

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. Flux spectra, multiplication factors, runtimes, speedup factors, and costs of various GPU and CPU platforms running either WARP, Serpent 2.1.21, 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 about 0.8 to 7.6 times as fast as new CPU platforms while being about half as expensive.

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

*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)

1. Introduction

WARP, 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, flux spectra, and fission source distributions for time-independent neutron transport problems, and can run in both criticality or fixed source modes. Fixed source mode is currently not robust or optimized, however. WARP can transport neutrons in unrestricted arrangements of parallelepipeds, hexagonal prisms, cylinders, and spheres.

The goal of developing WARP was to be the first step in creating 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, or CPUs, GPUs have a larger aggregate memory bandwidth, much larger rate of floating-point operations per second (FLOPS), and lower energy consumption per FLOP. GPUs execute efficiently on data-parallel problems, and since most CPU codes are task-parallel, the algorithms used had to be reconsidered. Data-parallelism is simply parallelism that arises from operating on many different pieces of data at one time, whereas task-parallelism is parallelism that arises from running many concurrent tasks that 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.

It seems that GPU cards would be perfect for running Monte Carlo neutron transport because they can run large numbers of concurrent threads. The concurrent thread number is based on the width of the processor’s SIMD (single instruction multiple data) units, however. SIMD is an execution model some processors use in order to lower the number of instructions needed per amount of computation done, which increases both power and computational efficiency [1]. SIMD requires the same instructions to be carried out over every element in a concurrently-processed data vector. From a thread standpoint, SIMD requires threads to execute the same instruction

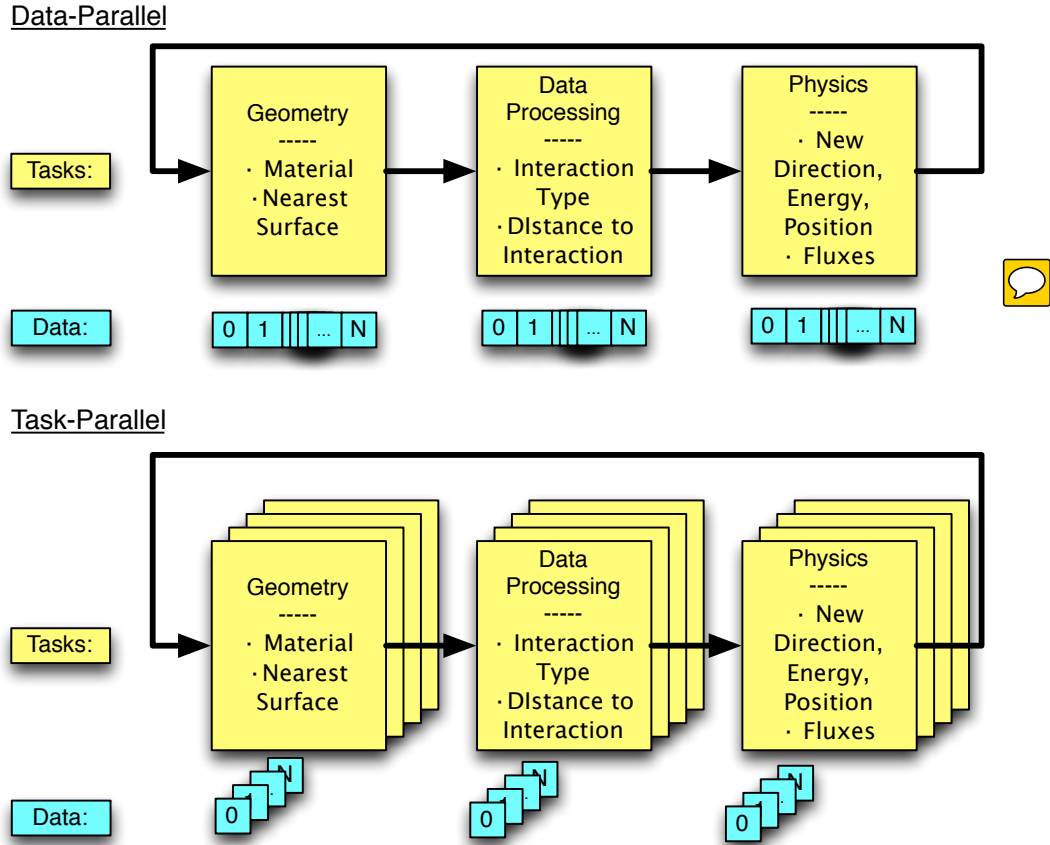




Figure 1: Data-parallel neutron transport loop vs. a task-parallel transport loop for transporting N neutrons in parallel.

at a point in time. If a set of threads does not execute the same instruction at the same time (the data they act on can still be different), the GPU will serialize them. The subset of threads executing the first instruction will all execute together, then the subset executing the second instruction will execute after them. 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.

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

48 of data must be loaded and used per transaction, and the only way multiple
 49 pieces of data can be loaded simultaneously is if they are adjacent in memory.
 50 In other words, if a program requests a single piece of data at location i ,
 51 then requests a piece at location $i + 10$, these requests will be split into
 52 two separate transactions that yield only one data element each. Two data
 53 elements in two transactions produces an effective bandwidth of one element
 54 per transaction time. On the other hand, if the program requests data at i ,
 55 $i + 1$, $i + 2$, and $i + 3$, the entirety of the requested data can be retrieved
 56 in a single transaction. Four data elements in a single transaction yields
 57 and effective memory bandwidth of four elements per transaction time, four
 58 times higher than the previous scenario. Having a memory subsystem that
 59 handles requests in this way is not ideal for Monte Carlo methods, however.
 60 Data is accessed in a very random way because of the random nature of the
 61 simulation, and requested data is unlikely to be adjacent. This means the
 62 full memory bandwidth of the GPU will not be used unless this problem is
 63 mitigated in some way.

64 Another undesirable feature of the GPU  is that they have very high global
 65 memory latency compared to a CPU. Memory latency is the number of clock
 66 cycles, or amount of time, it takes for a data request to be fulfilled. As Table
 67 1 shows, the GPU's global memory latency is about an order of magnitude
 68 higher than the CPU's [2, 3]. GPUs try to eliminate the effect of large global
 69 latency by pipelining memory access. Pipelining means threads that have
 70 received their data can execute as other threads are waiting for their data
 71 to load. If many requests are known, the data can be continually loaded as
 72 threads start to execute their jobs. The hope is that the jobs take longer than
 73 the memory loads, eventually all data arrives, and the later threads appear
 74 to have zero latency for their memory access. This is why it is important for
 75 GPUs to have such a large number of concurrent threads. It allows them to
 76 pipeline data access and minimize the impact of memory latency.

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


83 The impetus for developing WARP was the research done by Martin and
 84 Brown in 1984 [6] for Monte Carlo and by Vujić and Martin in 1991 [7] for
 85 collision probability method. In the 1984 paper, a method for mapping the

Table 1: NEEDS UPDATING. A comparison of an NVIDIA GPU and an Intel CPU [4, 2, 5].

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	3.46GHz	1.15 Ghz
Resident Strands / Threads (max)	48	21,504
SP GFLOP/s	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	6kB	2MB
Local Storage / L1 Cache	192 kB	896 kB
L2 Cache	1536 kB	0.75 MB
L3 Cache	12 MB	-

Monte Carlo problem onto SIMD (single instruction multiple data) 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 was placed in a buffer containing the data of other neutrons which have also been sampled to undergo scattering reactions. If a neutron needs to do a surface crossing, 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 makes the buffers 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.

105 Instead of trying to port an existing code to the GPU, the simplest way to
106 accommodate all the requirements for efficient GPU execution was to write
107 a new code from scratch. This project ultimately resulted in WARP. WARP
108 uses an event-based algorithm, but with some important differences. Vector-
109 izing the Monte Carlo algorithm should allow for efficient GPU execution,
110 but a paper by Zhang [8] also shows that by remapping data references, the
111 thread divergence in GPU warps can be minimized. Moving data is expen-
112 sive, especially when the data set is large, so WARP uses a remapping vector
113 of pointer/index pairs to direct GPU threads to the data they need to ac-
114 cess. The remapping vector is sorted by reaction type after every transport
115 iteration using a high-efficiency parallel radix sort (via the CUDPP library
116 [9]), which serves to keep the reaction types as contiguous as possible and
117 removes completed histories from the transport cycle. Sorting reduces the
118 amount of divergence in GPU “thread blocks,” keeps the SIMD units as full
119 as possible, and eliminates using memory bandwidth to check if a neutron in
120 the batch has been terminated or not. Using a remapping vector means the
121 data access pattern is irregular, but this is mitigated by using large batch
122 sizes where the GPU can effectively reduce the high cost of irregular global
123 memory access. The details about the existing algorithms used in WARP are
124 discussed in [10] and will not be discussed in detail here. New algorithmic
125 changes s are discussed in Section 2.

126 To use NVIDIA GPUs, parts of software must be written in a language
127 that facilitates interaction with the GPU, and CUDA [3], a set of extensions
128 for C/C++, was chosen. CUDA was first released in 2006 as NVIDIA’s
129 proprietary GPU programming platform. It makes minimal additions to
130 C/C++, and any C programmer would be very comfortable programming in
131 CUDA [3]. It was chosen over OpenCL, the open source GPU programming
132 platform, due to CUDA’s greater feature support, stability, ease of program-
133 ming, wider community usage, and ability to use new, cutting-edge features
134 in NVIDIA GPUs that OpenCL is not.

135 In this paper, the current features of WARP are summarized while high-
136 lighting the newest developments. The results of criticality calculations per-
137 formed by WARP are compared against those from Serpent 2.1.21 [11, 12]
138 and MCNP 6.1 [13], two widely-used production-level Monte Carlo neutron
139 transport codes. This comparison is done in order to ensure the accuracy of
140 WARP and to highlight its performance differences. The test case models
141 used to conduct the comparisons are outlined in detail, then the run times,
142 multiplication factors, and flux spectra calculated for the test cases are com-

pared in detail, after which a discussion is given summarizing comparisons and addressing any discrepancies shown. Lastly, conclusions about the initial development of WARP are drawn and future work is roughly outlined.

2. Features of WARP

WARP has only existed since 2013, and is not as fully-featured as more mature Monte Carlo codes. However, it has the functionality necessary to compare its performance against Serpent 2.1.21 and MCNP 6.1, and 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. Pythonic wrapping of the shared library is done via SWIG [14], 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. 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.

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 if necessary. 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.

2.1. 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. WARP loads ACE-formatted nuclear data libraries via the “ace” module in PyNE (Python for Nuclear Engineering) [15]. This module has been separated 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 (reducing “dependency hell”). 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 [16]. 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 re-formatted to use a unionized energy grid via NumPy [17], the re-formatted data are then passed to the C++ routines via the Python C API.

In its current state, WARP does not use $S(\alpha, \beta)$ 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) [18]. These laws cover all the reactions present in the test problems presented here and most nuclides in the ENDFB/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), 67 (laboratory angle-energy law). Problems containing any such nuclides cannot be simulated with the current version of WARP, but of course could be added later. These laws are not nearly as common as the ones currently included in WARP, and excluding them at this point should not greatly reduce the ability to benchmark

216 WARP's performance and accuracy.

217 WARP only has collision estimators implemented for both flux and mul-
218 tiplication factor estimation. The relative error of these estimations are cal-
219 culated, but any additional convergence estimation beyond relative error has
220 not been implemented. Variance reduction is not implemented in WARP.
221 Neutrons have statistical weight in WARP, and currently the only reactions
222 that adjust neutron weight are multiplicity reactions, like $(n,2n)$, where the
223 neutron weight is multiplied by the exiting neutron number in a manner
224 similar to OpenMC [19]. This means that new neutron histories do not have
225 to be created mid-batch in criticality calculations, and makes program flow
226 simpler. Analog capture was turned on in Serpent and MCNP in all the
227 simulations considered to ensure a fair comparison of the results.

228 2.2. Geometry

229 The geometry representation in WARP is handled by the NVIDIA OptiX
230 ray tracing framework [20]. OptiX provides good performance on the GPU,
231 but imposes some limitations on the how geometries must be input. For the
232 current algorithm used, the geometrical limitations are:

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

246 An improved tracking routine, which determines which cell a neutron
247 resides in based on its spatial coordinates, has been implemented in WARP
248 since the last publication [10]. Previously, a ray was traced from the neutron
249 position to the outer cell, and a list of cell numbers was stored, in order, for
250 each intersection. The double entries were removed as the list was written,

251 which at the end of the trace yielded a list of cells the neutron was nested
 252 in. This resulted in very poor performance in deeply-nested cases due to this
 253 list becoming large. For example, WARP had very significant performance
 254 decreases in the Jezebel spherical criticality test (discussed later) if the sphere
 255 was segmented into many spherical shells. Doing this made WARP need to
 256 store a long list of surface numbers and made the simulation very slow.

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

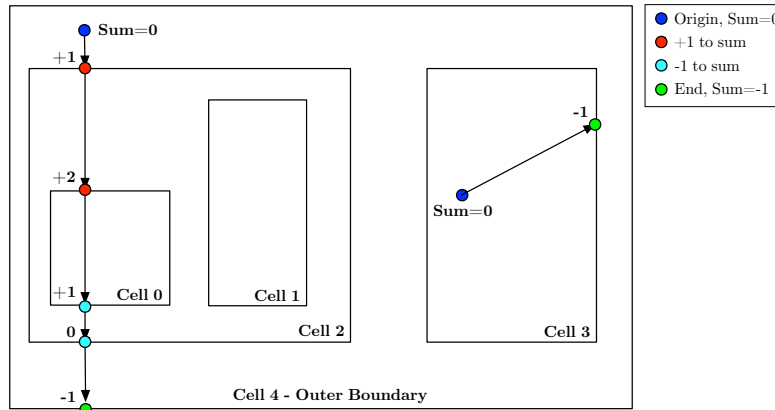



Figure 2: The improved geometry query algorithm showing how the cell number is calculated through ray tracing. A ray is iteratively drawn from the neutron starting position to the outer cell, and the surface normal values are summed along the way. When the sum reaches -1, the cell in which the neutron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

270 When the cumulative sense becomes negative, the containing cell has
 271 been intersected and tracing can stop. This approach can be used because

272 all bodies must be closed. This scheme takes advantage of the acceleration
273 structure in OptiX while still using the simplicity of combinatorial solid ge-
274 ometry. It essentially is the same scheme as MCNP/Serpent, except that the
275 acceleration structures allow surfaces that are part of cells that are far away
276 to be skipped. Also, compared to the past scheme of storing a list of doubles,
277 this method requires a single integer to be stored and operated on instead of
278 a potentially large and slow list.

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

3. 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 Berkeley, the departmental cluster, and Savio2, the shared campus high performance computing cluster, were used. For the GPU platforms, a consumer-level graphics card and two compute-specific cards were used. Descriptions of each piece of hardware can be found in Table 2. For the GPUs, the costs include the retail price of a minimal host computer with at least as much memory as the card (~\$400).



Table 2: Platforms used in the benchmark cases and their specifications.

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 nx360m5	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 nx360m5		2.3 GHz	24
NVIDIA Tesla K20	13	705.5 MHz	$2^{32} - 1$
NVIDIA Titan Black	15	1071.5 MHz	$2^{32} - 1$
NVIDIA Tesla K80	13	823.5 MHz	$2^{32} - 1$



All tests use ENDF/B-VII cross sections that are distributed with the Serpent 1.1.7 release from RSICC (Radiation Safety Information Computational Center), which regulates the licensing and distribution of the MCNP and ENDF data worldwide, and also distributes Serpent 1.1.7. These cross sections contain fewer energy grid points than the ENDF/B-VII.1 cross sections that are distributed with the MCNP 6.1 release from RSICC. Since the Serpent 1.1.7 data require less storage space, using them allows more isotopes to be used by WARP.



308 3.1. Test 1 - “Jezebel” Bare Pu Sphere

309 The “Jezebel” criticality test is a bare plutonium/gallium sphere with
 310 vacuum boundary conditions. The fission neutron rate from ^{239}Pu is bal-
 311 anced by the leakage rate from the 5.1 cm radius to give a k_{eff} of approx-
 312 imately 1. Since this system is so leaky, producing results consistent with
 313 MCNP and Serpent ensures that the boundary conditions are correctly be-
 314 ing enforced. The Jezebel test is a standard test used to validate neutron
 315 transport codes and is described in the International Handbook of Evaluated
 316 Criticality Safety Test Experiments under the name “Pu-MET-FAST-001”
 317 [21]. The geometry and materials are outlined in Table 3. All cross sections
 318 used were processed at 273.5 K.

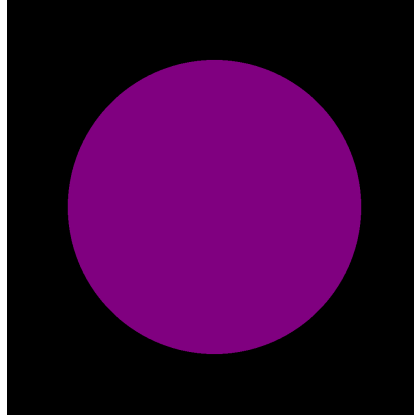


Figure 3: Horizontal slice of the geometry of the “Jezebel” test case. Colors represent cell numbers and black represents space outside the outermost boundary.

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)	

3.2. Test 2 - Homogenized Fuel Block

The homogenized block criticality test is a bare cube with vacuum boundary conditions. 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 outlines 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.

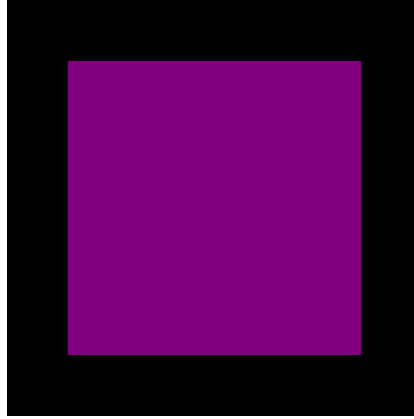


Figure 4: Horizontal slice of the geometry of the homogenized fuel block test case. Colors represent cell numbers and black represents space outside the outermost boundary.

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)	5.50 g/cm ³
	²³⁵ U	(0.10)	
	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)	

330 *3.3. Test 3 - Zr-Clad UO₂ Pin in Heavy Water*

331 This criticality test consists of a UO₂ cylinder clad in zirconium sur-
 332 rounded by a block of light water. This test increases complexity by having
 333 three materials, each with multiple isotopes, and three cells. The water block
 334 has vacuum boundary conditions. Its dimensions are relatively large and the
 335 absorption is low, so the mean neutron lifetime should be large. This test
 336 serves to highlight that all the processing routines work simultaneously, the
 337 effect of introducing more than one cell, and the effect of long-lived neutrons.
 338 The geometry and materials are outlined in Table 5. All cross sections used
 339 were also processed at 273.5 K.

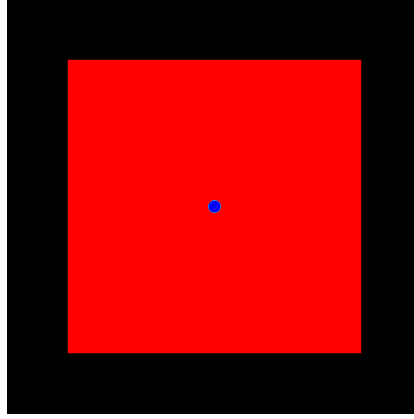


Figure 5: Horizontal slice of the geometry of the Zr-clad UO₂ pin in heavy water test case. Colors represent cell numbers and black represents space outside the outermost boundary.

Table 5: Geometry and materials used in the single UO₂ pin in H₂O test case.

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

340 3.4. Test 4 - Homogenized Fuel Pebble in FLiBe

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

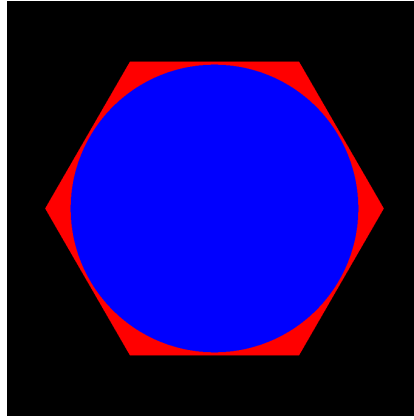


Figure 6: Horizontal slice of the geometry of the homogenized fuel pebble in FLiBe test case. Colors represent cell numbers and black represents space outside the outermost boundary.

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

348 *3.5. Test 5 - Stainless Steel Clad Metallic Uranium Pin in Liquid Sodium*

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

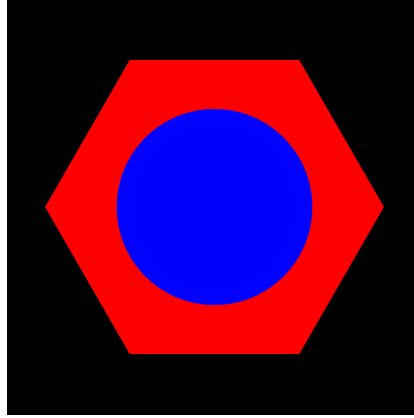


Figure 7: Horizontal slice of the geometry of the stainless steel clad metallic uranium pin in liquid sodium test case. Colors represent cell numbers and black represents space outside the outermost boundary.

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

357 *3.6. Test 6 - Zr-Clad Hexagonal UO₂ Pin Cell Lattice in Light Water*

358 This criticality test consists of 631 Zr-clad UO₂ cylinders laid out in a
 359 hexagonal lattice surrounded by light water. The material compositions,
 360 densities, and cylinder dimensions are similar to the pin cell test case, but,
 361 since this test has two orders of magnitude more objects, it serves to highlight
 362 the effect of introducing many geometric objects into the problem and further
 363 validates that the geometry processing routines work correctly. The lattice is
 364 in the x-y plane, has a pitch to diameter ratio of 1.164, and has 15 elements
 365 on a side. The geometry and materials are outlined in Table 8. All cross
 366 sections used were processed at 273.5 K.

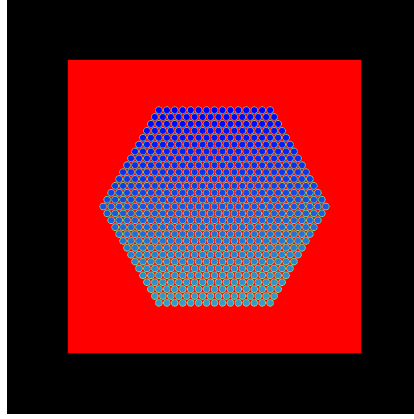


Figure 8: Horizontal slice of the geometry of the Zr-clad hexagonal UO₂ pin cell lattice in light water test case. Colors represent cell numbers and black represents space outside the outermost boundary.

Table 8: Geometry and materials used in the Zr-clad UO₂ pin hexagonal lattice in light water test case.

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

367 4. Results

368 The accuracy of the WARP calculations were measured by comparing the
369 multiplication factors and neutron spectra of the aforementioned geometries
370 against those calculated by Serpent 2.1.24 and MCNP 6.1. The multiplica-
371 tion factor differences shown in this section are reported in “per cent mille”
372 (PCM), which is a thousandth of a percent, or 10^{-5} . This is a standard way
373 of reporting differences in the multiplication factor, as any small change in
374 it can cause a significant change in system behavior. The flux spectra are
375 normalized per fission neutron and per unit lethargy. “Lethargy” means the
376 logarithm of the neutron energy, $\ln(E_0/E)$, where E_0 is the highest energy
377 possible in a system and E is another energy [22]. Normalizing the flux per
378 unit lethargy in conjunction with plotting on a logarithmic scale yields a plot
379 where the area under the curve gives the fraction of neutrons flux within the
380 bounding energy range.



381 The performance of WARP is measured by comparing the run times of
382 the WARP simulations against those of the production codes. The run times
383 do not include the time it takes to load and process cross sections before the
384 simulations. In other words, it is only the time it takes to complete all the
385 transport cycles. The difference in runtimes are reported in speedup factors
386 (t/t_{WARP}). The times reported are either for simulations performed on a
387 single card or a single node. This was done to compare the platforms fairly.
388 Resources are shared at the single card/node level, and therefore increasing
389 process and/or thread count does not necessarily scale linearly as resource
390 contention increases. It can be optimistically assumed that scaling occurs
391 linearly above this level if little inter-card/node communication is needed,
392 and therefore speedup ratios comparing single-node/card performance should
393 be equivalent to multi-node/card performance.

394 The benchmark cases were run with 6.5×10^6 neutrons per criticality
395 batch with 20 initial batches discard and 40 batches with statistic accumu-
396 lated (60 batches total). From the scaling results shown next in this section,
397 the neutron processing rate of the GPU cards all saturated near this number
398 of neutrons per batch. The CPU codes were run with identical batch param-
399 eters. The CPU cases were run with the optimal number of MPI processes
400 and threads per process for the specific node to fairly measure the perfor-
401 mance of the unit as a whole. Since the PowerWulf PSSC node contains 4
402 processors that each contain 2 non-uniform memory access (NUMA) nodes,
403 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 launches 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.

4.1. 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.

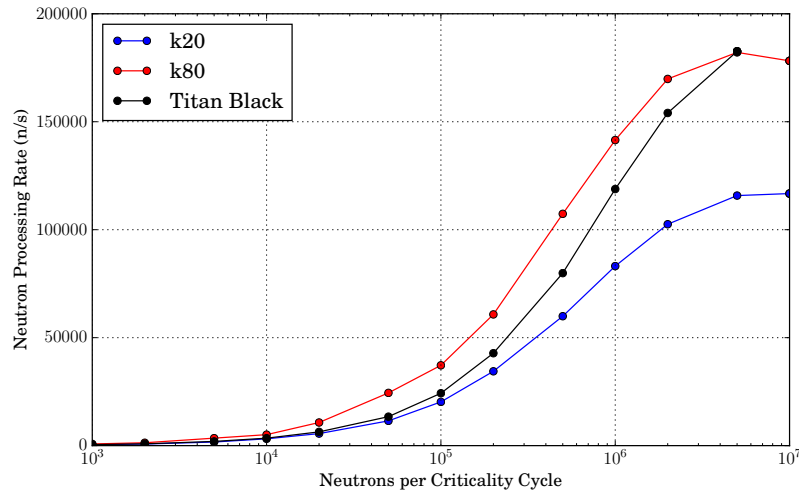


Figure 9: Neutron processing rate vs. number of neutrons per cycle for the hexagonal lattice test case.

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 a rough estimate of where maximum performance

occurs for WARP, and 6.5×10^6 neutrons per cycle was used for the test cases.

4.2. Multiplication Factors

Table 9 shows the multiplication factor deviations for the six criticality tests compared to MCNP 6.1 and Serpent 2.1.21 for WARP running on an NVIDIA Titan Black, K20, and K80 cards. The “ Δ MCNP” row contains WARP’s difference (in PCM) from MCNP, and the “ Δ Serpent” row contains WARP’s difference from Serpent. These difference rows also contain 1σ and 2σ values that indicate the level at which the multiplication factors calculated by WARP agree with either Serpent or MCNP. A “y” in the 1σ position indicates that the multiplication factor values are mutually within 1 standard deviation of each other, i.e. that $\Delta\text{MCNP} < \sigma_{\text{MCNP}} + \sigma_{\text{WARP}}$. This is the 90% confidence interval and there is a 10% probability that the the two codes could produce results outside of this interval. The 2σ position is simply double the 1σ width and corresponds to the 99.8% confidence interval (0.2% chance of being outside).

Table 9: The test case multiplication factors and runtimes for WARP using a Titan Black GPU versus those of Serpent 2.1.21 and MCNP 6.1.

	Jezebel	Homogenized Block	Pin Cell
Serpent 2.1.21	$0.9997840 \pm 9.50\text{E-}5$	$0.5934090 \pm 1.20\text{E-}4$	$0.2750510 \pm 1.80\text{E-}4$
MCNP 6.1	$0.9999600 \pm 7.00\text{E-}5$	$0.5933000 \pm 4.00\text{E-}5$	$0.2751700 \pm 7.00\text{E-}5$
WARP Titan Black	$1.0000620 \pm 9.58\text{E-}5$	$0.5933724 \pm 1.24\text{E-}4$	$0.2750294 \pm 1.46\text{E-}4$
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	$1.0000212 \pm 7.82\text{E-}5$	$0.5934539 \pm 1.66\text{E-}4$	$0.2751048 \pm 2.07\text{E-}4$
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	$0.9999602 \pm 9.58\text{E-}5$	$0.5932266 \pm 1.24\text{E-}4$	$0.2749884 \pm 1.56\text{E-}4$
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 2.1.21	$0.8804900 \pm 8.70\text{E-}5$	$1.0987100 \pm 2.20\text{E-}4$	$1.0503300 \pm 7.20\text{E-}5$
MCNP 6.1	$0.8805100 \pm 6.00\text{E-}5$	$1.0987700 \pm 6.00\text{E-}5$	$1.0506500 \pm 9.00\text{E-}5$
WARP Titan Black	$0.8807005 \pm 9.58\text{E-}5$	$1.0986860 \pm 7.82\text{E-}5$	$1.0510795 \pm 9.58\text{E-}5$
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	$0.8807086 \pm 9.58\text{E-}5$	$1.0987912 \pm 5.53\text{E-}5$	$1.0510620 \pm 9.58\text{E-}5$
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	$0.8807546 \pm 9.58\text{E-}5$	$1.0985898 \pm 5.53\text{E-}5$	$1.0511035 \pm 1.24\text{E-}4$
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

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

4.3. Runtimes

Table 10 shows the transport cycle runtimes for the six criticality tests compared to MCNP 6.1 and Serpent 2.1.21 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.

Table 10: The test case transport cycle run times (in minutes) for WARP, Serpent 2.1.21, 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 PSSC node and a Savio2 node.

		Jezebel	Homogenized Block	Pin Cell
Serpent 2.1.21	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 2.1.21	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

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

459 for the Jezebel test, the largest speedup over MCNP is for the homogenized
 460 block test, and the smallest speedup is for the sodium pin cell test over both
 461 Serpent and MCNP. This speedup factor is calculated over an entire compute
 462 node, so WARP can have performance equivalent to 0.84 to 7.61 compute
 463 nodes, depending on the type of simulation hardware considered. For the
 464 sodium pin case, where there is relatively a lot of cross section processing,
 465 a Savio2 node running Serpent actually performs better than WARP on a
 466 K80 card. For problems where there is less cross section processing (higher
 467 geometry processing fraction), WARP tend to perform better than a Savio2
 468 node running Serpent. WARP always performs better than a Savio2 node
 469 running MCNP, even with relatively heavy cross section processing, but this
 470 isn't a great surprise considering that MCNP doesn't use a unionized energy
 471 grid structure when processing cross sections.

472 4.4. Spectra



473 In the following subsections, spectra calculated by WARP are compared
 474 to those calculated by Serpent and MCNP for identical geometries and ma-
 475 terials. Every spectrum is binned into 1024 equi-log bins from 10^{-11} to 20
 476 MeV. In tests where there is a large region of little to no flux (like in the fast
 477 spectrum inside the Jezebel sphere), the energy bin structure is not changed,
 478 but the energy range of the plot is limited to the region where there is nonzero
 479 flux. The relative difference compared to the MCNP spectrum is shown in
 480 the top subplot below the main spectrum plot, and the relative difference
 481 from the Serpent spectrum is shown in the lower subplot. The green shaded
 482 area in the error subplots shows the space within 2 standard deviations of the
 483 statistical uncertainty of the production code. Unlike the intervals reported
 484 previous subsection, the interval shown in green is only for the production
 485 code results. In other words, the codes should have results within the 2σ
 486 interval 95.45% of the time (as opposed to 99.8% of the time).

487 4.4.1. Test 1 - “Jezebel” Bare Pu Sphere

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

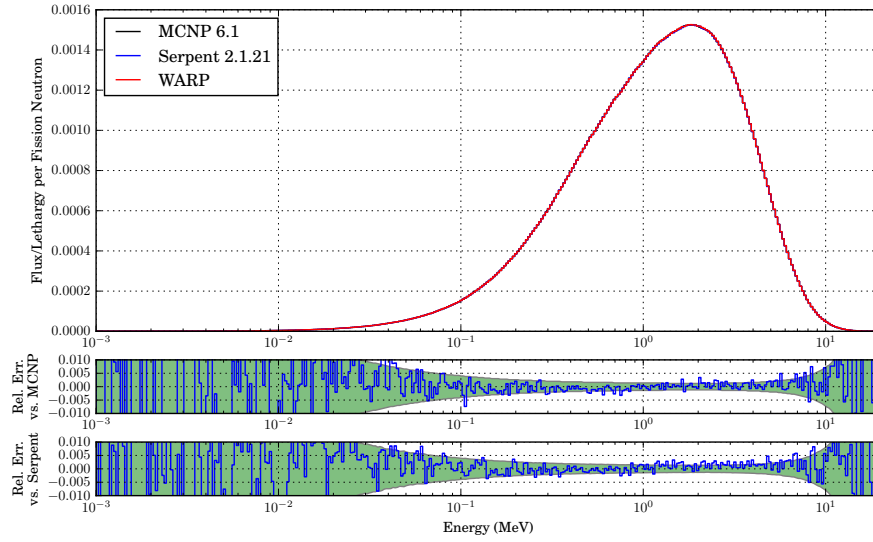


Figure 10: Volume-averaged flux spectra inside the sphere of the Jezebel test case.

4.4.2. Test 2 - Homogenized Fuel Block

Figure 11 shows the volume-averaged flux spectrum 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 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.

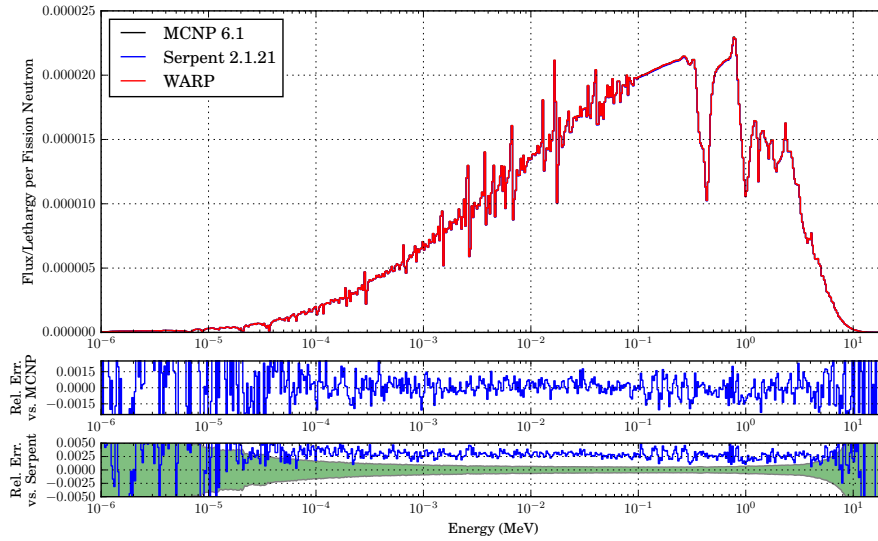


Figure 11: Volume-averaged flux spectra inside the homogenized fuel block.

507 4.4.3. Test 3 - Zr-Clad UO_2 Pin in Heavy Water

508 Figure 12 shows the volume-averaged flux spectrum inside the fuel rod in
 509 the Zr-clad UO_2 pin in heavy water test case. Again, the relative difference
 510 is generally less than 1% where the flux is large. Compared to Serpent,
 511 the WARP spectrum again appears to have a small constant offset. The
 512 entire energy range agrees well with MCNP, however, indicating that all the
 513 reaction laws were processed **indendctically** to MCNP for the nuclides present
 514 in this test case.

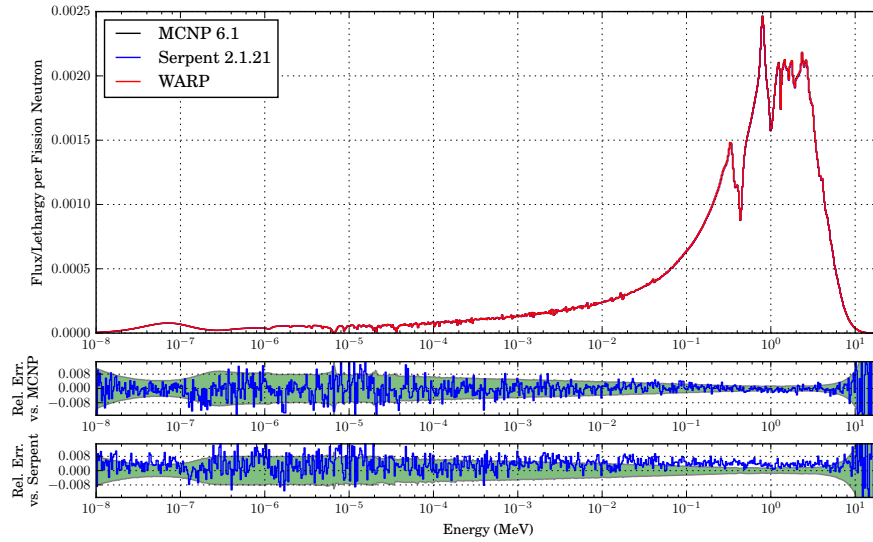


Figure 12: Volume-averaged flux spectra inside the oxide fuel of the Zr-clad pin in heavy test case

515 4.4.4. Test 4 - Homogenized Fuel Pebble in FLiBe

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

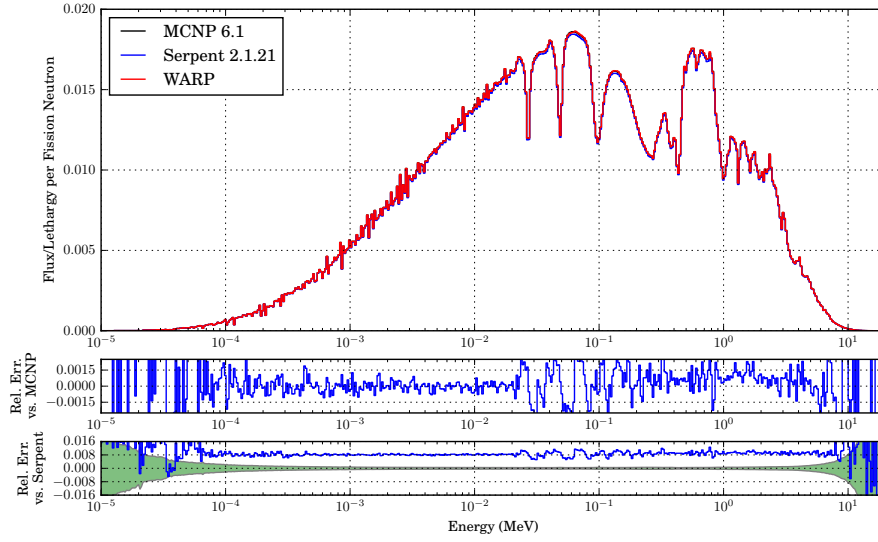


Figure 13: Volume-averaged flux spectra inside the fuel pebble of the reflective FLiBe test case.

522 4.4.5. Test 5 - Stainless Steel Clad Metallic Uranium Pin in Liquid Sodium

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

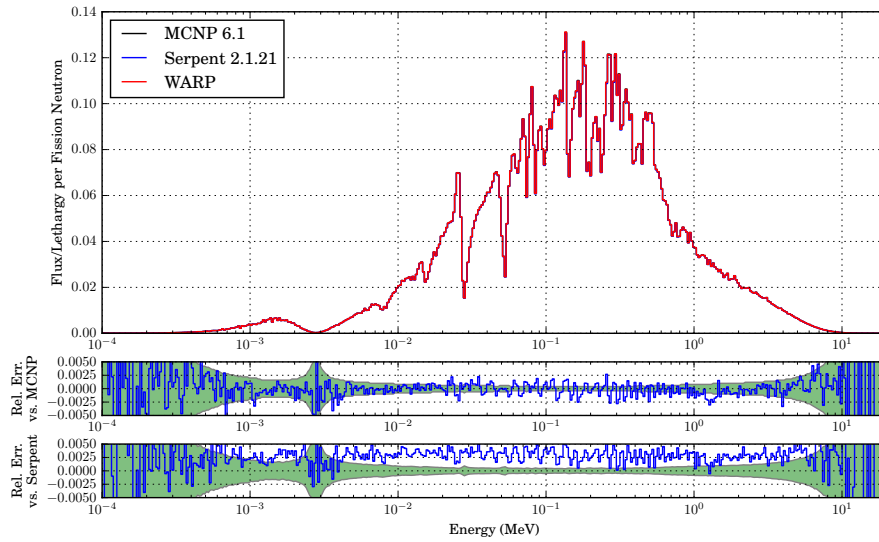


Figure 14: Volume-averaged flux spectra inside the fuel of the reflective, steel clad, sodium cooled pin test case.

530 4.4.6. Test 6 - Zr-Clad Hexagonal UO_2 Pin Cell Lattice in Light Water

531 Figure 15 shows the volume-averaged flux spectrum in the center pin of
 532 the Zr-clad hexagonal oxide fuel pin lattice in light water test case. The
 533 relative difference is generally less than 5% where the flux is large, but this
 534 time the WARP results are always within the statistical error of both MCNP
 535 and Serpent. There is no constant offset compared to Serpent or MCNP
 536 and no large deviations, indicating that all the reaction laws were processed
 537 indenctically to both Serpent and MCNP for the nuclides present in this test
 538 case. This is also the only test case other than Jezebel where MCNP and
 539 Serpent give identical spectra.

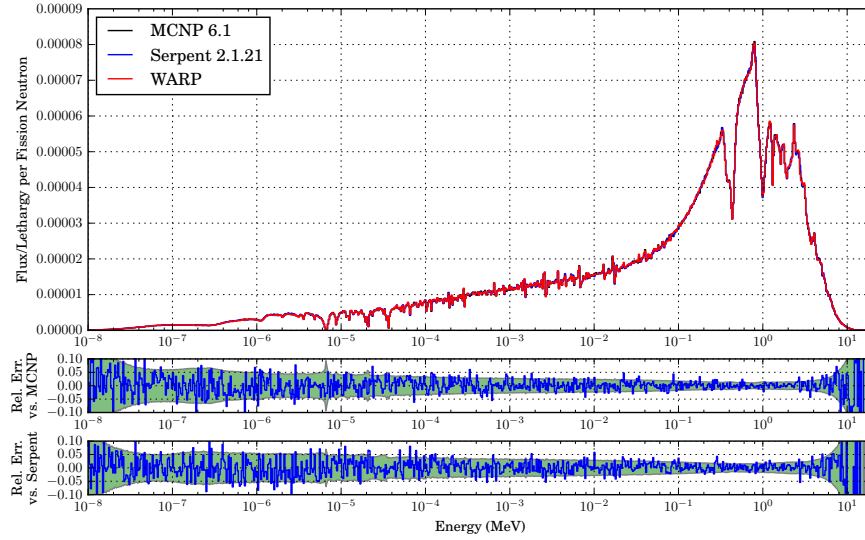


Figure 15: Volume-averaged flux spectra inside the center fuel pin of the hexagonal pin lattice test case.

540 4.5. History Power

541 For productivity, the “history power,” or number of neutron histories
 542 processed per unit time, is of primary interest when purchasing a system
 543 since it determines how quickly results can be produced. In addition to this,
 544 the history power per hardware cost is of interest for initial capital investment,
 545 and the history power per unit energy is of interest for operational costs.
 546 Table 11 shows the history power and relative costs of GPU systems running
 547 WARP compared to MCNP and Serpent running on a the aforementioned
 548 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	H.P. / Watt
Serpent 2.1.21	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



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

559 Table 11 shows that, on average, for the test cases considered, a new
 560 NVIDIA GPU card can process neutrons about 1.6 times as fast as a new
 561 multiprocessor/multicore CPU node running Serpent and about 3 times as
 562 fast as a CPU node running MCNP. From a capital investment standpoint,
 563 the a K80 card costs about half as much as a Savio2 node per neutron
 564 processing power and consumes about the same amount of electricity. The
 565 Titan Black, however, performs about as well as a K80 from the neutron

566 processing rate standpoint, but costs about a third as much and consumes
567 20% less electricity. Since WARP uses single-precision data and math in
568 order to fully exploit the capabilities available on consumer GPUs (like the
569 Titan Black), the capital and electrical costs of running WARP can be greatly
570 reduced.

571 5. Discussion

572 The differences seen in the multiplication factors and  spectra calculated
573 by WARP compared to those calculated by Serpent and MCNP are most
574 likely due to either a reaction law being inexactly processed or reactions
575 types being incorrectly sampled slightly. Whether this is due to a bug or if
576 it has to do with single precision math has not been determined. Some laws
577 potentially require division by small numbers then  comparing or adding the
578 result to other numbers. Roundoff error from single precision math could
579 make these operations inaccurate and ultimately lead to visible differences
580 in the neutron spectrum and multiplication factor. 

581 Monte Carlo is a great place  take advantage of single precision math
582 since solution convergence does not normally depend on derivatives, except
583 perhaps in a cross section processing law, and the calculated results are typi-
584 cally not required to be more precise than single-precision numbers. The ma-
585 jor places roundoff error can affect the solution is in tally accumulation and
586 cross section processing. Currently, WARP calculates the total macroscopic
587 cross section of a material on the fly, and roundoff could become a prob-
588 lem around resonances where the cross section of one nuclide (or several) is
589 much larger than the that of the other nuclides in the material. In this case,
590 the additions from the minor cross section nuclides could be neglected and
591 could lead to oversampling their reactions in the subsequent routines (since 
592 the material's calculated total macroscopic cross section would be smaller
593 than it should be). This problem could be circumvented by preprocessing
594 the material's total macroscopic cross section using double precision math
595 on the host before the data is given to the GPU as single precision numbers.
596 This, however, would increase the memory footprint of the data.

597 Another place roundoff error could cause problems is in tally accumula-
598 tion. WARP calculates the flux in a cell by calculating the inverse of the
599 macroscopic cross section at a collision site and adding it to the sum of all
600 previous collisions. If a large resonance lies within an energy bin, very dissim-
601 ilar values could be added together and precision could be lost. This could

602 be circumvented by using double precision data for the tally arrays. The cost
603 of doing so would have to be measured, but hopefully it would be small since
604 tally accumulation only happens once per transport cycle and is therefore a
605 relatively infrequent operation.

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

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

627 6. Conclusions and Future Development


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



637 scenarios where high computational throughput and/or low cost calculations
638 are of importance.

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

650 Geometry is currently handled by NVIDIA OptiX, which provides a con-
651 venient way to obtain high-performance results, but OptiX had to be coerced
652 into providing WARP with the extra information it needed beyond the dis-
653 tance to the nearest intersection, namely the material and tally **indices**.
654 The way OptiX is used to determine these numbers is not efficient since
655 it must be done iteratively using OptiX's native functions instead of calcu-
656 lating the sum of surface normals in a single trace. NVIDIA has released
657 "OptiX Prime" with OptiX 3.5 [23], which promises to provide a more "to
658 the metal" ray tracing experience, and might be leveraged to provide more
659 efficient single-traverse functionality. OptiX could also be replaced by Ray-
660 force [24], a high-performance GPU ray tracing library developed by VSL
661 that has this functionality built-in and is currently available free of charge
662 for noncommercial use.

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

672 WARP also suffers from "tail effect," where performance drops off as the
673 number of remaining active neutrons in a batch decreases [10]. Developing
674 a method that keeps the active neutron number high would greatly improve



675 performance, but could become quite complicated as generations would start
676 to overlap and calculation of the multiplication factor and converging the
677 source distribution couldn't be done in between cycles as it is now.

678 If an entire overhaul of the WARP transport algorithm is feasible, using
679 a streaming multiprocessor (SM)-based algorithm might be investigated for
680 replacing the current global one. This type of algorithm would treat each
681 SM as an independent processor and would provide each a bank of neutrons
682 to transport, as is done by Liu and Henderson [25, 26]. This way, neutron
683 data could be stored in very fast shared memory, but using this memory
684 space would compete with storing geometric information there. Also, since
685 a smaller set of neutrons could be stored, the SMs would need to commu-
686 nicate to determine which of the next neutrons they would take out of the
687 global bank, or they would need to periodically rendezvous to shared source
688 information and ensure that the distributions they use are each converged.
689 This type of transport algorithm would also preclude using OptiX, since it
690 does not have SM-level functionality [20].

691 An efficient way to handle situations where there are many different ma-
692 terial and isotopes present needs to be explored. The work done by Scudiero
693 on porting OpenMC's macroscopic cross section processing benchmarking
694 tool, "xsbench," may elucidate this endeavor [19, 27]. Since global memory
695 comes at a premium on GPUs, an on-the-fly temperature treatment for nu-
696 clides would likely be required if more than a handful of isotopes are desired
697 at more than one temperature. Methods like those used in Serpent could be
698 adapted for use on the GPU [12]. On-the-fly methods reduce the amount of
699 storage needed, but they require more computation per data element loaded
700 since the loaded value is adjusted according to the temperature of the ma-
701 terial. This kind of method may work well on the GPU since GPUs have a
702 larger FLOP/byte ratio than CPUs and the additional work may cost little.
703 A grid thinning routine could also be added to WARP which discards closely-
704 spaced energy grid points, saving memory but potentially costing accuracy.
705 Fractional cascading may also help reduce the memory footprint of the cross
706 section data while keeping energy search cost low [28].

707 WARP would gain usability if more features were incorporated as well.
708 Importance cutoffs could be used to terminate neutrons, leading to shorter
709 runtimes; cell importances (easily attached to cell primitives), track length
710 tallies, and implicit absorption could help improve tally statistics. Developing
711 an efficient way to include many reaction rate tallies would also make WARP
712 useful for performing depletion analysis. It would also be helpful for WARP





713 to have statistical tests like Shannon entropy to ensure the fission source
714 is fully converged before tallies and multiplication factors are accumulated.
715 Accuracy would also be improved by adding $S(\alpha, \beta)$ thermal scattering tables
716 and unresolved resonance parameters. Neutrons have statistical weight in
717 WARP, so simple variance reduction techniques like implicit capture should
718 be a straight-forward part of future development. Path lengths are already
719 calculated in OptiX, so therefore implementing track length tallies could also
720 be a straight-forward variance reduction technique to introduce into WARP
721 in the future. Multi-GPU support should also be added so that WARP can
722 be used effectively on computers with more than one GPU.

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

729 Acknowledgements

730 This research is based upon work partially supported by the U.S. De-
731 partment of Energy National Nuclear Security Administration under Award
732 Number DENA0000979 through the Nuclear Science and Security Consor-
733 tium: <http://nssc.berkeley.edu>, as well as upon work supported under an
734 Integrated University Program Graduate Fellowship. This research also used
735 the Savio computational cluster resource provided by the Berkeley Research
736 Computing program at the University of California, Berkeley (supported by
737 the UC Berkeley Chancellor, Vice Chancellor of Research, and Office of the
738 CIO).

739 Disclaimer

740 This report was prepared as an account of work sponsored by an agency
741 of the United States Government. Neither the United States Government
742 nor any agency thereof, nor any of their employees, makes any warranty,
743 express or limited, or assumes any legal liability or responsibility for the
744 accuracy, completeness, or usefulness of any information, apparatus, product,
745 or process disclosed, or represents that its use would not infringe privately
746 owned rights. Reference herein to any specific commercial product, process,

747 or service by trade name, trademark, manufacturer, or otherwise does not
748 necessarily constitute or imply its endorsement, recommendation, or favoring
749 by the United States Government or any agency thereof. The views and
750 opinions of authors expressed herein do not necessarily state or reflect those
751 of the United States Government or any agency thereof.

752 References

- 753 [1] Y. Lee, R. Avizienis, A. Bishara, R. Xia, D. Lockhart, C. Batten,
754 K. Asanović, Exploring the tradeoffs between programmability and ef-
755 ficiency in data-parallel accelerators, SIGARCH Comput. Archit. News
756 39 (2011) 129–140.
- 757 [2] J. Ruggiero, Measuring Cache and Memory Latency and CPU to Mem-
758 ory Bandwidth, Technical Report, Intel Co., 2008.
- 759 [3] CUDA C Programming Guide, NVIDIA Co, Santa Clara, CA, pg-02829-
760 001_v5.5 edition, 2013.
- 761 [4] B. Catanzaro, An introduction to cuda/opencl and manycore graphics
762 processors, Lecture slides, 2011. University of California, Berkeley CS267
763 - Applications of Parallel Computers.
- 764 [5] The AMD Opteron processor is the ideal solution for High Performance
765 Computing, Technical Report, Advanced Micro Devices, Inc., 2011.
- 766 [6] F. B. Brown, W. R. Martin, Monte carlo methods for radiation transport
767 analysis on vector computers, Progress in Nuclear Energy 14 (1984) 269
768 – 299.
- 769 [7] J. L. Vujic, W. R. Martin, Vectorization and parallelization of a pro-
770 duction reactor assembly code, Progress in Nuclear Energy 26 (1991)
771 147 – 162.
- 772 [8] E. Z. Zhang, Y. Jiang, Z. Guo, X. Shen, On-the-fly elimination of
773 dynamic irregularities for gpu computing, SIGPLAN Not. 46 (2011)
774 369–38.
- 775 [9] M. Harris, S. Sengupta, J. D. Owens, Parallel prefix sum (scan) with
776 cuda, in: H. Nguyen (Ed.), GPU Gems 3, Addison Wesley, 2007, pp.
777 851 – 876.

- 778 [10] R. M. Bergmann, J. L. Vujić, Algorithmic choices in WARP—a frame-
779 work for continuous energy monte carlo neutron transport in general 3D
780 geometries on GPUs, *Annals of Nuclear Energy* 77 (2015) 176–193.
- 781 [11] J. Leppänen, Development of a new monte carlo reactor physics code,
782 2007. D.Sc. Dissertation.
- 783 [12] Serpent website, <http://montecarlo.vtt.fi/>, 2014.
- 784 [13] D. B. Pelowitz, J. T. Goorley, M. R. James, T. E. Booth, F. B. Brown,
785 J. S. Bull, L. J. Cox, J. W. Durkee, J. S. Elson, M. L. Fensin, R. A.
786 Forster, J. S. Hendricks, H. G. Hughes, R. C. Johns, B. C. Kiedrowski,
787 R. L. Martz, S. G. Mashnik, G. W. McKinney, R. E. Prael, J. E. Sweezy,
788 L. S. Waters, T. A. Wilcox, A. Zukaitis, MCNP6 USERS MANUAL
789 VERSION 1.0, Los Alamos National Laboratory,, Los Alamos, NM,
790 rev. 0 edition, 2013.
- 791 [14] D. M. Beazley, Automated Scientific Software Scripting with SWIG,
792 *Future Gener. Comput. Syst.* 19 (2003) 599–609.
- 793 [15] A. Scopatz, S. Johnson, P. Romano, P. Wilson, N. Touran, K. Huff,
794 C. Dembia, E. Relson, E. Biondo, M. Gidden, C. Bates, K. Manalo,
795 PyNE: The Nuclear Engineering Toolkit, 2014.
- 796 [16] L. A. N. Laboratory, <http://t2.lanl.gov/nis/endl/>, 1998. An Intro-
797 duction to the ENDF Formats.
- 798 [17] S. van der Walt, S. C. Colbert, G. Varoquaux, The numpy array: A
799 structure for efficient numerical computation, *Computing in Science*
800 *Engineering* 13 (2011) 22–30.
- 801 [18] X.-. M. C. Team, MCNP - A General Monte Carlo N-Particle Transport
802 Code, Version 5, Los Alamos National Laboratory,, Los Alamos, NM,
803 volume i: overview and theory edition, 2003. (Revised 2/1/2008).
- 804 [19] OpenMC Documentation, Cambridge, MA, 0.5.3 edition, 2013.
805 [Http://mit-crpg.github.io/openmc/index.html](http://mit-crpg.github.io/openmc/index.html).
- 806 [20] OptiX Programming Guide, NVIDIA Co, Santa Clara, CA, v3.0 edition,
807 Nov. 2012.

- 808 [21] O. N. E. Agency, International Handbook of Evaluated Criticality Safety
809 Benchmark Experiments, Nuclear Energy Agency, OECD, 1995.
- 810 [22] J. J. Duderstadt, L. J. Hamilton, Nuclear Reactor Analysis, John Wiley
811 & Sons, Inc., New York, NY, 1976.
- 812 [23] OptiX Programming Guide, NVIDIA Co, Santa Clara, CA, v3.5 edition,
813 Nov. 2012.
- 814 [24] C. Gribble, L. A. Butler, Advances in High-Performance GPU Ray
815 Tracing for Physics-Based Simulation, GPU Technology Conference,
816 San Jose, CA.
- 817 [25] T. Liu, A. Ding, W. Ji, X. G. Xu, C. D. Carothers, F. B. Brown, A
818 Monte Carlo Neutron Transport Code For Eigenvalue Calculations on
819 a Dual-GPU System and CUDA Environment, PHYSOR, Knoxville,
820 Tennessee.
- 821 [26] N. Henderson, K. Murakami, K. Amako, M. Asai, T. Aso, A. Dotti,
822 A. Kimura, M. Gerritsen, H. Kurashige, T. S. J. Perl, A CUDA Monte
823 Carlo simulator for radiation therapy dosimetry based on Geant4, Su-
824 percomputing in Nuclear Applications and Monte Carlo (SNA+MC),
825 Paris, France.
- 826 [27] T. Scudiero, Monte Carlo Neutron Transport - Simulating Nuclear Re-
827 actions One Neutron at a Time, GPU Technology Conference 2014, San
828 Jose, California.
- 829 [28] A. L. Lund, A. R. Siegel, B. Forget, C. Josey, P. K. Romano, Using
830 fractional cascading to accelerate cross section lookups in Monte Carlo
831 neutron transport calculations, in: Proceedings of Joint International
832 Conference on Mathematics and Computation (M&C), Supercomputing
833 in Nuclear Applications (SNA), and the Monte Carlo (MC) Method,
834 Nashville, Tennessee.