Using Numba for GPU acceleration of Neutron Beamline Digital Twins

Coleman J. Kendrick‡*, Jiao Y. Y. Lin‡, Garrett E. Granroth‡

Abstract—Digital twins of neutron instruments using Monte Carlo ray tracing have proven to be useful in neutron data analysis and verifying instrument and sample designs. However, these simulations can become quite complex and computationally demanding with tens of billions of neutrons. In this paper, we present a GPU accelerated version of MCViNE using Python and Numba to balance user extensibility with performance. Numba is an open-source just-in-time (JIT) compiler for Python using LLVM to generate efficient machine code for CPUs and GPUs with NVIDIA CUDA. The JIT nature of Numba allowed complex instrument kernels to be generated easily. Initial simulations have shown a speedup between 200-1000x over the original CPU implementation. The performance gain with Numba enables more sophisticated data analysis and impacts neutron scattering science and instrument design.

Index Terms—Monte Carlo, numba, digital twin, Python, neutron, GPU, ray tracing, CUDA

Introduction

MCViNE [1], [2] is a software package for creating digital twins of neutron scattering experiments using a Monte Carlo ray-tracing approach. In this method, randomly generated probability packets (representing neutrons) are propagated through a series of components. Each component changes the probability packets according to the physics of the component. As an example of a component, consider a neutron mirror with less than perfect reflectivity. The interaction between the probability packet and the mirror would cause the velocity component perpendicular to the mirror to reverse and would reduce the probability of the packet to take into account that there is a finite probability that the neutron would not be reflected. The physics of each MCViNE component is documented in the code. An extensive description of components for a similar package, McStas is provided at [3]. There is no correlation between packets, so the system is embarrassingly parallel. These simulations are useful in performing advanced neutron data analysis [4], [5], [6], [7], [8], [9] as well as in designing novel neutron instruments [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20] and sample environments [21], [22]. Specifically, it has been used in the initial designs for instruments in the Second Target Station at the Spallation Neutron Source (SNS) [23] at Oak Ridge National Laboratory. Currently, MCViNE only runs on CPUs which is a bottleneck in large simulations with tens of billions of neutrons, and in modelling complex multiple scattering, with some simulations taking months to complete. Due to the massively parallel nature of Monte Carlo methods, bringing GPU acceleration to these simulations would offer superior performance and scalability. MCViNE was originally implemented in C++ and parallelized using MPI, with bindings to Python for user interaction. However, extensibility for the end user can be very difficult.

To further improve performance and to create an easily extensible code base, Python and Numba [24] were chosen to create a new package of MCViNE providing GPU acceleration, `mcvine.acc` [25]. Numba is an open-source JIT (just-in-time) compiler for Python using LLVM to generate efficient machine code and supports GPUs using NVIDIA CUDA. Numba is designed for scientific computing and can support NumPy arrays and functions. Currently, we are only using Numba for its GPU capabilities as the original version of MCViNE is used to run on CPUs. This accelerated MCViNE package is compatible with existing MCViNE scripts, and using a mixture of CPU and GPU components is supported.

This paper will first describe how MCViNE works at a high-level, how components and instruments are created using Numba to generate CUDA kernels, and how Numba is also used to generate kernels for complex sample geometries and scatterers. Next, we will compare performance of the CPU version of MCViNE to the Numba GPU version. After that we will describe how the MCViNE GPU acceleration will be used in the larger context of a workflow for data analysis. Finally, we will discuss our experience using Numba for this application.

Method

MCViNE Overview

MCViNE simulations are run from a script that defines an instrument, where an instrument is composed of multiple components. A simple 4 component instrument is shown in Figure 1. Full instruments may have several hundred components. Each instrument script is run with a specified number of probability packets, where each packet has several state variables: position, velocity, spin, probability, and time.

At a high-level, a component takes a neutron as input and performs some action on the neutron. Components can be attached to the instrument at a specified position and orientation. Some of the main component types are sources, guides, monitors, samples,

* Corresponding author: kendrickcj@ornl.gov
‡ Oak Ridge National Laboratory

and detectors. In a full instrument simulation, neutrons are generated from a source component and are propagated through each component in the instrument. Sample components are a special type of component with additional input files to specify geometry and material composition.

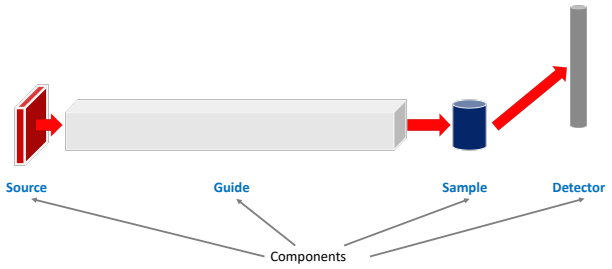

Fig. 1: *Example instrument with four components: a source, guide, sample, and detector.*

Component Hierarchy

One of the major benefits of using Python for this application is the ease of utilizing an object-oriented design and polymorphism. Each component inherits from a base `AbstractComponent` class. This `AbstractComponent` class requires a "propagate" method to be defined which takes in a neutron for the first parameter. The "propagate" method is responsible for defining the action of the component and is decorated with `@cudajit` to indicate that it is a CUDA kernel. An example of creating a simple component can be seen below. Additional parameters can be specified, but must also be defined in the `propagate_params` list. There are several major types of components that have their own subclasses: `SampleBase`, `SourceBase`, and `MonitorBase`.

```
from mcvine.acc.ComponentBase import ComponentBase
class Arm(ComponentBase):

    def __init__(self, name, **kwargs):
        self.name = name
        self.propagate_params = ()

        # Aim a neutron at this arm to
        # cause JIT compilation.
        import mcni
        neutrons = mcni.neutron_buffer(1)
        neutrons[0] = \
            mcni.neutron(r=(0, 0, -1),
                         v=(0, 0, 1),
                         prob=1, time=0)
        self.process(neutrons)

    @cuda.jit(void(NB_FLOAT[:]),
              device=True)
    def propagate(in_neutron):
        pass
```

CUDA Kernel Generation for an Instrument

In order to run a simulation from an input instrument script, `mcvine.acc` first generates a GPU kernel from the instrument specification. The input script will be parsed and then "compiled" to generate a Python script representing an instrument kernel. This compiled version is then imported and executed to run the simulation. An example of a simple MCViNE instrument script containing a source, guide, and monitor can be seen below.

```
def instrument():
    instrument = mcvine.instrument()
    source = Source_simple(
        'src',
        radius = 0., width = 0.03,
        height = 0.03, dist = 1.,
        xw = 0.035, yh = 0.035
    )
    instrument.append(source,
                      position=(0,0,0.))

    guide1 = Guide(
        'guide',
        w1=0.035, h1=0.035,
        w2=0.035, h2=0.035,
        l=10
    )
    instrument.append(guide1,
                      position=(0,0,1.))

    mon = PosDiv_monitor(
        'mon', xwidth=0.08, yheight=0.08,
        maxdiv=2.,
        npos=250, ndiv=251,
    )
    instrument.append(mon,
                      position=(0,0,12.))
    return instrument
```

When `mcvine.acc` is called on the above instrument script, a new Python file is generated which contains a Numba CUDA kernel for the entire instrument. Each of the instrument components' `process` kernels are collected and inserted to form a generic kernel in this process. This generated kernel effectively models a neutron travelling through the entire instrument.

Depending on the kernel launch configuration, each GPU thread might be responsible for more than one neutron. An example of a compiled instrument script can be seen in the code listing below. As seen in the script, a CUDA kernel is defined using Numba. Inside the kernel, each GPU thread will loop over the number of neutrons it is processing. Each `propagate` function has a number appended to it. These propagate functions correspond to the Component's propagate method. For this case, `propagate0` matches the Source component propagate kernel, `propagate1` matches the Guide component propagate kernel, and so on. The `applyTransformation` function is inserted in-between instrument components and is responsible for translating the position/velocity of a neutron in the current component's coordinate system to that of the next component by applying a transformation matrix.

```
@cuda.jit
def process_kernel_no_buffer(
    rng_states, N, n_neutrons_per_thread,
    args
):
    args0, args1, args2, offsets, rotmats, neutron_counter = args
    thread_index = cuda.grid(1)
    start_index = thread_index*n_neutrons_per_thread
    end_index = min(start_index+n_neutrons_per_thread, N)
    neutron = cuda.local.array(shape=10, dtype=NB_FLOAT)
    r = cuda.local.array(3, dtype=NB_FLOAT)
    v = cuda.local.array(3, dtype=NB_FLOAT)
    for neutron_index in range(start_index, end_index):
        cuda.atomic.add(neutron_counter, 0, 1)
        propagate0(thread_index, rng_states,  neutron, *args0)
        applyTransformation(neutron[:3], neutron[3:6],
                            rotmats[0], offsets[0], r, v)
        propagate1(neutron, *args1)
        applyTransformation(neutron[:3], neutron[3:6],
                            rotmats[1], offsets[1], r, v)
        propagate2(neutron, *args2)

from mcvine.acc.components.sources.SourceBase import SourceBase
class _Base(SourceBase): # has to be named Base in definition
    def __init__(self, instrument):
        offsets, rotmats = calcTransformations(instrument)
        self.neutron_counter = neutron_counter = np.zeros(1, dtype=int)
        self.propagate_params = tuple(
            c.propagate_params for c in instrument.components)
        self.propagate_params += (offsets, rotmats, neutron_counter)
```

```
        return
    def propagate(self): return
InstrumentWrapper = _Base
InstrumentWrapper.process_kernel_no_buffer = process_kernel_no_buffer

def run(ncount, ntotalthreads=None, threads_per_block=None, **kwds):
    instrument = loadInstrument(script, **kwds)
    iw = InstrumentWrapper(instrument)
    iw.process_no_buffer(ncount, ntotalthreads=ntotalthreads,
                         threads_per_block=threads_per_block)
    processed = iw.neutron_counter[0]
    saveMonitorOutputs(instrument, scale_factor=1.0/ncount)
```

Sample Kernels and CSG

One unique feature of MCViNE is its sample component, which allows for simulation of complex sample/sample environment and detector systems with flexible, sophisticated geometry and scattering physics. This feature has enabled simulations that produce virtual experiment data that closely resemble real experimental data and make MCViNE a useful tool for both instrument design and advanced data analysis.

A Sample is made up of a Sample Assembly which has `Shape` and `Phase` tags. The `Shape` tag contains the geometry specification. The `Phase` tag, together with additional XML file(s) named `{name}-scatterer.xml`, contain the scattering physics specification where `{name}` is the name of a scatterer included in the sample assembly. A simple example for an aluminum (Al) sphere is shown below. Such an Al sphere is an idealized sample, where the sphere matches the scattering condition and Al is the material most transparent to neutron and is used for mounts and sample containers. Therefore it is well studied and well understood.

```
<?xml version="1.0"?>

<!DOCTYPE SampleAssembly>

<SampleAssembly name="isotropic_sphere">
  <PowderSample name="sample" type="sample">
    <Shape>
      <sphere radius="1*cm" />
    </Shape>
    <Phase type="crystal">
      <ChemicalFormula>Al</ChemicalFormula>
      <xyzfile>Al.xyz</xyzfile>
    </Phase>
  </PowderSample>
  <LocalGeometer registry-coordinate-system="InstrumentScientist">
    <Register name="sample" position="(0,0,0)" orientation="(0,0,0)"/>
  </LocalGeometer>
</SampleAssembly>
```

Simple shapes can be created easily, but more complex shapes can be created too. To create the sample geometry, MCViNE uses Constructive Solid Geometry (CSG). CSG can create complex geometries from operations such as intersection and union, on primitive shapes, such as cubes, spheres, and cylinders. For example, Figure 2 shows how CSG is used to create an annular sample can from cylindrical primitives. This can is similar to those used in experiment design [26]. First a ring for the outside of the can is made from two cylinders by subtraction, then a ring for the inside of the can is also made from two cylinders. These two rings are then combined in union with another cylinder that is the bottom of the can.

Ray tracing routines are implemented as CUDA kernels for these primitive shapes and operations. To support complex geometries that might involve many operations and shapes, the visitor pattern is used which constructs a single CUDA kernel to handle the ray intersection of that shape. This highlights one of the major advantages of using Python and Numba, as the ability to generate a CUDA kernel dynamically at run-time would be much more difficult to implement in other languages.

The specification of the scattering physics of a particular neutron scatterer is described in a dedicated "scatterer" XML file,

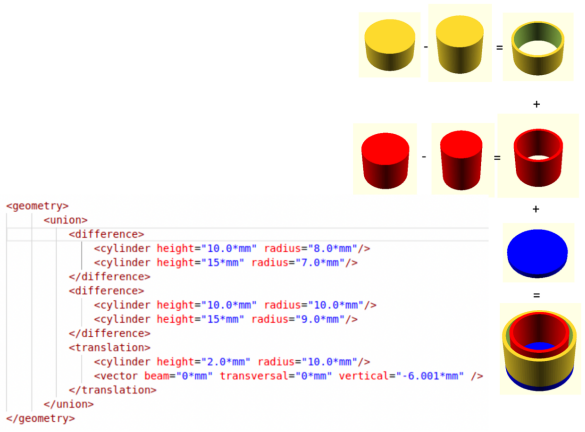

Fig. 2: *An example of an annular sample can created using CSG (right) and how the primitives and operations are specified in XML as input to MCViNE.*

where one or more sample kernels can be specified. An example of a scatterer XML file for specifying scattering physics can be seen below.

```
<?xml version="1.0"?>

<!DOCTYPE scatterer>

<!-- weights: absorption, scattering, transmission -->
<homogeneous_scatterer mcweights="0, 1, 0">
  <IsotropicKernel absorption_coefficient="10./m" scattering_coefficient="10./m">
  </IsotropicKernel>
</homogeneous_scatterer>
```

Note the format is extensible enough to allow a composite scatterer with multiple scatters, though at the time of writing this paper the Numba version of this functionality is still under development.

Results

Two types of comparisons were performed to show the usefulness of `mcvine.acc`. First, simulations comparing the CPU and GPU shows a significant performance gain by using a GPU (Figures 3 – 5). Second, simulations from a more complete instrument solution showing equivalent outcomes from a CPU and GPU simulation were performed (Figure 6 and 7).

For the first study, we focus on the performance gain achieved by the GPU accelerated version of MCViNE. We used a simple instrument consisting of a source, sample, and detector to focus on the sample assembly component. We performed tests with two different samples: a simple spherical sample with an isotropic scattering kernel, and a second with a more complex Uranium Nitride (UN) sample. The UN sample was chosen as it has been experimentally studied and is well modeled by single and multiple scattering of a Quantum Harmonic oscillator model [27], [9]. The UN sample is treated as a 1 cm polycrystalline cube to match the experimental configuration [27]. The UN structure is the same as rock salt structure with the light N atoms located between the much heavier U atoms. The N vibrates as a harmonic oscillator which provides equally spaced lines in energy transfer E. The lines are modeled in MCViNE with a sample scattering kernel containing a composite of $E(Q)$ kernels with constant E values of a 50 meV spacing from 0 to 350 meV.

Figure 3 shows the performance of the isotropic sphere sample for the CPU version of MCViNE with one and 16 cores (blue and

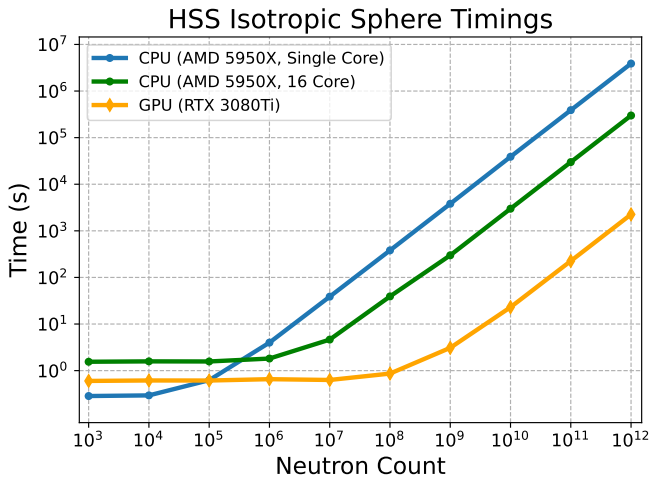

Fig. 3: *Time versus neutron counts for a single core CPU (blue line), multi-core CPU (green line), and GPU (orange line) for a simple instrument using a homogeneous single scatterer (HSS) with a aluminum sphere sample.*

green curves) compared to the GPU version of MCViNE (orange curve). At 10^{12} neutrons, `mcvine.acc` achieves a speedup of 1725x over a single core and 133x over 16 cores.

Two versions of this simulation were run for the UN sample: the first with only single scattering events (Figure 4), and the second with single and multiple scattering events (Figure 5). Single scattering was implemented first to verify the overall workflow and kernel generation of `mcvine.acc`. Multiple scattering was then added to fully capture the realistic scattering physics. Multiple scattering is much more computationally intensive since each neutron can scatter more than once.

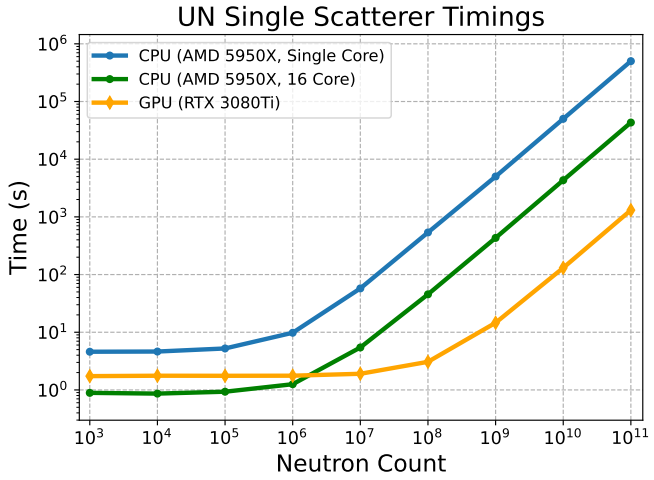

Fig. 4: *Time versus neutron counts for a single core CPU (blue line), multi-core CPU (green line), and GPU (orange line) and GPU (orange line) for the UN instrument with single scattering.*

For the UN single scattering case, Figure 4 shows that for 10^{11} neutrons, the GPU version obtained a speedup of 383x over a single core, and 33x over 16 cores. For the UN multiple scattering case, Figure 5 shows that for 10^{10} neutrons, the GPU version obtained a speedup of 137x over a single core, and 10x over 16 cores. Comparing the speedup achieved for the simple isotropic

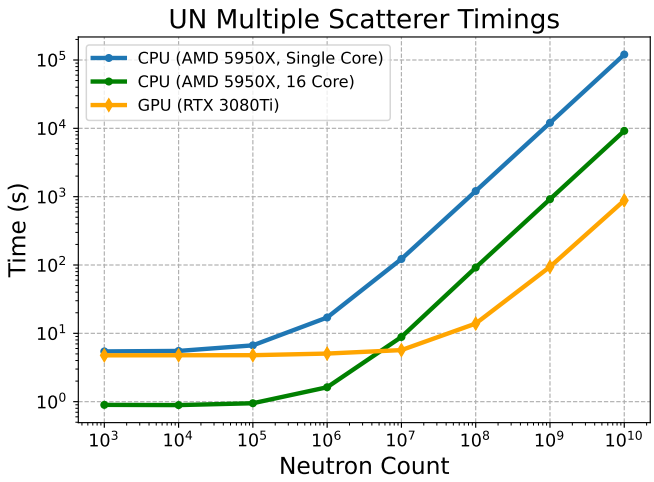

Fig. 5: *Time versus neutron counts for a single core CPU (blue line), multi-core CPU (green line), and GPU (orange line) for the UN instrument with multiple scattering.*

Simulation Type	Speedup over 1 Core	Speedup over 16 Cores
HSS Isotropic Sphere	1725	133
UN Single Scatterer	383	33
UN Multiple Scatterer	137	10

TABLE 1: *Speedup achieved by `mcvine.acc` over the CPU with one core and 16 cores for each type of simulation.*

sphere to the UN with single and multiple scattering shows the additional complexity required for the UN sample. A speedup of 10x over 16-core CPU for the UN multiple scattering case is still significant as some simulations can take on the order of days to months to complete.

While the speedup over the CPU version of MCViNE is significant, further optimization is possible. Currently, each GPU thread executes a single large kernel that models the instrument. For large instruments that contain many components, the instrument kernel can use too many registers which limits device occupancy. Additionally, a lot of components involve many conditional statements which do not perform well on the GPU. This can be seen by comparing the performance of complex sample components, such as the UN sample, to the simple isotropic sphere.

Next we run a more complete test with the Uranium Nitride sample to verify the same result between the CPU and GPU. A study on UN was the first time the multiple-scattering as well as the multiple-scattering physics in the CPU version of MCViNE was used to explain experimental results [9]. Specifically, one of the puzzles from the measured data was that the equally spaced lines extend over all Q. It was determined that this was due to multiple scattering. To more conclusively check this, a CPU simulation using MCViNE was performed [9]. At the time this CPU simulation was run, it took days to do such calculations. Therefore, this is a good test case to check the speed increase gained from using GPUs with `mcvine.acc`.

In this case, the incident beamline simulation (the simulation up to the sample containing the SNS source, guide choppers and slits) for the Wide Angular-Range Chopper Spectrometer (ARCS) instrument [28], [29] was run using McStas [30], [31] inside a workflow tool [32]. To configure the incident beam simulation in this workflow a user simply provides an existing experimental data

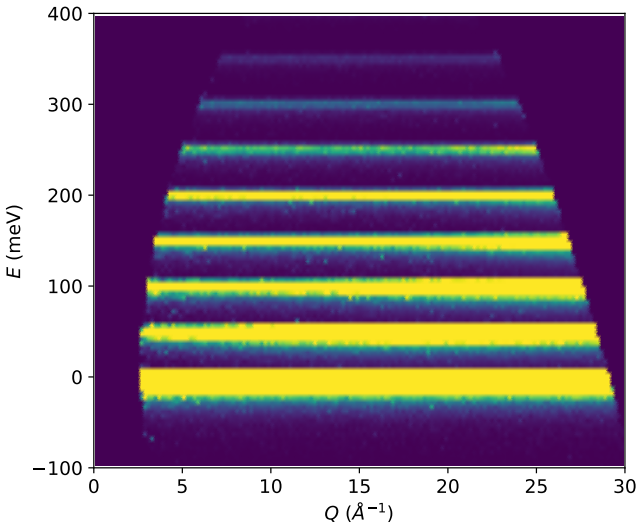

Fig. 6: *The results from the UN simulation on ARCS with multiple scattering turned on. Color indicates the scattering intensity where brighter regions represent higher intensity. Note that the equally spaced lines are visible even at low Q.*

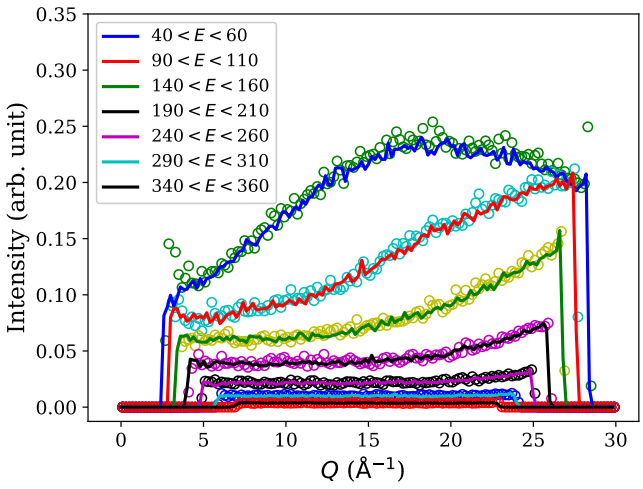

Fig. 7: *Constant-E cuts around the individual energy levels in the $I(Q,E)$ displayed in Figure 6. Note the functional dependence of each level and the background increase are consistent with the physics and the multiple scattering. The comparisons between the CPU and GPU calculations are shown with the solid line and circles respectively.*

file that provides the necessary parameters. Specifically, an ARCS data file with an $E_i = 500$ meV was fed into the workflow which then generates an mcpl [33] file [34] for use in MCViNE. The rest of the virtual instrument uses `mcvine.acc` to leverage the GPU acceleration. It consists of a source component that reads the mcpl file to generate neutrons for the neutron source component, a sample assembly component, and a powder $S(Q,E)$ monitor component for direct-geometry inelastic neutron spectrometers. The results are shown in Figure 6 and 7.

First, note the equally spaced lines in E shown in Figure 6. This the expected quantum oscillator behavior. Furthermore, Figure 7 shows the Q dependence of the scattering intensity along each of the E lines in Figure 6. The expected functional dependence for each successive transition and the overall increase in background expected from multiple scattering are both observed.

As this paper focuses on the GPU implementation, Figure 6 and 7 also show the agreement between the CPU and GPU versions of MCViNE. The majority of the speed increase for this particular simulation is in the incident beam line simulation leveraging the McStas GPU implementation, and is now under an hour rather than days. The MCViNE part of the simulation has a speed up similar to the simpler test from $\sim 10^3$s to $\sim 10^2$s. For a virtual neutron experiment the incident beam simulation can often be reused in a series of source-sample-detector simulations with various sample and detector configurations. For example, a researcher may run the case with and without multiple scattering or a series of related samples. Thus fast sample simulations are critical to the overall speed of experimental analysis which highlights the need for using `mcvine.acc`.

Conclusions

Python and Numba were used successfully to create `mcvine.acc`, a new GPU accelerated version of MCViNE, which has so far achieved significant performance gains over the original CPU implementation. Using Python for this application has helped increase the usability, extensibility, and maintainability of the codebase, while gaining performance benefits of GPUs by using Numba. Additionally, the JIT nature of Numba allowed complex combinations of CUDA kernels to be generated at runtime, which would have been significantly harder to implement in other languages.

The performance gains from using Numba have shown to be beneficial. For a simple isotropic sphere sample, a speedup of 133x was achieved over a 16-core CPU using a consumer-grade GPU. For the more complex UN sample with multiple scattering, a speedup of 10x was achieved over a 16-core CPU. These performance gains are crucial for current simulations that take on the order of days to weeks to complete. However, there are still opportunities to further optimize these simulations to better leverage the full capability of the GPU.

Using Numba for GPU acceleration has enabled more sophisticated data analysis for neutron scattering and instrument design, while overall lowering the development cost needed to obtain significant performance improvements. The techniques used in this project could also be applied to other scientific computing applications.

Acknowledgements

For initial Framework development and instrument component development, this research used resources of the Spallation Neutron Source Second Target Station Project at Oak Ridge National Laboratory (ORNL). Sample development was sponsored by ORNL's Laboratory Director's Research and Development Fund. ORNL is managed by UT-Battelle LLC for DOE's Office of Science, under Contract No. DE-AC05-00OR22725.

The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

REFERENCES

[1] J. Y. Y. Lin, H. L. Smith, G. E. Granroth, D. L. Abernathy, M. D. Lumsden, B. Winn, A. A. Aczel, M. Aivazis, and B. Fultz, "MCViNE–an object oriented monte carlo neutron ray tracing simulation package," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 810, pp. 86–99, 2016, https://doi.org/10.1016/j.nima.2015.11.118.

[2] J. Y. Y. Lin, F. Islam, G. Sala, I. Lumsden, H. Smith, M. Doucet, M. B. Stone, D. L. Abernathy, G. Ehlers, J. F. Ankner, and G. E. Granroth, "Recent developments of MCViNE and its applications at SNS," *Journal of Physics Communications*, vol. 3, no. 8, p. 085005, Aug. 2019, https://doi.org/10.1088/2399-6528/ab3622. [Online]. Available: https://doi.org/10.1088%2F2399-6528%2Fab3622

[3] P. Willendrup, E. Farhi, E. Knudsen, U. Filges, and K. Lefmann, *Component Manual for the Neutron Ray-Tracing Package McStas*. Danish Techincal University, 2022.

[4] J. Y. Y. Lin, G. Sala, and M. B. Stone, "A super-resolution technique to analyze single-crystal inelastic neutron scattering measurements using direct-geometry chopper spectrometers," *Review of Scientific Instruments*, vol. 93, no. 2, p. 025101, 2022, https://doi.org/10.1063/5.0079031.

[5] F. Islam, J. Y. Y. Lin, R. Archibald, D. L. Abernathy, I. Al-Qasir, A. A. Campbell, M. B. Stone, and G. E. Granroth, "Super-resolution energy spectra from neutron direct-geometry spectrometers," *Review of Scientific Instruments*, vol. 90, no. 10, p. 105109, 2019, https://doi.org/10.1063/1.5116147.

[6] G. Sala, J. Y. Y. Lin, A. M. Samarakoon, D. S. Parker, A. F. May, and M. B. Stone, "Ferrimagnetic spin waves in honeycomb and triangular layers of $Mn_3Si_2Te_6$," *Phys. Rev. B*, vol. 105, p. 214405, Jun 2022, https://doi.org/10.1103/PhysRevB.105.214405. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.105.214405

[7] S.-H. Do, K. Kaneko, R. Kajimoto, K. Kamazawa, M. B. Stone, J. Y. Y. Lin, S. Itoh, T. Masuda, G. D. Samolyuk, E. Dagotto *et al.*, "Damped dirac magnon in the metallic kagome antiferromagnet FeSn," *Physical Review B*, vol. 105, no. 18, p. L180403, 2022, https://doi.org/10.1103/physrevb.105.l180403.

[8] J. C. Leiner, H. O. Jeschke, R. Valentí, S. Zhang, A. T. Savici, J. Y. Y. Lin, M. B. Stone, M. D. Lumsden, J. Hong, O. Delaire, W. Bao, and C. L. Broholm, "Frustrated magnetism in mott insulating $(v_{1-x}cr_x)_2o_3$," *Phys. Rev. X*, vol. 9, p. 011035, Feb 2019, https://doi.org/10.1103/PhysRevX.9.011035. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.9.011035

[9] J. Y. Y. Lin, A. A. Aczel, D. L. Abernathy, S. E. Nagler, W. Buyers, and G. E. Granroth, "Using monte carlo ray tracing simulations to model the quantum harmonic oscillator modes observed in uranium nitride," *Physical Review B*, vol. 89, no. 14, p. 144302, 2014, https://doi.org/10.1103/physrevb.89.144302.

[10] E. Mamontov, C. Boone, M. Frost, K. Herwig, T. Huegle, J. Y. Y. Lin, B. McCormick, W. McHargue, A. Stoica, P. Torres *et al.*, "A concept of a broadband inverted geometry spectrometer for the Second Target Station at the Spallation Neutron Source," *Review of Scientific Instruments*, vol. 93, no. 4, p. 045101, 2022, https://doi.org/10.1063/5.0086451.

[11] K. An, A. D. Stoica, T. Huegle, J. Y. Y. Lin, and V. Graves, "MENUS—materials engineering by neutron scattering," *Review of Scientific Instruments*, vol. 93, no. 5, p. 053911, 2022, https://doi.org/10.1063/5.0089783.

[12] G. Sala, M. Mourigal, C. Boone, N. P. Butch, A. Christianson, O. Delaire, A. DeSantis, C. Hart, R. P. Hermann, T. Huegle *et al.*, "CHESS: The future direct geometry spectrometer at the Second Target Station," *Review of Scientific Instruments*, vol. 93, no. 6, p. 065109, 2022, https://doi.org/10.1063/5.0089740.

[13] V. O. Garlea, S. Calder, T. Huegle, J. Y. Y. Lin, F. Islam, A. Stoica, V. B. Graves, B. Frandsen, and S. D. Wilson, "VERDI: Versatile diffractometer with wide-angle polarization analysis for magnetic structure studies in powders and single crystals," *Review of Scientific Instruments*, vol. 93, no. 6, p. 065103, 2022, https://doi.org/10.1063/5.0090919.

[14] G. E. Borgstahl, W. B. O'Dell, M. Egli, J. F. Kern, A. Kovalevsky, J. Y. Y. Lin, D. Myles, M. A. Wilson, W. Zhang, P. Zwart *et al.*, "EWALD: A macromolecular diffractometer for the Second Target Station," *Review of Scientific Instruments*, vol. 93, no. 6, p. 064103, 2022, https://doi.org/10.1063/5.0090810.

[15] Y. Liu, H. Cao, S. Rosenkranz, M. Frost, T. Huegle, J. Y. Y. Lin, P. Torres, A. Stoica, and B. C. Chakoumakos, "PIONEER, a high-resolution single-crystal polarized neutron diffractometer," *Review of Scientific Instruments*, vol. 93, no. 7, p. 073901, 2022, https://doi.org/10.1063/5.0089524.

[16] S. Qian, W. Heller, W.-R. Chen, A. Christianson, C. Do, Y. Wang, J. Y. Y. Lin, T. Huegle, C. Jiang, C. Boone *et al.*, "CENTAUR—the small-and wide-angle neutron scattering diffractometer/spectrometer for the Second Target Station of the Spallation Neutron Source," *Review of Scientific Instruments*, vol. 93, no. 7, p. 075104, 2022, https://doi.org/10.1063/5.0090527.

[17] C. Do, R. Ashkar, C. Boone, W.-R. Chen, G. Ehlers, P. Falus, A. Faraone, J. S. Gardner, V. Graves, T. Huegle *et al.*, "EXPANSE: A time-of-flight expanded angle neutron spin echo spectrometer at the Second Target Station of the Spallation Neutron Source," *Review of Scientific Instruments*, vol. 93, no. 7, p. 075107, 2022, https://doi.org/10.1063/5.0089349.

[18] J. Ankner, R. Ashkar, J. Browning, T. Charlton, M. Doucet, C. Halbert, F. Islam, A. Karim, E. Kharlampieva, S. Kilbey *et al.*, "Cinematic reflectometry using QIKR, the quite intense kinetics reflectometer," *Review of Scientific Instruments*, vol. 94, no. 1, p. 013302, 2023, https://doi.org/10.1063/5.0122279.

[19] A. Brugger, H. Z. Bilheux, J. Y. Y. Lin *et al.*, "The Complex, Unique, and Powerful Imaging instrument for Dynamics (CUPI2D) at the Spallation Neutron Source," *Review of Scientific Instruments*, vol. 94, no. 5, p. 051301, 2023, https://doi.org/10.1063/5.0131778. [Online]. Available: https://doi.org/10.1063/5.0131778

[20] J. Y. Y. Lin, T. Huegle, L. Coates, and A. D. Stoica, "A realistic guide misalignment model for the Second Target Station instruments at the Spallation Neutron Source," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 1047, p. 167881, 2023, https://doi.org/10.1016/j.nima.2022.167881.

[21] M. B. Stone, G. Sala, and J. Y. Y. Lin, "Design of a radial collimator for the SEQUOIA direct geometry chopper spectrometer," *Physica B: Condensed Matter*, vol. 564, pp. 17–21, 2019, https://doi.org/10.1016/j.physb.2018.11.042.

[22] J. L. Niedziela, R. Mills, M. J. Loguillo, H. D. Skorpenske, D. Armitage, H. L. Smith, J. Y. Y. Lin, M. S. Lucas, M. B. Stone, and D. L. Abernathy, "Design and operating characteristic of a vacuum furnace for time-of-flight inelastic neutron scattering measurements," *Review of Scientific Instruments*, vol. 88, no. 10, p. 105116, 2017, https://doi.org/10.1063/1.5007089.

[23] T. E. Mason, D. Abernathy, I. Anderson, J. Ankner, T. Egami, G. Ehlers, A. Ekkebus, G. Granroth, M. Hagen, K. Herwig, J. Hodges, C. Hoffmann, C. Horak, L. Horton, F. Klose, J. Larese, A. Mesecar, D. Myles, J. Neuefeind, M. Ohl, C. Tulk, X.-L. Wang, and J. Zhao, "The Spallation Neutron Source in Oak Ridge: A powerful tool for materials research," *Physica B: Condensed Matter*, vol. 385, pp. 955–960, 2006, https://doi.org/10.1016/j.physb.2006.05.281.

[24] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015, pp. 1–6, https://doi.org/10.1145/2833157.2833162.

[25] J. Lin and C. Kendrick, "mcvine.acc," https://github.com/mcvine/acc. [Online]. Available: https://github.com/mcvine/acc

[26] T. R. Prisk, R. T. Azuah, D. L. Abernathy, G. E. Granroth, T. E. Sherline, P. E. Sokol, J. Hu, and M. Boninsegni, "Zero-point motion of liquid and solid hydrogen," *Phys. Rev. B*, vol. 107, p. 094511, Mar 2023, https://doi.org/10.1103/PhysRevB.107.094511. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.107.094511

[27] A. A. Aczel, G. E. Granroth, G. J. MacDougall, W. Buyers, D. L. Abernathy, G. D. Samolyuk, G. M. Stocks, and S. E. Nagler, "Quantum oscillations of nitrogen atoms in uranium nitride," *Nature Communications*, vol. 3, p. 1124, 2012, https://doi.org/10.1038/ncomms2117.

[28] D. L. Abernathy, M. B. Stone, M. J. Loguillo, M. S. Lucas, O. Delaire, X. Tang, J. Y. Y. Lin, and B. Fultz, "Design and operation of the wide angular-range chopper spectrometer ARCS at the Spallation Neutron Source," *Review of Scientific Instruments*, vol. 83, no. 1, 2012, https://doi.org/10.1063/1.3680104.

[29] M. B. Stone, J. L. Niedziela, D. L. Abernathy, L. DeBeer-Schmitt, G. Ehlers, O. Garlea, G. E. Granroth, M. Graves-Brook, A. I. Kolesnikov, A. Podlesnyak, and B. Winn, "A comparison of four direct geometry time-of-flight spectrometers at the Spallation Neutron Source," *Review of Scientific Instruments*, vol. 85, no. 4, p. 045113, 2014, https://doi.org/10.1063/1.4870050.

[30] P. K. Willendrup and K. Lefmann, "McStas (i): Introduction, use, and basic principles for ray-tracing simulations," *Journal of Neutron Research*, vol. 22, no. 1, pp. 1–16, 2020, https://doi.org/10.3233/JNR-190108.

[31] ——, "McStas (ii): An overview of components, their use, and advice for user contributions," *Journal of Neutron Research*, vol. 23, no. 1, pp. 7–27, 2021, https://doi.org/10.3233/JNR-200186.

[32] G. R. Watson, G. Cage, J. Fortney, G. E. Granroth, H. Hughes, T. Maier, M. McDonnell, A. Ramirez-Cuesta, R. Smith, S. Yakubov, and W. Zhou, "Calvera: A platform for the interpretation and analysis of neutron scattering data," in *Accelerating Science and Engineering Discoveries*

Through Integrated Research Infrastructure for Experiment, Big Data, Modeling and Simulation, K. Doug, G. Al, S. Pophale, H. Liu, and S. Parete-Koon, Eds. Cham: Springer Nature Switzerland, 2022, pp. 137–154.

[33] T. Kittelmann, E. Klinkby, E. Knudsen, P. Willendrup, X. Cai, and K. Kanaki, "Monte Carlo Particle Lists: MCPL," *Computer Physics Communications*, vol. 218, pp. 17–42, 2017, https://doi.org/10.1016/j.cpc.2017.04.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465517301261

[34] G. Granroth, D. L. Abernathy, J. Lin, W. Zhou, and P. K. Wilendrup, "Incident beamline simulation for ei= 510 mev on the ARCS spectrometer at the Spallation Neutron Source," https://doi.org/10.13139/ORNLNCCS/1975747, 6 2023, https://doi.org/10.13139/ORNLNCCS/1975747. [Online]. Available: https://doi.ccs.ornl.gov/ui/doi/438