 (UK law 6), and 67 (laboratory angle-energy law). Problems containing any such nuclides cannot be simulated with the current version of WARP, but could be added later. These unincluded laws are not nearly as common as the ones currently included in WARP, and excluding them at this point should not greatly impact the ability to benchmark WARP's performance and accuracy.

WARP only has collision estimators implemented for both flux and multiplication factor estimation. The relative errors of these estimations are calculated, but any additional convergence estimation beyond relative error has not been implemented. Variance reduction is not implemented in WARP. Neutrons have statistical weight in WARP, and currently the only reactions that adjust neutron weight are multiplicity reactions, like (n,2n), where the neutron weight is multiplied by the exiting neutron number in a manner similar to OpenMC Mas (2013). This means that new neutron histories do not have to be created mid-batch in criticality calculations, and makes program flow simpler. Analog capture was turned on in Serpent and MCNP in all simulations considered to ensure a fair comparison of the results.

3.2. Geometry

The geometry representation in WARP is handled by the NVIDIA OptiX ray tracing framework NVI (Nov. 2012a). OptiX provides good performance on the GPU, but imposes some limitations on how geometries must be input. For the current algorithm used, the geometrical limitations are:

1. Cells are the basic building blocks of a model, not surfaces.
2. All cells must be finite, closed, and non-overlapping.
3. Implicit nesting determines what material exists inside of a cell. In other words, since cells are defined directly, the material of cell A is only that space where cell A is the lowest nested cell. If cell B is within cell A, the entire space within cell B is excluded from cell A simply because cell B resides within cell A. It does not need to be excluded explicitly.
4. The only cell types available are spheres, cylinders, right rectangular prisms, and right hexagonal prisms.
5. Only translational transforms have been implemented.
6. Vacuum (i.e. infinitely absorbing) and specular reflection (i.e. mirror) boundary conditions are available.

An improved tracking routine, which determines which cell a neutron resides in based on its spatial coordinates, has been implemented in WARP since the last publication Bergmann and Vujić (2015). Previously, a ray was traced from the neutron position to the outer cell, and a list of cell numbers was stored, in order, for each intersection. The double entries were removed as the list was written, which, at the end of the trace, yielded a list of cells the neutron was nested in. This resulted in very poor performance in deeply-nested cases due to this list becoming large. For example, WARP had very significant performance decreases in the Jezebel spherical criticality test (discussed later) if the sphere was segmented into many spherical shells. Doing this caused WARP to store a long list of surface numbers

[2] A "unionized energy grid" is a common set of energy points where all cross sections have defined values as opposed to every cross section having its own individual set of energy points.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

and made the simulation very slow. WARP now uses "cell sense" to reduce the memory impact of determining in which cell/material a neutron resides. "Surface sense" is a basic feature of many computational solid geometry (CSG) systems, and is simply the sign of the remainder left after evaluating a set of coordinates in a surface equation. "Cell sense" is like surface sense in that it is positive if a neutron is outside a cell and is negative if a neutron is inside the cell. Cell sense can be calculated by taking the product of the surface senses of the cell's constituent planes. In the new and improved tracking routine, the cell sense (which is either 1 or -1) is calculated and summed as the query ray traverses the geometry. When the sum becomes negative, the last intersected surface is the cell in which the neutron is located. Figure 2 shows an illustration of the new geometry query algorithm. Using the new surface sense traversal method results in high performance even in the previously mentioned segmented Jezebel case.

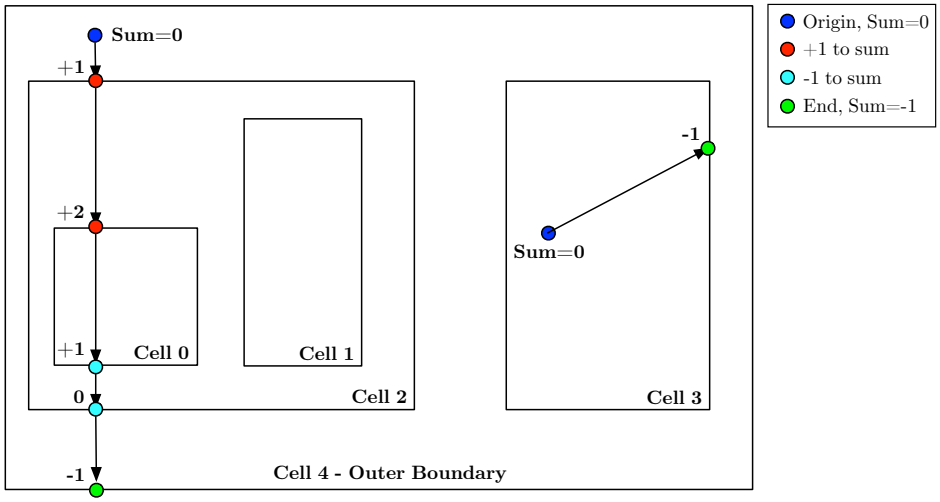


Figure 2: The improved geometry query algorithm showing how the cell number is calculated through ray tracing. A ray is iteratively drawn from the neutron starting position to the outer cell, and the surface normal values are summed along the way. When the sum reaches -1, the cell in which the neutron resides has been found and the trace stops.

When the cumulative sense becomes negative, the containing cell has been intersected and tracing can stop. This approach can be used because all bodies must be closed. This scheme takes advantage of the acceleration structure in OptiX while still using the simplicity of combinatorial solid geometry. It essentially is the same scheme as MCNP/Serpent, except that the acceleration structures allow surfaces that are part of cells that are far away to be skipped. Also, compared to the past scheme of storing a list of doubles, this method requires a single integer to be stored and operated on instead of a potentially large and slow list.

Since OptiX allows variables to be attached to individual geometrical objects, this feature has been exploited in order to create an efficient general tally routine. After the geometry has been input and desired tallies have been specified, WARP creates an array of tally data structures and attaches the tally index directly to each geometrical object that the tally is associated with. When OptiX is called to calculate the nearest intersection points and determine which cell a neutron is in, it also reports back the index of the tally (if any) that should be scored at a collision. This way, the high performance of the OptiX library is leveraged to perform the hash between cell number and tally index, and a separate search does not have to be performed. The material index is returned by OptiX in an identical way.

4. Tests

A small set of CPU and GPU platforms were chosen for use in testing WARP. These platforms were chosen simply because they were easily accessible from UC Berkeley. For the CPU platforms, a single cluster node of both Berkelium (Bk), the departmental cluster, and Savio2, the shared campus high performance computing cluster, were used. For the GPU platforms, a consumer-level graphics card (Titan) and two compute-specific cards (Tesla) were used. Descriptions of each piece of hardware can be found in Table 2. For the GPUs, the costs include the retail price

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

paid by the department plus that of a minimal host computer with at least as much memory as the card (~$400). The costs of the The "physical processor" count shown in Table 2 for the GPUs is the number of SMs present in each card.

Table 2: Platforms used in the benchmark cases and their specifications. Cost are those incurred when purchasing the systems and are approximately 2014 values.

Platform	Cost (USD)	Total Processor Power (Watts)	Local Memory	Memory Frequency
Bk PSSC PowerWulf Blade	9,310	460	96 GB	1.6 GHz
Savio2 Lenovo NeXtScale nx360 M5	6,770	210	64 GB	2.133 GHz
NVIDIA Tesla K20	3,325	225	4.8 GB	2.6 GHz
NVIDIA Titan Black	1,521	250	6.14 GB	3.5 GHz
NVIDIA Tesla K80	5,000	300	12 GB	2.505 GHz

Platform	Physical Processors	Processor Frequency	Maximum Threads
Bk PSSC PowerWulf Blade	4x AMD Opteron 6172	2.1 GHz	48
Savio2 Lenovo NeXtScale nx360 M5	2x Intel Xeon E5-2670 v3	2.3 GHz	24
NVIDIA Tesla K20	13	705.5 MHz	$2^{31} - 1$
NVIDIA Titan Black	15	1071.5 MHz	$2^{31} - 1$
NVIDIA Tesla K80	13	823.5 MHz	$2^{31} - 1$

All tests on all codes use ENDF/B-VII cross sections that are distributed with the Serpent 1.1.7 release from RSICC (Radiation Safety Information Computational Center). RSICC regulates the licensing and distribution of the MCNP and ENDF data worldwide, and also distributes Serpent 1.1.7 ser (2016). The cross sections distributed with Serpent are used since they contain fewer energy grid points than the ENDF/B-VII.1 cross sections distributed with the MCNP 6.1. The Serpent 1.1.7 data therefore require less storage space, and using them allows more isotopes to fit into card memory when running WARP.

4.1. Test 1 - "Jezebel" bare Pu sphere

The "Jezebel" criticality test is a bare plutonium/gallium sphere with vacuum boundary conditions, which can be seen in Figure 3. The fission neutron rate from ^{239}Pu is balanced by the leakage rate from the 6.6595 cm radius to give a k_{eff} of approximately 1. Since this system is so leaky, producing results consistent with MCNP and Serpent ensures that the boundary conditions are correctly being enforced. The Jezebel test is a standard test used to validate neutron transport codes and is described in the International Handbook of Evaluated Criticality Safety Test Experiments under the name "Pu-MET-FAST-001" Agency (1995). The geometry and materials are outlined in Table 3. All cross sections used were processed at 273.5 K.

Table 3: Geometry and materials used in the "Jezebel" test case.

Cells	Isotopes	(Atm. %)	Density
1 sphere, r=6.6595 cm	^{239}Pu (0.7381)	^{240}Pu (0.1942)	15.73 g/cm^3
	^{241}Pu (0.0299)	^{242}Pu (0.0038)	
	^{69}Ga (0.0203)	^{71}Ga (0.0135)	

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

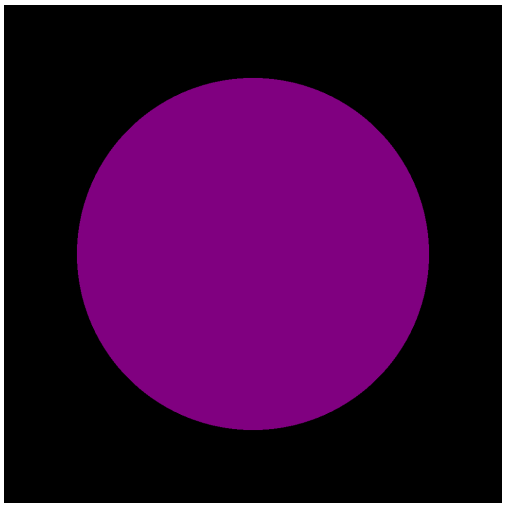


Figure 3: Horizontal slice of the geometry of the "Jezebel" test case. The purple color represents the material in the single cell, and the black color represents space outside the outermost boundary.

4.2. Test 2 - Homogenized fuel block

The homogenized block criticality test is a bare cube with vacuum boundary conditions as seen in Figure 4. This test keeps the same boundary condition, temperature, and number of cells as test 1, but is larger (reducing leakage), contains light isotopes (which readily produce thermal neutrons), and introduces new isotopes. Introducing new isotopes into the simulation is significant in that new isotopes may use any of the various sampling laws outlined earlier. Comparing spectra from calculations containing many different isotopes simply shows that WARP can handle sampling the laws correctly. The geometry and materials are outlined in Table 4. All cross sections used were processed at 273.5 K.

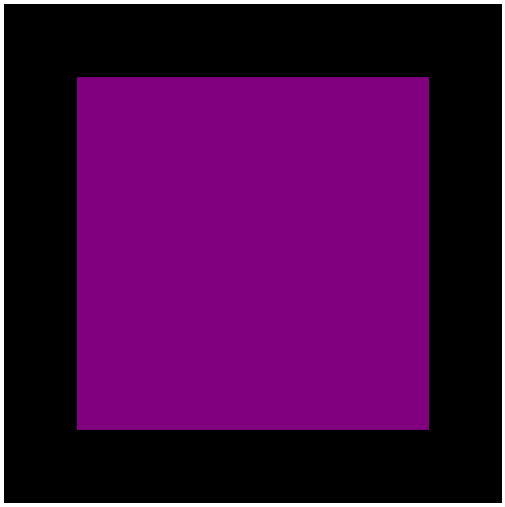


Figure 4: Horizontal slice of the geometry of the homogenized fuel block test case. The purple color represents the material in the single cell, and the black color represents space outside the outermost boundary.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

Table 4: Geometry and materials used in the homogenized fuel block test case.

Cells	Isotopes	(Atm. %)	Density
1 cube, 100x100x50 cm	^{238}U (0.90)	^{235}U (0.10)	5.50 g/cm^3
	^{16}O (3.00)	^{2}H (2.0)	
	^{90}Zr (0.5145)	^{91}Zr (0.1122)	
	^{92}Zr (0.1715)	^{94}Zr (0.1738)	
	^{96}Zr (0.0280)		

4.3. Test 3 - Zr-clad UO_2 pin in heavy water

This criticality test, shown in Figure 5, consists of a UO_2 cylinder clad in zirconium surrounded by a large block of heavy water. This test increases complexity by having three materials, each with multiple isotopes, and three cells. The water block has vacuum boundary conditions. Its dimensions are relatively large and the absorption is low, so the mean neutron lifetime should be large. This test serves to demonstrate that all the processing routines work simultaneously, the effect of introducing more than one cell, and the effect of long-lived neutrons. The geometry and materials are outlined in Table 5. All cross sections used were also processed at 273.5 K.

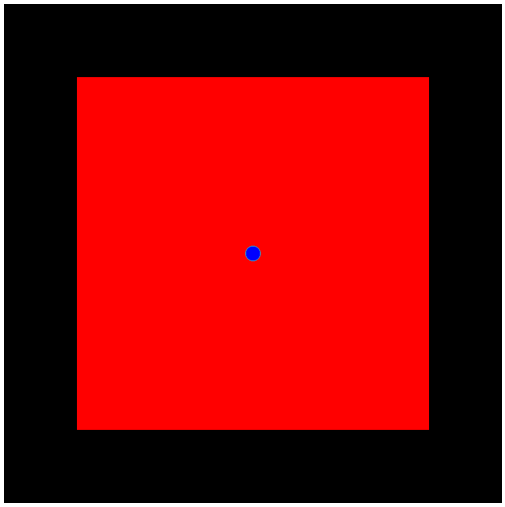


Figure 5: Horizontal slice of the geometry of the Zr-clad UO_2 pin in heavy water test case. Colors represent cell numbers and black represents space outside the outermost boundary.

Table 5: Geometry and materials used in the single UO_2 pin in D_2O test case.

Cells	Isotopes	(Atm. %)	Densities
1 cylinder, r=2.0 cm z=±20	^{238}U (0.90)	^{235}U (0.10)	10.97 g/cm^3
	^{16}O (2.00)		
1 cylinder, r=2.2 cm z=±20.2	^{90}Zr (0.5145)	^{91}Zr (0.1122)	6.52 g/cm^3
	^{92}Zr (0.1715)	^{94}Zr (0.1738)	
	^{96}Zr (0.0280)		
1 box, 50x50x50 cm	^{2}H (2.0)	^{16}O (1.0)	1.11 g/cm^3

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

4.4. *Test 4 - Homogenized fuel pebble in FLiBe*

This criticality test consists of a single sphere of homogenized UO_2 and C surrounded by molten FLiBe (Li_2BeF_4) salt, shown in Figure 6. The outer cell is a right hexagonal prism with specular reflective boundary conditions. This test uses two temperatures simultaneously and tests the reflective boundary condition setting. The geometry and materials are outlined in Table 6, where "r" shown for the hex prism is the length of the apothem. The FLiBe cross sections used were processed at 900 K and the pebble cross sections at 1200 K.

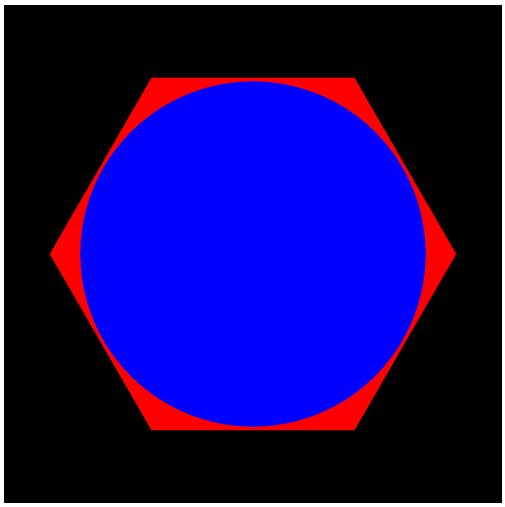


Figure 6: Horizontal slice of the geometry of the homogenized fuel pebble in FLiBe test case. Colors represent cell numbers and black represents space outside the outermost boundary.

Table 6: Geometry and materials used in the pebble in molten FLiBe test case.

Cells	Isotopes	(Atm. %)	Densities
1 sphere, r=5.0 cm	^{238}U (0.90) ^{16}O (2.00) ^{13}C (0.022)	^{235}U (0.10) ^{12}C (1.978)	8.75 g/cm^3
1 right hex prism, r=5.1 cm	^{6}Li (0.15) ^{9}Be (1.00)	^{7}Li (1.85) ^{19}F (4.00)	1.94 g/cm^3

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

4.5. Test 5 - Stainless steel clad metallic uranium pin in liquid sodium

The criticality test in Figure 7 consists of a single cylinder of metallic uranium surrounded by molten sodium. The outer cell is a right hexagonal prism with specular reflective boundary conditions. This test serves to benchmark a fast system with a coolant as well as introduce more and different isotopes. The geometry and materials are outlined in Table 7. The fuel and clad cross sections used were processed at 900 K, and the sodium coolant cross sections were processed at 600 K. In order to keep the specification tenable, only the major components of 316 stainless steel were included in the clad material.

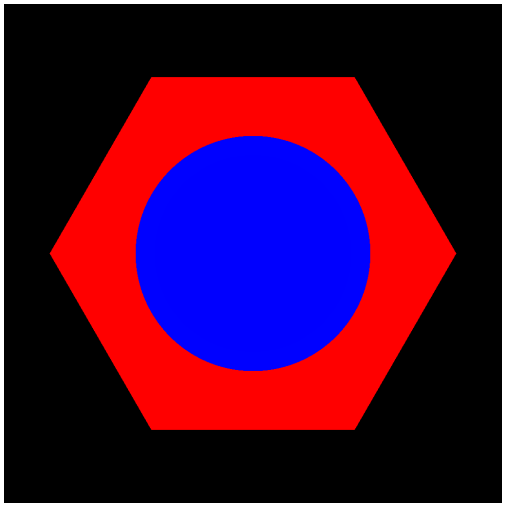


Figure 7: Horizontal slice of the geometry of the stainless steel clad metallic uranium pin in liquid sodium test case. Colors represent cell numbers and black represents space outside the outermost boundary. The material inside the pin cells is the same (UO_2 and Zr), but in this plot the colors represent cell numbers, and therefore each pin has a slightly different color.

Table 7: Geometry and materials used in the 316 stainless steel clad metallic uranium single pin test case.

Cells	Isotopes	(Atm. %)	Densities
1 cylinder, r=1.0 cm	^{238}U (0.90)	^{235}U (0.10)	19.1 g/cm^3
1 cylinder, r=1.2 cm	^{54}Fe (0.0435) ^{57}Fe (0.0165) ^{50}Cr (0.0065) ^{53}Cr (0.0143) ^{58}Ni (0.0681) ^{62}Ni (0.0036)	^{56}Fe (0.6879) ^{58}Fe (0.0021) ^{52}Cr (0.1257) ^{54}Cr (0.0035) ^{60}Ni (0.0262) ^{64}Ni (0.0009)	7.99 g/cm^3
1 right hex prism, r=1.8 cm	^{23}Na (1.00)		0.927 g/cm^3

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

4.6. Test 6 - Zr-clad hexagonal UO_2 pin cell lattice in light water

This criticality test consists of 631 Zr-clad UO_2 cylinders laid out in a hexagonal lattice surrounded by light water, as shown in Figure 8. The material compositions, densities, and cylinder dimensions are similar to the pin cell test case, but, since this test has two orders of magnitude more objects, it serves to highlight the effect of introducing many geometric objects into the problem and further validates that the geometry processing routines work correctly. The lattice is in the x-y plane, has a pitch to diameter ratio of 1.164, and has 15 elements on a side. The geometry and materials are outlined in Table 8. All cross sections used were processed at 273.5 K.

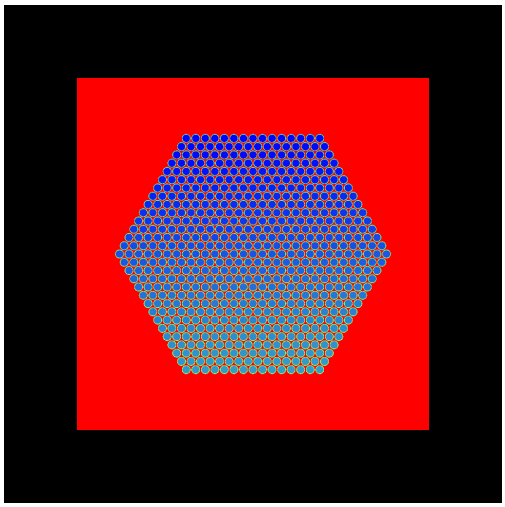


Figure 8: Horizontal slice of the geometry of the Zr-clad hexagonal UO_2 pin cell lattice in light water test case. Colors represent cell numbers and black represents space outside the outermost boundary.

Table 8: Geometry and materials used in the Zr-clad UO_2 pin hexagonal lattice in light water test case.

Cells	Isotopes	(Atm. %)	Densities
631 cylinders, r= 1.0 cm	^{238}U (0.90) ^{16}O (2.00)	^{235}U (0.10)	10.97 g/cm^3
631 cylinders, r= 1.2 cm	^{90}Zr (0.5145) ^{92}Zr (0.1715) ^{96}Zr (0.0280)	^{91}Zr (0.1122) ^{94}Zr (0.1738)	6.52 g/cm^3
1 box, 48x48x48 cm	^{1}H (2.0)	^{16}O (1.0)	1 g/cm^3

5. Results

The accuracy of the WARP calculations were measured by comparing the multiplication factors and neutron spectra of the aforementioned geometries against those calculated by Serpent 2.1.24 and MCNP 6.1. The multiplication factor differences shown in this section are reported in "per cent mille" (PCM), which is a thousandth of a percent, or 10^{-5}. This is a standard way of reporting differences in the multiplication factor, as any small change in it can cause a significant change in system behavior. The flux spectra are normalized per fission neutron and per unit lethargy. "Lethargy" is defined as the logarithm of the neutron energy, $\ln(E_0/E)$, where E_0 is the highest energy possible in a system and E is another energy Duderstadt and Hamilton (1976). Normalizing the flux per unit lethargy in conjunction with plotting on a logarithmic scale yields a plot where the area under the curve gives the fraction of neutrons flux within the bounding energy range.

The performance of WARP is measured by comparing the runtimes of the WARP simulations against those of the production codes. The runtimes do not include the time it takes to load and process cross sections before the

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

simulations. In other words, it is only the time it takes to complete all transport cycles. The difference in runtimes are reported in speedup factors (t/t_{WARP}). The times reported are either for simulations performed on a single card or a single node. This was done to compare the platforms fairly. Resources are shared at the single card/node level, and therefore increasing process and/or thread count does not necessarily scale linearly as resource contention increases. It can be optimistically assumed that scaling occurs linearly above this level if little inter-card/node communication is needed, and therefore speedup ratios comparing single-node/card performance should be equivalent to multi-node/card performance.

It must be noted that WARP uses single-precision data and math, while the production CPU codes both use double-precision data and math. The theoretical throughput of single-precision math is typically double that of double-precision math on modern CPU platforms as well as the Tesla GPU cards NVI (2013); Langou et al. (2006); Itu et al. (2011), but this kind of speedup is very problem- and platform-dependent. For dense linear algebra operations, the factor of two is usually realized, but for sparse multiplications or Monte Carlo simulations where data access is more irregular, a less dramatic speedup is seen by switching to single precision math sin (accessed 10/2016a,a); Buttari et al. (2008); Chan (2013). This implies that the production codes aren't at a huge disadvantage compared to WARP simply becasue they must use double-precision math. The codes are complex and overall performance depends on many factors, but single precision math is used in WARP since using it means the consumer-level cards have similar performance parameters as the compute-specific cards, and therefore can provide substantial hardware investment savings. For instance, a Tesla K40 card is theoretically capable of 5040 FLOPs in single precision or 1680 FLOPs in double precision. A GeForce GTX 780 Ti card is theoretically capable of 5046 FLOPs in single precision, but only 210 FLOPs in double precision wik (accessed 01/2017). When they each were first released, the Tesla cost $5,500 whereas the GeForce only cost $700. If used in single-precision mode, the GeForce is the clear winner due to its much lower price and equivalent performance. In contrast, a Titan Black card is theoretically capable of 5121 FLOPs in single precision or 1707 FLOPs in double precision and only cost $1,000 when it was released. Compared to the K40, it seems the Titan only lacks ECC and RDMA capabilities which could be important for numerically sensitive or communication-intensive calculations. In the case of the Titan Black, where double precision performance is similar to the K40, it would be interesting to see the real-world impact of switching to double precision data and math in WARP and could be an area of future investigation.

The benchmark cases were run with 6.5×10^6 neutrons per criticality batch with 20 initial batches discarded and 40 batches with statistics accumulated (60 batches total). From the scaling results shown next in this section, the neutron processing rate of the GPU cards all saturated near this number of neutrons per batch. The CPU codes were run with identical batch parameters. The CPU cases were run with the optimal number of MPI processes and threads per process for the specific node to fairly measure the performance of the unit as a whole. Since the PowerWulf PSSC node contains 4 processors that each contain 2 non-uniform memory access (NUMA) nodes, the most efficient configuration for running the CPU codes was to use 8 MPI processes (one for each NUMA node), which each ran 6 threads for the 6 cores in each NUMA node. The Lenovo Savio2 runs were done in a similar way, but with 4 MPI processes that each ran 6 threads. The MCNP runs were launched with one additional MPI process since the master process does not perform any neutron tracking in MCNP. All codes were set to not use thermal scattering or unresolved resonance tables and to use analog capture only.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

5.1. Scaling

Larger neutron datasets allow the GPU to schedule threads better, but the size of the dataset is limited by the available on-card memory. Cross sections are also stored on-card and compete with the neutrons for space in memory. It is of interest to know where performance saturates so a user can choose to run the number of neutrons per cycle that gives the best performance. Figure 9 shows the neutron processing rate of WARP compared to the neutron dataset size.

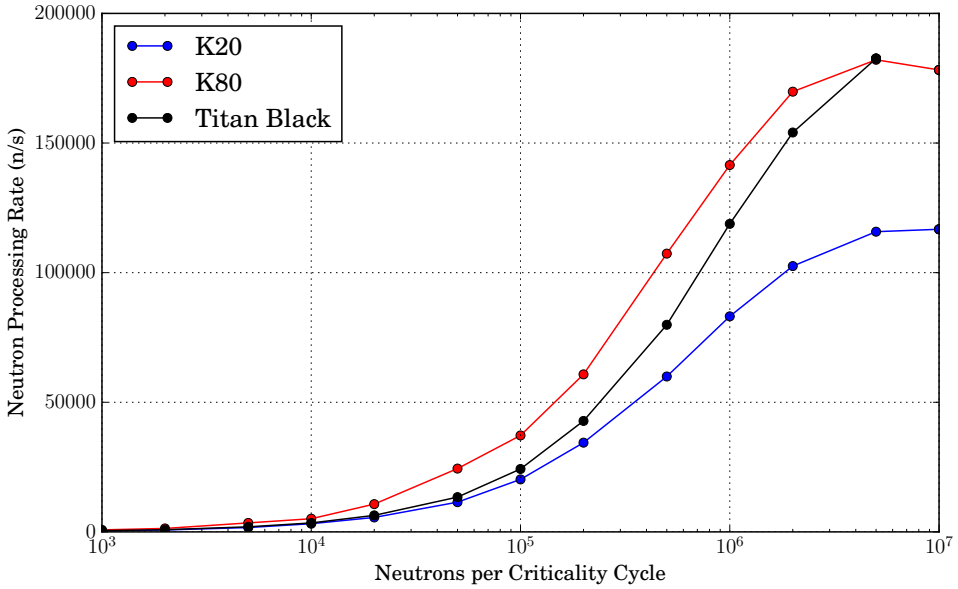


Figure 9: Neutron processing rate vs. number of neutrons per cycle for the hexagonal lattice test case.

Performance increases until about 5×10^6 neutrons per cycle, then flattens above this for all cards. This curve was calculated using the hexagonal pin lattice test case, and may change if a different model were input. Despite this, 5×10^6 neutrons per cycle is a rough estimate of where maximum performance occurs for WARP, and 6.5×10^6 neutrons per cycle was used for the test cases. There is no data point for the Titan Black at 10^7 neutrons per cycle since a crash always occured and the simulation could not be completed. The hexagonal pin lattice case was chosen for the scaling study because it is the most complicated geometry considered, and was assumed to be the most sensitive to thread saturation and therefore would require the most neutrons per cycle for the neutron processing rate to level out.

5.2. Multiplication factors

Table 9 shows the multiplication factor deviations for the six criticality tests compared to MCNP 6.1 and Serpent 2.1.24 for WARP running on an NVIDIA Titan Black, K20, and K80 cards. The "ΔMCNP" rows contains WARP's difference (in PCM) from MCNP, and the "Δ Serpent" rows contains WARP's difference from Serpent. These difference rows also contain 1σ standard deviation values in terms of relative error. The standard deviation values indicate the level at which the multiplication factors calculated by WARP agree with either Serpent or MCNP, and have been used to produce the "y" and "n"flags shown after the standard deviation value. A "y" in the 1σ position indicates that the multiplication factor values are mutually within 1 standard deviation of each other, i.e. that $\Delta\text{MCNP} < \sigma_{\text{MCNP}} + \sigma_{\text{WARP}}$. In other words, this is the space where the estimated distributions of the two codes overlap and is the interval where they could produce the same results. For one standard deviation, there is a 68% chance that the next calculated value will be within a single standard deviation, a 32% chance it will be outside, and a 16% chance it will be outside and on a single side of the mean. Therefore, there is a 0.16^2=2.6% probability that the codes could produce values that are outside a the sum of their standard deviations. Outside two standard deviation is even

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

more unlikely (.05%), so any results that are outside two mutual standard deviations are most likely not statistically similar at all. The results are presented in this way to incorporate the uncertainties of both codes into the results, and to give an easy way of seeing which test cases are problematic. If a problem is within a single mutual standard deviation, it is assumed the solutions are equivalent. If the results are outside two mutual standard deviations, it is assumed the solutions are different.

Table 9: The test case multiplication factors for WARP versus those of Serpent 2.1.24 and MCNP 6.1.

	Jezebel	Homogenized Block	Pin Cell
Serpent 2.1.24	0.9997840 ± 9.50E-5	0.5934090 ± 1.20E-4	0.2750510 ± 1.80E-4
MCNP 6.1	0.9999600 ± 7.00E-5	0.5933000 ± 4.00E-5	0.2751500 ± 7.00E-5
WARP Titan Black	1.0000620 ± 9.58E-5	0.5933724 ± 1.24E-4	0.2750294 ± 1.46E-4
Δ Serpent	27.80, 1σ n, 2σ y	-3.66, 1σ y, 2σ y	-2.16, 1σ y, 2σ y
Δ MCNP	10.20, 1σ y, 2σ y	7.24, 1σ y, 2σ y	-14.06, 1σ n, 2σ n
WARP K20	1.0000212 ± 7.82E-5	0.5934539 ± 1.66E-4	0.2751048 ± 2.07E-4
Δ Serpent	23.72, 1σ n, 2σ y	4.49 , 1σ y, 2σ y	5.38, 1σ y, 2σ y
Δ MCNP	6.12, 1σ y, 2σ y	15.39, 1σ n, 2σ y	-6.52, 1σ y, 2σ y
WARP K80	0.9999602 ± 9.58E-5	0.5932266 ± 1.24E-4	0.2749884 ± 1.56E-4
Δ Serpent	17.62, 1σ y, 2σ y	-18.24, 1σ n, 2σ y	-6.36, 1σ y, 2σ y
Δ MCNP	0.02, 1σ y, 2σ y	-7.34, 1σ y, 2σ y	-18.26, 1σ n, 2σ n

	FLiBe Cell	Sodium Pin Cell	Hex Assembly
Serpent 2.1.24	0.8804900 ± 8.70E-5	1.0987100 ± 2.20E-4	1.0503300 ± 7.20E-5
MCNP 6.1	0.8805100 ± 6.00E-5	1.0987700 ± 6.00E-5	1.0506500 ± 9.00E-5
WARP Titan Black	0.8807005 ± 9.58E-5	1.0986860 ± 7.82E-5	1.0510795 ± 9.58E-5
Δ Serpent	21.05, 1σ n, 2σ y	-2.40, 1σ y, 2σ y	74.95, 1σ n, 2σ n
Δ MCNP	19.05, 1σ n, 2σ y	-8.40, 1σ y, 2σ y	42.95, 1σ n, 2σ n
WARP K20	0.8807086 ± 9.58E-5	1.0987912 ± 5.53E-5	1.0510620 ± 9.58E-5
Δ Serpent	21.86, 1σ n, 2σ y	8.12, 1σ y, 2σ y	73.20, 1σ n, 2σ n
Δ MCNP	19.86, 1σ n, 2σ y	2.12, 1σ y, 2σ y	41.20, 1σ n, 2σ n
WARP K80	0.8807546 ± 9.58E-5	1.0985898 ± 5.53E-5	1.0511035 ± 1.24E-4
Δ Serpent	26.56, 1σ n, 2σ y	-12.02, 1σ y, 2σ y	77.35, 1σ n, 2σ n
Δ MCNP	24.56, 1σ n, 2σ y	-18.02, 1σ y, 2σ y	45.35, 1σ n, 2σ n

From Table 9, it can be seen that WARP agrees with either MCNP *or* Serpent (they sometimes do not exactly agree with each other!) at the 1σ level for all cases except the FliBe cell and the hex assembly cases. The FliBe cell case agrees on the 2σ level, but the hex assembly case does not, indicating that there is a problem with the way WARP is solving the hex assembly case. Speculation on what could be causing this is discussed in Section 6.

5.3. Runtimes

Table 10 shows the transport cycle runtimes for the six criticality tests compared to MCNP 6.1 and Serpent 2.1.24 for WARP running on an NVIDIA Titan Black, K20, and K80 cards. The values in the time rows are reported in minutes. The "Δ MCNP" row contains WARP's speedup factor over MCNP, and the "Δ Serpent" row contains WARP's speedup factor over Serpent.

It can be seen from Table 10 that WARP runs fastest on the K80 card, and both Serpent and MCNP run fastest on the Savio2 node. This makes sense since these are the two newest pieces of hardware considered. It is important to note that the Titan Black card performs almost as well as the K80 card, however. Comparing the K80 card to a Savio2 node (the most relevant comparison), WARP runs 0.84 to 5.34 times as fast as Serpent and 2.45 to 7.61 times as fast as MCNP. The largest speedup over Serpent is for the Jezebel test, the largest speedup over MCNP is for the homogenized block test, and the smallest speedup is for the sodium pin cell test over both Serpent and MCNP. This speedup factor is calculated over an entire compute node, so WARP on a GPU gives a performance equivalent to 0.84 to 7.61 compute nodes, depending on the type of simulation hardware considered. For the sodium pin case, where there is a relatively large amount of cross section processing, a Savio2 node running Serpent actually performs better than WARP on a K80 card. For problems where there is less cross section processing (higher geometry processing fraction), WARP tends to perform better than a Savio2 node running Serpent. WARP always performs better than a Savio2 node running MCNP, even with relatively heavy cross section processing, but this isn't a great surprise considering that MCNP doesn't use a unionized energy grid structure when processing cross sections.

5.4. Spectra

In the following subsections, neutron flux spectra calculated by WARP are compared to those calculated by Serpent and MCNP for identical geometries and materials. Every spectrum is binned into 1024 equi-log bins from 10^{-11} to

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

Table 10: The test case transport cycle run times (in minutes) for WARP, Serpent 2.1.24, and MCNP 6.1. Values in the "Δ" rows contain the speedup factor of WARP compared the the corresponding production code running on a Berkelium (Bk) PSSC node and a Savio2 node.

		Jezebel	Homogenized Block	Pin Cell
Serpent 2.1.24	Bk PSSC	7.24	39.41	146.15
	Savio2	6.32	24.10	76.51
MCNP 6.1	Bk PSSC	30.57	98.40	131.27
	Savio2	4.95	44.03	64.18
WARP Titan Black		1.47	7.53	26.47
Δ Serpent (Bk, S2)		4.91, 4.29	5.23, 3.20	5.52, 2.89
Δ MCNP (Bk, S2)		20.74, 3.36	13.07, 5.85	4.96, 2.42
WARP K20		2.45	12.60	42.55
Δ Serpent (Bk, S2)		2.96, 2.58	3.13, 1.91	3.44, 1.80
Δ MCNP (Bk, S2)		12.49, 2.02	7.81, 3.50	3.09, 1.51
WARP K80		1.18	5.79	20.91
Δ Serpent (Bk, S2)		6.11, 5.34	6.81, 4.17	6.99, 3.66
Δ MCNP (Bk, S2)		25.80, 4.18	17.01, 7.61	6.28, 3.07

		FLiBe Cell	Sodium Pin Cell	Hex Assembly
Serpent 2.1.24	Bk PSSC	79.16	120.86	81.54
	Savio2	45.08	68.46	44.32
MCNP 6.1	Bk PSSC	269.47	320.40	160.68
	Savio2	81.03	199.15	112.25
WARP Titan Black		25.78	81.18	34.91
Δ Serpent (Bk, S2)		3.07, 1.75	1.49, 0.84	2.34, 1.27
Δ MCNP (Bk, S2)		10.45, 3.14	3.95, 2.45	4.60, 3.22
WARP K20		39.83	121.34	56.03
Δ Serpent (Bk, S2)		1.99, 1.13	1.00, 0.56	1.46, 0.79
Δ MCNP (Bk, S2)		6.77, 2.03	2.64, 1.64	2.87, 2.00
WARP K80		24.68	81.40	35.63
Δ Serpent (Bk, S2)		3.21, 1.83	1.48, 0.84	2.29, 1.24
Δ MCNP (Bk, S2)		10.92, 3.28	3.94, 2.45	4.51, 3.15

20 MeV. In tests where there is a large region of little to no flux (like in the fast spectrum inside the Jezebel sphere), the energy bin structure is not changed, but the energy range of the plot is limited to the region where there is nonzero flux. The relative difference compared to the MCNP spectrum is shown in the top subplot below the main spectrum plot, and the relative difference from the Serpent spectrum is shown in the lower subplot. The green shaded area in the error subplots shows the space within 2 standard deviations of the statistical uncertainty of the production code. Unlike the intervals reported in the previous subsection, the interval shown in green is only for the production code results. In other words, the codes should have results within the 2σ interval 95.45% of the time (as opposed to 99.95% of the time for two mutual standard deviations).

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

5.4.1. Test 1 - "Jezebel" bare Pu sphere

Figure 10 shows the volume-averaged flux spectra in the Jezebel sphere. The relative difference of WARP compared to Serpent and MCNP is very low, with the normalized tally bins being less than 0.5% from each other in regions where the flux is large. Of course, when the flux is small, the statistical uncertainty becomes much higher, and the relative difference becomes noisy. The relative error is almost always inside the 2σ confidence interval as well, indicating that the simulations are statistically identical.

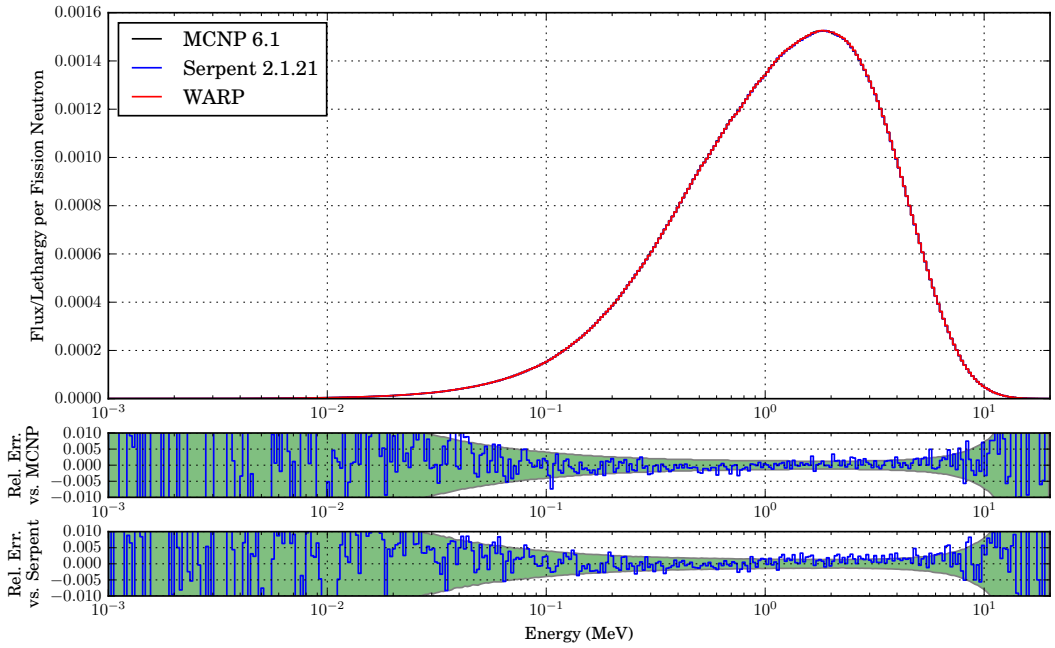


Figure 10: Volume-averaged flux spectra inside the sphere of the Jezebel test case.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

5.4.2. *Test 2 - Homogenized fuel block*

Figure 11 shows the volume-averaged flux spectra in the homogenized fuel block. The relative difference compared to MCNP and Serpent is again shown in the lower subplots. The relative difference compared to MCNP is again generally less than 0.15% and appears to have a zero mean, but the error compared to Serpent has a constant positive offset, indicating a slightly different normalization. This kind of offset appears in other spectra as well, but only when comparing to Serpent. This indicates a possible normalization discrepancy between MCNP and Serpent. There also appears to be some deviations with similar structure from 0.1 to 0.3 MeV, indicating that some kind of reaction sampling may not be treated exactly the same way as in Serpent and MCNP.

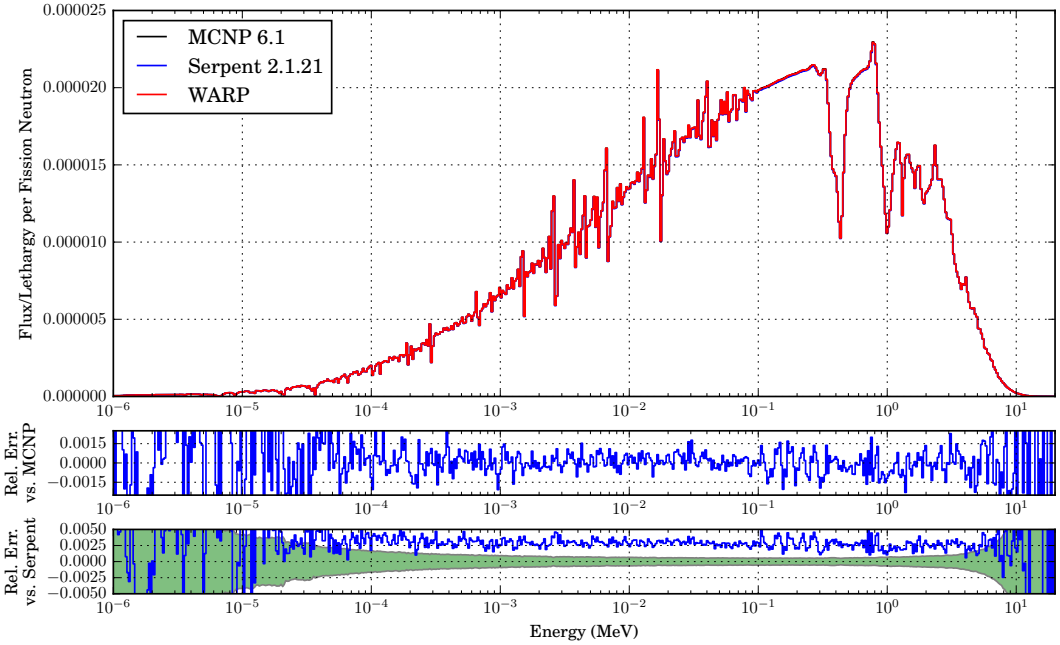


Figure 11: Volume-averaged flux spectra inside the homogenized fuel block.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

5.4.3. *Test 3 - Zr-clad UO_2 pin in heavy water*

Figure 12 shows the volume-averaged flux spectra inside the fuel rod in the Zr-clad UO_2 pin in the heavy water test case. Again, the relative difference is generally less than 1% where the flux is large. Compared to Serpent, the WARP spectrum again appears to have a small, constant offset. The entire energy range agrees well with MCNP, however, with the relative error is almost always inside the 2σ confidence interval. This indicaties that all the reaction laws were processed identically to MCNP for the nuclides present in this test case. The 2σ confidence interval is larger here than the previous cases since the tally volume is much smaller than previously and it simply has fewer scores per source particle.

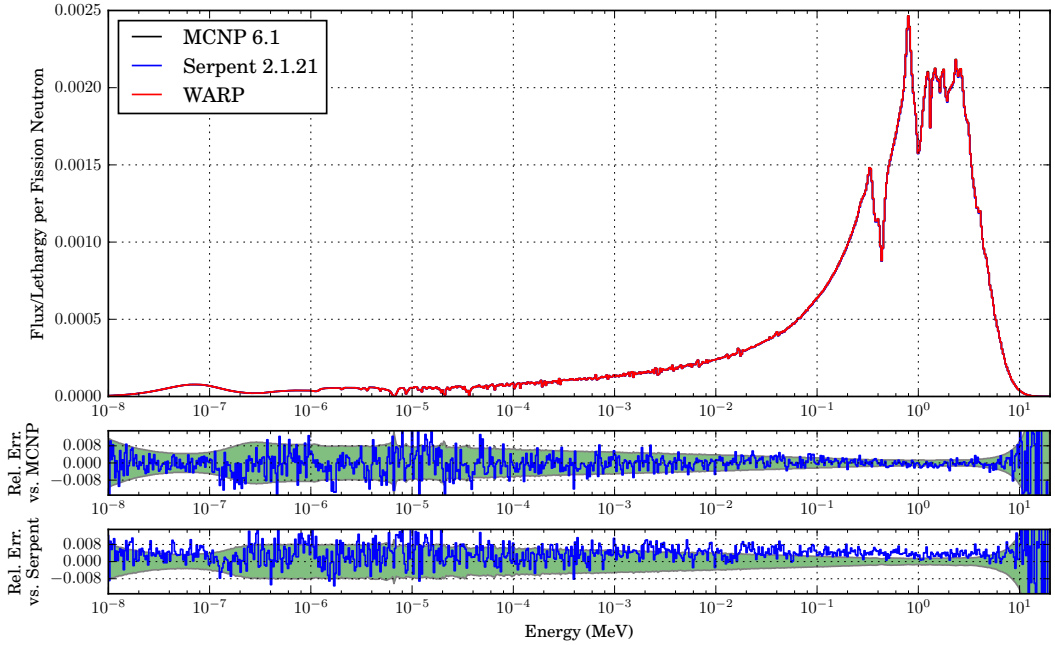


Figure 12: Volume-averaged flux spectra inside the oxide fuel of the Zr-clad pin in the pin in heavy water test case.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

5.4.4. *Test 4 - Homogenized fuel pebble in FLiBe*

Figure 13 shows the volume-averaged flux spectra in the fuel pebble for the reflective FLiBe test case. Here, the relative difference is generally less than 0.1% where the flux is large. The WARP spectrum again has a constant offset compared to the Serpent spectrum, but compares well to MCNP apart from the significant deviations around large resonances in the 20 keV to 1 MeV range.

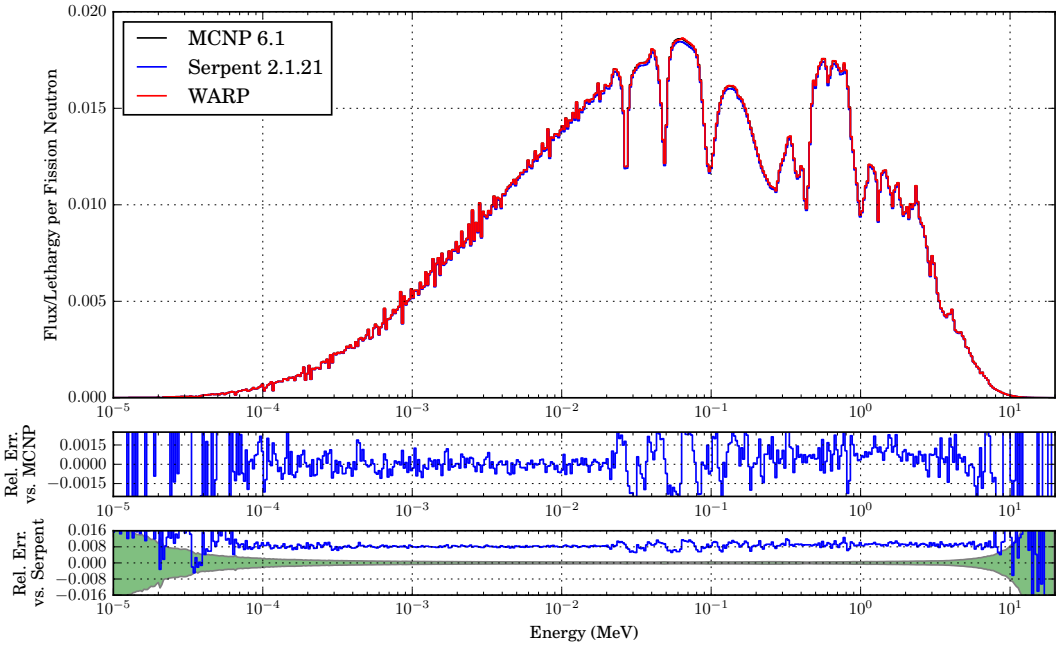


Figure 13: Volume-averaged flux spectra inside the fuel pebble of the reflective FLiBe test case.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

5.4.5. *Test 5 - Stainless steel clad metallic uranium pin in liquid sodium*

Figure 14 shows the volume-averaged flux spectra in the fuel for the reflective sodium cooled, steel clad, metallic fuel test case. The relative difference is generally less than 0.1% where the flux is large. The WARP spectrum again has a constant offset compared to the Serpent spectrum, but compares well to MCNP apart from the significant deviations around large resonances in the 20 keV to 1 MeV range. This again highlights that there is probably a reaction law that isn't processed identically to MCNP.

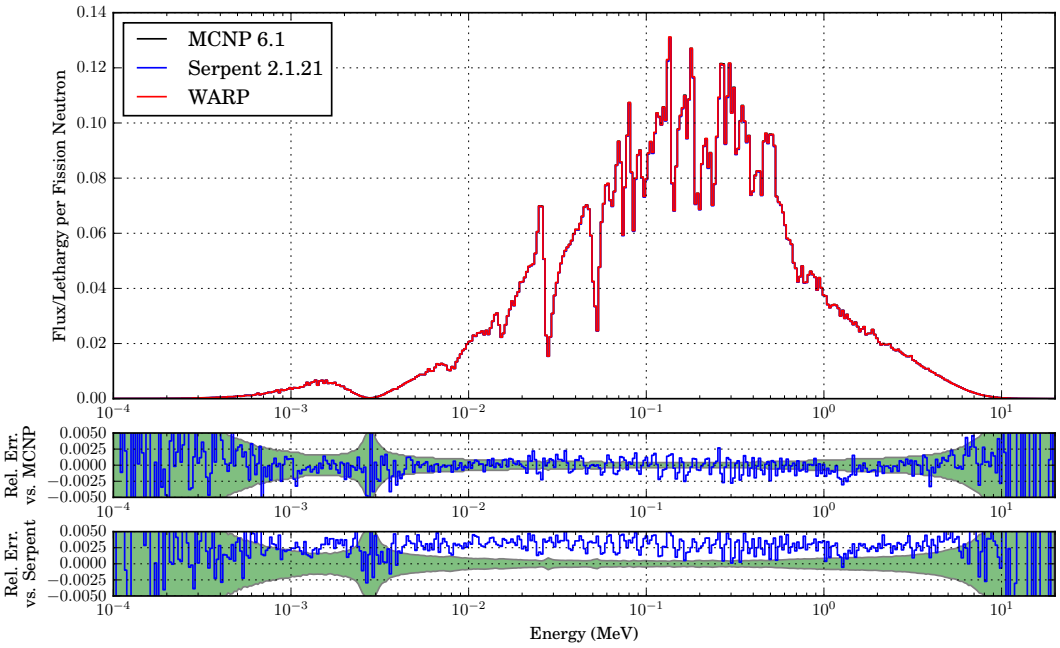


Figure 14: Volume-averaged flux spectra inside the fuel of the reflective, steel clad, sodium cooled pin test case.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

5.4.6. *Test 6 - Zr-clad hexagonal UO_2 pin cell lattice in light water*

Figure 15 shows the volume-averaged flux spectrum in the center pin of the Zr-clad hexagonal oxide fuel pin lattice in light water test case. The relative difference is generally less than 5% where the flux is large, but this time the WARP results are always within the statistical error of both MCNP and Serpent. There is no constant offset compared to Serpent or MCNP and no large deviations, indicating that all the reaction laws were processed identically to both Serpent and MCNP for the nuclides present in this test case. This is also the only test case other than Jezebel where MCNP and Serpent give identical spectra. Again the 2σ confidence interval is large compared to other test problems because the tally volume is small compared to the overall geometry.

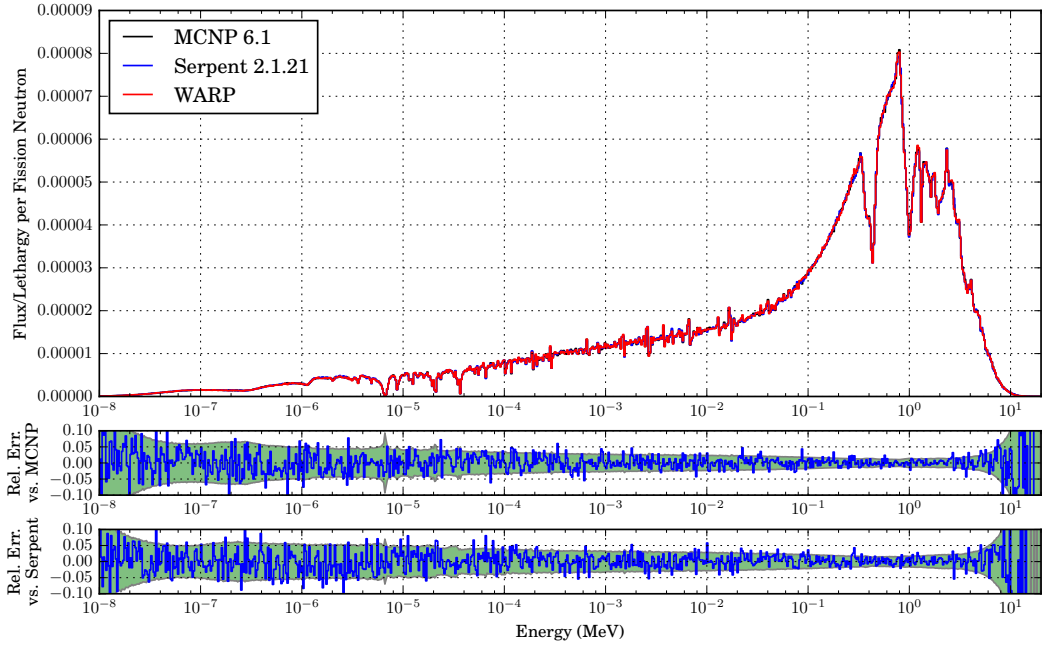


Figure 15: Volume-averaged flux spectra inside the center fuel pin of the hexagonal pin lattice test case.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

5.5. History power

For productivity, the "history power," or number of neutron histories processed per unit time, is of primary interest when purchasing a system since it determines how quickly results can be produced. In addition to this, the history power per hardware cost is of interest for initial capital investment, and the history power per unit energy is of interest for operational costs. Table 11 shows the history power and relative costs of GPU systems running WARP compared to MCNP and Serpent running on the aforementioned CPU platforms.

Table 11: The history power and costs of performing Monte Carlo neutron transport on GPU and CPU systems.

		H. Power	H.P. / Capital (USD)	H.P. / Watt
Serpent 2.1.24	Bk PSSC	2.32E+6	249	5,033
	Savio2	8.84E+6	1,305	42,082
MCNP 6.1	Bk PSSC	2.32E+6	249	5,033
	Savio2	4.63E+6	684	22,039
WARP	Titan Black	1.32E+7	8,675	52,778
	K20	8.52E+6	2,561	37,847
	K80	1.38E+7	2,760	45,995

The history powers shown in Table 11 were calculated by summing the total number of neutrons processed in all six test cases and dividing this number by the sum of the runtimes of the six cases for each code/platform combination (i.e. 3.9×10^8/sum of run times). This way, the average is not biased towards a special case that allows a code to process neutrons especially quickly. The history power per capital is simply the averaged neutron processing rate divided by the hardware cost of the platform. The history power per Watt is similar, except divided by the platform's total maximum electrical power consumption rather than the cost. The costs and electrical power consumptions used to calculate these values were shown in Table 2.

Table 11 shows that, on average, for the test cases considered, a new NVIDIA GPU card can process neutrons about 1.6 times as fast as a new multiprocessor/multicore CPU node running Serpent and about 3 times as fast as a CPU node running MCNP. From a capital investment standpoint, a K80 card costs about half as much as a Savio2 node per neutron processing power and consumes about the same amount of electricity. The Titan Black, however, performs about as well as a K80 from the neutron processing rate standpoint, but costs about a third as much and consumes 20% less electricity. Since WARP uses single-precision data and math, consumer-level GPUs have similar performance compared to compute-specific ones when running WARP. Because of this, the capital and electrical costs of running WARP are greatly reduced compared to a case where double precision data and math are required.

6. Discussion

The differences seen in the multiplication factors and neutron flux spectra calculated by WARP compared to those calculated by Serpent and MCNP are most likely due to either a reaction law being inexactly processed or from a type of reaction incorrectly sampled from time to time. Whether this is due to a software "bug", or if it has to do with single precision math has not been determined. Some sampling laws potentially require division by small numbers when comparing or adding the result to other numbers. Roundoff error from single precision math could make these operations inaccurate and ultimately lead to visible differences in the neutron spectra and multiplication factor. The sampling routines have been spot tested to produce very similar results to the production code routines given narrow incoming energy bands, but rigorous testing of all energies has not been done. There has been some effort in implementing the "Google Test" testing framework gte (accessed 10/2016) in WARP, and it has been used in older versions to performing unit testing. Since WARP has been in heavy development, creating standard unit tests was difficult since the inputs and outputs of routines were constantly changing. A full testing suite to ensure proper data sampling is an area of future development that can be done once WARP becomes more stable. Until then, WARP is sufficiently accurate to be a good testbed for Monte Carlo neutron transport calculations on GPU platforms.

Monte Carlo is a great candidate simulation method to take advantage of single precision math since solution convergence does not normally depend on derivatives, except perhaps in a cross section processing law, and the calculated results are typically not required to be more precise than single-precision numbers. The major places

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

roundoff error can affect the solution is in tally accumulation and cross section processing. Currently, WARP calculates the total macroscopic cross section of a material on the fly, and roundoff could become a problem around resonances where the cross section of one nuclide (or several) is much larger than the that of the other nuclides in the material. In this case, the additions from the minor cross section nuclides could be neglected and could lead to oversampling their reactions in the subsequent routines (since the material's calculated total macroscopic cross section would be smaller than it should be). This problem could be circumvented by preprocessing the material's total macroscopic cross section using double precision math on the host before the data is given to the GPU as single precision numbers. This, however, would increase the memory footprint of the data.

Another place roundoff error could cause problems is in tally accumulation. WARP calculates the flux in a cell by calculating the inverse of the macroscopic cross section at a collision site and adding it to the sum of all previous collisions. If a large resonance lies within an energy bin, very dissimilar values could be added together and precision could be lost. This could be circumvented by using double precision data for the tally arrays. The cost of doing so would have to be measured, but hopefully it would be small since tally accumulation only happens once per transport cycle and is therefore a relatively infrequent operation.

WARP uses integer math to calculate multiplication factors. Long integer values for the total number of fissions produced in a simulation are stored on the host, and integer yield values from each batch are accumulated into it. At the end of the simulation, the long integer value containing the total number of fission neutrons produced is divided by the total number of source neutrons run, yielding the final value for the multiplication factor. Since integer math is used during this whole process, roundoff error from accumulation can not affect it as long as overflow doesn't occur (hence using the long integer data type). Roundoff error in the reaction selection and processing routines could affect the individual yield values and their frequency, however.

It is worth noting that MCNP and Serpent calculated somewhat different solutions in these studies, which shows that getting exactly the same answer is quite difficult since each code handles things slightly differently. This makes it difficult to know what is really correct, but as long as the results are under the error level of what can be detected in the physical world, any differences in the codes should be irrelevant. One potential niche use for WARP would be to do cheap and fast calculations where large total throughput is needed, not precision. For example, in a large parametric study, once an optimal region has been found in the parameter space, WARP's job would be done. This knowledge could then be used as a starting point for more precise calculations with CPU codes.

7. Conclusions and future development

It has been shown that WARP is able to accelerate high-fidelity neutron transport on GPUs. Depending on the problem type and the card used, WARP can achieve performance equivalent to that of 0.84 to 7.61 of modern CPU nodes on a single GPU card, depending on the CPU code and type of calculation being considered. The low capital and electrical cost of GPU platforms then lead to about 2 to 3 times more histories calculated per dollar spent. This clearly shows that GPUs can be very cost effective for continuous energy Monte Carlo neutron transport. This efficiency comes at the cost of precision, however, but WARP could still be a useful tool for scenarios where high computational throughput and/or low-cost calculations are of importance.

WARP, which is developmental, may still have some "bugs", as evidenced by the slight deviations of the multiplication factors and spectra compared to MCNP and Serpent. A large part of future development will be eliminating these discrepancies via rigorous testing. The initial goals of WARP have been completed, but there is much work still to be done if it is going to be of real use to the nuclear engineering community. Basic functionality is currently good enough to assure that GPUs can accelerate high-fidelity Monte Carlo neutron transport calculations, but many capabilities need to be expanded and ensured to scale well to large numbers of neutrons, isotopes, and geometric zones. The rest of this section goes over the major areas where WARP could be improved in order to ensure good scaling in these dimensions.

Geometry is currently handled by NVIDIA OptiX, which provides a convenient way to obtain high-performance results. However, OptiX had to be coerced into providing WARP with the extra information it needed beyond the distance to the nearest intersection, namely the material and tally indices. The way OptiX is used to determine these numbers is not efficient since it must be done iteratively using OptiX's native functions instead of calculating the sum of surface normals in a single trace. NVIDIA has released "OptiX Prime" with OptiX 3.5 NVI (Nov. 2012b),

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

which promises to provide a more "to the metal" ray tracing experience, and might be leveraged to provide more efficient single-traverse functionality. OptiX could also be replaced by Rayforce Gribble and Butler (2013), a high-performance GPU ray tracing library developed by VSL that has the desired functionality built-in and is currently available free of charge for noncommercial use.

The OptiX geometry routines could also be replaced by handwritten routines that use combinatorial solid geometry like Serpent and MCNP. This would make writing input for WARP more like what most nuclear engineers are already used to and, more importantly, could provide a potential performance increase. A universe-based CSG representation may map very well to the GPU and may even be able to fit inside of shared memory for small numbers of surfaces. Using an efficient CSG method would further lend itself to using Woodcock delta-tracking for the neutrons and thus getting rid of the tracing algorithms and libraries altogether Leppänen (2010).

WARP also suffers from "tail effect," where performance drops off as the number of remaining active neutrons in a batch decreases Bergmann and Vujić (2015). Developing a method that keeps the active neutron number high would greatly improve performance, but could become quite complicated as generations would start to overlap and calculation of the multiplication factor and converging the source distribution could not be done between cycles as it is now.

If an entire overhaul of the WARP transport algorithm is feasible, using a streaming multiprocessor (SM)-based algorithm might be investigated for replacing the current global one. This type of algorithm would treat each SM as an independent processor and would provide each a bank of neutrons to transport, as is done by Liu and Henderson Liu et al. (2012); Henderson et al. (2013). This way, neutron data could be stored in very fast shared memory, though this would compete with storing geometric information. Also, since a smaller set of neutrons could be stored, the SMs would need to communicate to determine which of the next neutrons they would take out of the global bank, or they would need to periodically rendezvous to shared source information and ensure that the distributions they use are each converged. This type of transport algorithm would also preclude using OptiX, since it does not have SM-level functionality NVI (Nov. 2012a).

An efficient way to handle situations where there are many different materials and isotopes present needs to be explored. The work done by Scudiero on porting OpenMC's macroscopic cross section processing benchmarking tool, "xsbench," may elucidate this endeavor Mas (2013); Scudiero (2014). Since global memory comes at a premium on GPUs, an on-the-fly temperature treatment for nuclides would likely be required if more than a handful of isotopes are desired at more than one temperature. Methods like those used in Serpent could be adapted for use on the GPU ser (2014). On-the-fly methods reduce the amount of storage needed, but they require more computation per data element loaded since the loaded value is adjusted according to the temperature of the material. This kind of method may work well on the GPU since GPUs have a larger FLOP/byte ratio than CPUs and the additional work may cost little. A grid thinning routine that discards closely-spaced energy grid points could also be added to WARP, saving memory but potentially costing accuracy. Fractional cascading may also help reduce the memory footprint of the cross section data while keeping energy search cost low Lund et al. (2015).

WARP would gain usability if more features were incorporated as well. Importance cutoffs could be used to terminate neutrons, leading to shorter runtimes; cell importances (easily attached to cell primitives), track length tallies, and implicit absorption could help improve tally statistics. Developing an efficient way to include many reaction rate tallies would also make WARP useful for performing depletion analysis. It would further be helpful for WARP to have statistical tests like Shannon entropy to ensure the fission source is fully converged before tallies and multiplication factors are accumulated Ueki and Brown (2005). Accuracy would also be improved by adding $S(\alpha,\beta)$ thermal scattering tables and unresolved resonance parameters. Neutrons have statistical weight in WARP, so simple variance reduction techniques like implicit capture should be a straight-forward part of future development. Path lengths are already calculated in OptiX, so implementing track length tallies could also be a straight-forward variance reduction technique to introduce into WARP. Multi-GPU support should also be added so that WARP can be used effectively on computers with more than one GPU.

In conclusion, the initial development of WARP has been completed. It is the first code able to perform Monte Carlo neutron transport calculations in general 3D geometries entirely on GPUs using continuous energy, ACE-formatted cross sections. Using high-end consumer graphics cards, WARP can perform these calculations in very cost-effective way in a timeframe comparable to using multiple supercomputer nodes. The results calculated by WARP compare well with production codes in many cases, and are at least accurate enough to provide a realistic testbed for further GPU development in the field of Monte Carlo neutron transport, or provide fast precalculations for

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027

more accurate production code runs.

WARP will be released as open source software, pending DOE approval, at http://github.com/weft. In a weave of cloth, The "warp" is the high-tension, longitudinal thread, whereas the "weft" is the low tension, transverse thread. On a loom, the warp threads are all moved in sync, then the weft is passed through them. Naming the repository for the single thread that defines the phase of the warp seemed appropriate.

Acknowledgements

This research is based upon work partially supported by the U.S. Department of Energy National Nuclear Security Administration under Award Number DE-NA0000979 through the Nuclear Science and Security Consortium: http://nssc.berkeley.edu, as well as upon work supported under an Integrated University Program Graduate Fellowship. This research also used the Savio computational cluster resource provided by the Berkeley Research Computing program at the University of California, Berkeley (supported by the UC Berkeley Chancellor, Vice Chancellor of Research, and Office of the CIO).

Disclaimer

References

The amd opteron processor is the ideal solution for high performance computing. Technical report, Advanced Micro Devices, Inc., 2011. URL http://sites.amd.com/us/Documents/AMD_Opteron_ideal_for_HPC.pdf.

Serpent website. http://montecarlo.vtt.fi/, April 28 2014.

RSICC website. https://rsicc.ornl.gov/codes/dlc/dlc2/dlc-249.html, June 2016.

List of Nvidia Graphics Processing Units. https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units, accessed 01/2017.

Google Test. https://github.com/google/googletest, accessed 10/2016.

LAMMPS Benchmarks. http://lammps.sandia.gov/bench.html, accessed 10/2016a.

Roy Longbottoms PC Benchmarks Colleciton, Linux PC benchmarks. http://www.roylongbottom.org.uk/linux%20benchmarks.htm, accessed 10/2016b.

OECD Nuclear Energy Agency. *International Handbook of Evaluated Criticality Safety Benchmark Experiments*. Nuclear Energy Agency, OECD, 1995.

D. M. Beazley. Automated Scientific Software Scripting with SWIG. *Future Gener. Comput. Syst.*, 19(5):599–609, July 2003. ISSN 0167-739X. doi: 10.1016/S0167-739X(02)00171-1. URL http://dx.doi.org/10.1016/S0167-739X(02)00171-1.

Ryan M Bergmann and Jasmina L Vujić. 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.

Forrest B. Brown and William R. Martin. Monte Carlo methods for radiation transport analysis on vector computers. *Progress in Nuclear Energy*, 14(3):269 – 299, 1984. ISSN 0149-1970. doi: http://dx.doi.org/10.1016/0149-1970(84)90024-6. URL http://www.sciencedirect.com/science/article/pii/0149197084900246.

Alfredo Buttari, Jack J. Dongarra, Jakub Kurzak, Piotr Luszczek, and Stanimire Tomov. Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy. *ACM Trans. Math. Softw.*, 34, 2008.

Brian Catanzaro. An Introduction to CUDA/OpenCL and Manycore Graphics Processors. Lecture slides, Feb 2011. University of California, Berkeley CS267 - Applications of Parallel Computers.

M.B. Chadwick, M. Herman, P. ObloÅinskÅ, M.E. Dunn, Y. Danon, A.C. Kahler, D.L. Smith, B. Pritychenko, G. Arbanas, R. Arcilla, R. Brewer, D.A. Brown, R. Capote, A.D. Carlson, Y.S. Cho, H. Derrien, K. Guber, G.M. Hale, S. Hoblit, S. Holloway, T.D. Johnson, T. Kawano, B.C. Kiedrowski, H. Kim, S. Kunieda, N.M. Larson, L. Leal, J.P. Lestone, R.C. Little, E.A. McCutchan, R.E. MacFarlane, M. MacInnes, C.M. Mattoon, R.D. McKnight, S.F. Mughabghab, G.P.A. Nobre, G. Palmiotti, A. Palumbo, M.T. Pigni, V.G. Pronyaev, R.O. Sayer, A.A. Sonzogni, N.C. Summers, P. Talou, I.J. Thompson, A. Trkov, R.L. Vogt, S.C. van der Marck, A. Wallner, M.C. White, D. Wiarda, and P.G. Young. ENDF/B-VII.1 Nuclear Data for Science and Technology: Cross Sections, Covariances, Fission Product Yields and Decay Data. *Nuclear Data*

Sheets, 112(12):2887 – 2996, 2011. ISSN 0090-3752. doi: http://dx.doi.org/10.1016/j.nds.2011.11.002. URL http://www.sciencedirect.com/science/article/pii/S009037521100113X.

Y.K. Chan. Benchmarks for Intel MIC Architecture. http://www.clustertech.com/wp-content/uploads/2014/01/MICBenchmark.pdf, 2013.

James J. Duderstadt and Louis J. Hamilton. *Nuclear Reactor Analysis*. John Wiley & Sons, Inc., New York, NY, 1976.

Christiaan Gribble and Lee A. Butler. Advances in High-Performance GPU Ray Tracing for Physics-Based Simulation. San Jose, CA, March 2013. GPU Technology Conference.

Mark Harris, Shubhabrata Sengupta, and John D. Owens. Parallel Prefix Sum (Scan) with CUDA. In Hubert Nguyen, editor, *GPU Gems 3*, pages 851 – 876. Addison Wesley, 2007. URL http://www.idav.ucdavis.edu/publications/print_pub?pub_id=916.

Nicholas Henderson, K. Murakami, K. Amako, M. Asai, T. Aso, A. Dotti, A. Kimura, M. Gerritsen, H. Kurashige, and T. Sasaki J. Perl. A CUDA Monte Carlo simulator for radiation therapy dosimetry based on Geant4. Paris, France, October 2013. Supercomputing in Nuclear Applications and Monte Carlo (SNA+MC).

LM Itu, C Suciu, F Moldoveanu, and A Postelnicu. Comparison of single and double floating point precision performance for Tesla architecture GPUs. *Bulletin of the Transilvania University of Br ov Series I: Engineering Sciences*, 4(53), 2011.

Los Alamos National Laboratory. http://t2.lanl.gov/nis/endf/, January 23 1998. An Introduction to the ENDF Formats.

Julie Langou, Julien Langou, Piotr Luszczek, Jakub Kurzak, Alfredo Buttari, and Jack Dongarra. Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems). In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM. ISBN 0-7695-2700-0. doi: 10.1145/1188455.1188573. URL http://doi.acm.org/10.1145/1188455.1188573.

Yunsup Lee, Rimas Avizienis, Alex Bishara, Richard Xia, Derek Lockhart, Christopher Batten, and Krste Asanović. Exploring the tradeoffs between programmability and efficiency in data-parallel accelerators. *SIGARCH Comput. Archit. News*, 39(3):129–140, June 2011. ISSN 0163-5964. doi: 10.1145/2024723.2000080. URL http://doi.acm.org/10.1145/2024723.2000080.

Jaakko Leppänen. Development of a New Monte Carlo Reactor Physics Code, 2007. D.Sc. Dissertation.

Jaakko Leppänen. Performance of Woodcock delta-tracking in lattice physics applications using the Serpent Monte Carlo reactor physics burnup calculation code. *Annals of Nuclear Energy*, 37(5):715–722, 2010. ISSN 0306-4549. doi: http://dx.doi.org/10.1016/j.anucene.2010.01.011. URL http://www.sciencedirect.com/science/article/pii/S0306454910000320.

Tianyu Liu, Aiping Ding, Wei Ji, X. George Xu, Christopher D. Carothers, and Forrest B. Brown. A Monte Carlo Neutron Transport Code For Eigenvalue Calculations on a Dual-GPU System and CUDA Environment. Knoxville, Tennessee, April 2012. PHYSOR.

Amanda L. Lund, Andrew R. Siegel, Benoit Forget, Colin Josey, and Paul K. Romano. Using fractional cascading to accelerate cross section lookups in Monte Carlo neutron transport calculations. In *Proceedings of Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA), and the Monte Carlo (MC) Method*, Nashville, Tennessee, April 19?23 2015.

OpenMC Documentation. Massachusetts Institute of Technology, Cambridge, MA, 0.5.3 edition, 2013. http://mit-crpg.github.io/openmc/index.html.

CUDA C Programming Guide. NVIDIA Co, Santa Clara, CA, pg-02829-001_v5.5 edition, 2013.

OptiX Programming Guide. NVIDIA Co, Santa Clara, CA, v3.0 edition, Nov. 2012a.

OptiX Programming Guide. NVIDIA Co, Santa Clara, CA, v3.5 edition, Nov. 2012b.

Denise B. Pelowitz, John T. Goorley, Michael R. James, Thomas E. Booth, Forrest B. Brown, Jeffrey S. Bull, Lawrence J. Cox, Joe W. Durkee, Jay S. Elson, Michael L. Fensin, R. Arthur Forster, John S. Hendricks, H. Grady Hughes, Russell C. Johns, Brian C. Kiedrowski, Roger L. Martz, Stepan G. Mashnik, Gregg W. McKinney, Richard E. Prael, Jeremy E. Sweezy, Laurie S. Waters, Trevor A. Wilcox, and Anthony Zukaitis. *MCNP6 USERS MANUAL VERSION 1.0*. Los Alamos National Laboratory,, Los Alamos, NM, rev. 0 edition, May 2013.

Python Language Reference. Python Software Foundation, 2.7 edition, 2016. http://www.python.org.

Joshua Ruggiero. Measuring cache and memory latency and cpu to memory bandwidth. Technical report, Intel Co., 2008.

Anthony Scopatz, Seth Johnson, Paul Romano, Paul Wilson, Nick Touran, Katy Huff, Chris Dembia, Eric Relson, Elliott Biondo, Matthew Gidden, Cameron Bates, and Kevin Manalo. PyNE: The Nuclear Engineering Toolkit. pyne.io, 2014.

Tony Scudiero. Monte Carlo Neutron Transport - Simulating Nuclear Reactions One Neutron at a Time. San Jose, California, March 2014. GPU Technology Conference 2014.

X-5 Monte Carlo Team. *MCNP - A General Monte Carlo N-Particle Transport Code, Version 5*. Los Alamos National Laboratory,, Los Alamos, NM, volume i: overview and theory edition, April 24 2003. (Revised 2/1/2008).

Taro Ueki and Forrest B. Brown. Stationarity Modeling and Informatics-Based Diagnostics in Monte Carlo Criticality Calculations. *Nuclear Science and Engineering*, 149:38–50, 2005.

S. van der Walt, S. C. Colbert, and G. Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, 13(2):22–30, March 2011. ISSN 1521-9615. doi: 10.1109/MCSE.2011.37.

Jasmina L. Vujic and William R. Martin. Vectorization and parallelization of a production reactor assembly code. *Progress in Nuclear Energy*, 26(3):147 – 162, 1991. ISSN 0149-1970. doi: http://dx.doi.org/10.1016/0149-1970(91)90033-L. URL http://www.sciencedirect.com/science/article/pii/014919709190033L.

Eddy Z. Zhang, Yunlian Jiang, Ziyu Guo, and Xipeng Shen. On-the-fly elimination of dynamic irregularities for GPU computing. *SIGPLAN Not.*, 46(3):369–38, 2011.

FINAL ARTICLE – doi:10.1016/j.anucene.2017.01.027