5-6-2022

Improving the Efficiency of the Preconditioning of Iterative Solutions to the Kinetic Equation

D. Caleb Price
Utah State University

Improving the Efficiency of the Preconditioning of Iterative Solutions to the Kinetic Equation

D. Caleb Price

May 6, 2022

Abstract

To achieve the reality of fusion, a greater understanding of plasma is required. The kinetic equation can be evolved simultaneously alongside the fluid equations to solve for kinetic closures. NIMROD performs this with numerical solvers where the General Minimum Residual (GMRES) solver becomes more efficient with a preconditioning matrix as input. Using a GPU-enabled library, the efficiency of GPU offloading to the preconditioning step was tested. A significant decrease in the factoring time of preconditioning matrix was observed. This suggests that the allocation of GPUs is worth investigating for NIMROD's own benefit, but also anyone seeking to improve the efficiency of their scientific program.

Keywords: Plasma, iterative solutions, GPU offloading

Introduction

The quest for net fusion energy as a clean and practically limitless energy source continues as organizations like ITER seek to create a tokamak capable of maintaining thermonuclear fusion long enough to be viable. Non-Ideal MHD with Rotation - Open Discussion (NIMROD) seeks to solve for kinetic closures to equations that describe the plasma behavior within tokamaks through several numerical methods that approximate such solutions. To achieve such a computation now requires the use of supercomputers graciously provided by national organizations all in the pursuit of accomplishing the monumental feat of a fusion energy source.

However, supercomputing recently has evolved to include the GPUs or graphics processing units in their nodes due to their capability to perform small tasks more efficiently compared to their counterpart, the CPU. Anything that improves the efficiency of computation is desired to achieve a more accurate simulation or quicker computation, especially for NIMROD's code. Therefore, this research is to assess the efficiency achieved by GPU offloading the preconditioning step within NIMROD through the use of a GPU-enabled third-party linear algebra package, SuperLU.

Theory

Plasma, as a 4th state of matter, is often modelled as a fluid. It's primarily described by 3 macroscopic quantities: density, flow, and temperature. The kinetic equation describes the time evolution of the plasma distribution function,

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f + \frac{q}{m}[\mathbf{E} + \mathbf{v} \times \mathbf{B}] \cdot \nabla_v f = C(f) \tag{1}$$

where f is the distribution function of spatial location, **x**, particle velocity, **v**, and time, t, **E** is the electric field and **B** is the magnetic field present which NIMROD solves for **E** and **B** respectively using Ohm's and Ampere's Law, ∇_v is a gradient in the velocity space (where the velocity distribution function can be defined as a density of particles within an infinitesimal volume of phase space dxdv), and $C(f)$ is defined as the collision operator and is a function of f.

The kinetic equation can be simplified to its lowest order moment equations, known as the fluid equations, [1]

Density (Continuity)

$$\frac{dn}{dt} + n\nabla \cdot V = 0 \tag{2}$$

Flow (Momentum)

$$mn\frac{dV}{dt} + \nabla \cdot \overleftrightarrow{p} - en(\mathbf{E} + \mathbf{V} \times \mathbf{B}) = F \tag{3}$$

Temperature (Energy)

$$\frac{3}{2}(\frac{d\overleftrightarrow{p}}{dt} + \overleftrightarrow{p}\nabla \cdot V) + \overleftrightarrow{p} : \nabla V + \nabla \cdot q = W \tag{4}$$

where n is the number density, V is the velocity, $\overleftrightarrow{p}$ is the pressure tensor, F is the collisional friction (the first velocity moment of the collision operator), q is the heat flux density, and W is the collisional energy change (the second velocity moment of the collision operator).[1]

If the fluid being observed is highly collisional, then fluid closures can be assumed and the velocity distribution is nearly Maxwellian. However, plasma in tokamaks is not highly collisional and fluid closures cannot be assumed for $\overleftrightarrow{p}$, q, F, and W which are all dependent on the distribution function.

Therefore, by simultaneously evolving the distribution function from the kinetic equation, the kinetic closures found in the fluid equations can be directly solved for.

The distribution function alone is not simple to solve, seeing as it is a 7-dimensional quantity a dimension of time, t, three spatial dimensions, **x**, and three velocity dimensions, **v**. However for a highly magnetized plasma, the distribution function is nearly symmetric with respect to the gyroangle of the particle's velocity, which is an assumption specific to the drift-kinetic equation. Even with the assumption of toroidal spatial symmetry and poloidal velocity symmetry, the NIMROD code is only capable of achieving a "low resolution" solution efficiently by the following numerical analysis. It begins with discretization.[4]

Lets consider then the five dimensional distribution function, $f = f(R, Z, \xi, s, t)$, as is found in the kinetic drift equation. R, Z, ξ are cylindrical coordinates that describe

the radius, height, and toroidal angle respectively while s is the normalized speed that is normalized by the local thermal speed. First, it is discretized spatially.[3]

$$f(R,Z,\xi,s,t) = \sum_{i}^{I} f_{i,n=0}(\xi,s,t)\alpha_{i,n=0}(R,Z,\phi) \tag{5}$$

$$+ \sum_{i,n>0}^{IN} f_{i,n}(\xi,s,t)\alpha_{i,n}(R,Z,\phi) + f^{*}_{i,n}(\xi,s,t)\alpha^{*}_{i,n}(R,Z,\phi)$$

where N is the number of degrees of freedom in the polodial plane (R,Z) plane and the coefficients to be solved for are $f_{i,n=0}(\xi,s,t)$ while the set of finite element basis functions are defined as $\alpha_{i,n} = \psi_i(R,Z) * e^{in\phi}$ and ψ_i is a 2D polynomial basis function. See Reference 3 for a deeper explanation of the derivation. This to employ a Fourier expansion method of numerical approximation.

Then to new coefficients are discretized according to the velocity elements, as shown by,

$$f_{i,n}(\xi,s,t) = \sum_{l}^{L} f_{i,n,l}(v,t)Q_l(\xi) \tag{6}$$

where the coefficients are only dependent now on v and t while the basis functions will be defined as a set of continuous finite elements using Gauss-Lobatto-Legendre (GLL) polynomials over the pitch angle, $\theta = cos^{-1}(\xi)$, otherwise known as the finite element method.[2]

Finally, all that remains is,

$$f_{i,n,l}(v,t) = \sum_{k}^{K} f_{i,n,l,k}(t)L_k(s)e^{-s^2} \tag{7}$$

where s is the velocity divided by the local thermal speed, N is the number of degrees of freedom, and L_k are non-classical polynomials.[7] This is known as the collocation method which discretizes the speed dependence of the distribution function found in the kinetic drift equation

To understand the scale of this kind of problem, lets define the degrees of freedom to be the number of coefficients present in the linear system, S. Thus depending on how large S is determines how large the matrix that will be solved to determined kinetic closures of the system is. The total number of coefficients is equivalent to $S = I * N * L * K$ or the degrees of freedom found per dimension multiplied by each other. This defines the size of the vector in question and therefore, the matrix must then be size S^2 and thus it isn't hard to see how large this matrix truly is. An example shown in Reference 3. highlights a matrix that would be size $700,000^2$ as a result of the large degrees of freedom present. To solve this matrix then requires an adept iterative solver as well as an effective preconditioning strategy. NIMROD employs a preconditioned GMRES method to perform this.

The iteration of each matrix during the GMRES method has a cost of about $O(2m^2)$ for a dense matrix of size m, which doesn't seem very efficient for a choice of algorithm. However, this can be reduced to an O(m) when a sparse matrix, or a matrix with mostly zeros, with the diagonal containing almost all the values, is assumed to be used.[5] The

matrices that result from the discretization are in fact sparse and so GMRES is indeed efficient as an iterative solver. Yet to achieve far greater efficiency due to the extreme size of the matrix required to provide kinetic closures to the fluid equations, the preconditioning operation before the GMRES iteration is essential.

Preconditioning has been defined as "a means of transforming the original linear system into one which has the same solution, but which is likely to be easier to solve with an iterative solver."[6]. Effectively, it's taking the matrix, $\mathbf{A}$, and simplifying it into a form, $\tilde{\boldsymbol{A}}$ that is easier to factor. Then the preconditioner matrix, $\mathbf{M}$, is the inverse of our approximated form, $\tilde{\boldsymbol{A}}$, such that it obeys the following relationship.

$$\mathbf{M} \approx \tilde{\boldsymbol{A}}^{-1} \longrightarrow \mathbf{MA} \approx \mathbf{I} \tag{8}$$

NIMROD useds a matrix free implemenation of GMRES where the full matrix A is never formed, yet the preconditioning requires forming $\tilde{\boldsymbol{A}}$ and its LU factors. It's this approximation to the original linear system that then is solved for by the GMRES iterative solver and overall, leads to a low resolution solution to the coefficients in the discretized equations.

Therefore, a key aspect to the solving of the coefficients that define the distribution function to provide a greater understanding to the additional moments of the kinetic drift equation, depends on the factoring time of the preconditioning of the matrix.

Computational Efficiency

The computational efficiency of the preconditioning process is the research in question. To improve the efficiency of the numerical solvers that approximate the distribution function, the preconditioning process can be sped up by offloading the factoring of the preconditioning matrix onto GPUs.

While offloading the factoring of these matrices to GPUs isn't ideal due to the large amount of communication required during the process, GPU offloading is best achieved in cases of low-level math and in the case of NIMROD, this is about the best that can be done with the use of a third-party library. Due to GPU computing being such a new advent, scientific computing hasn't evolved to accommodate with this change quite yet, so this testing quite literally might determine whether direct GPU coding and application within NIMROD's code is worth investigating when the technology/code catches up.

Methods

The experiment was achieved primarily through the use of a NERSC (National Energy Research Scientific Computer Center) supercomputer called Perlmutter with practice on Cori. On a virtual server, both the third-party linear algebra package SuperLU and the test version of NIMROD's preconditioning code were compiled along with the supporting libraries. This required greater effort than originally anticipated due to how relatively new Perlmutter was and its continuous updates as well as the learning curve of determining which libraries were necessary for the GPU offloading to be made available for the preconditioning. (Perlmutter is one of the first supercomputers to include 4 GPUs directly on the node)

To test the compiling of the third party libraries and the GPU offloading, a trial session was performed on the Ascent supercomputer from Oak Ridge Leadership Computing

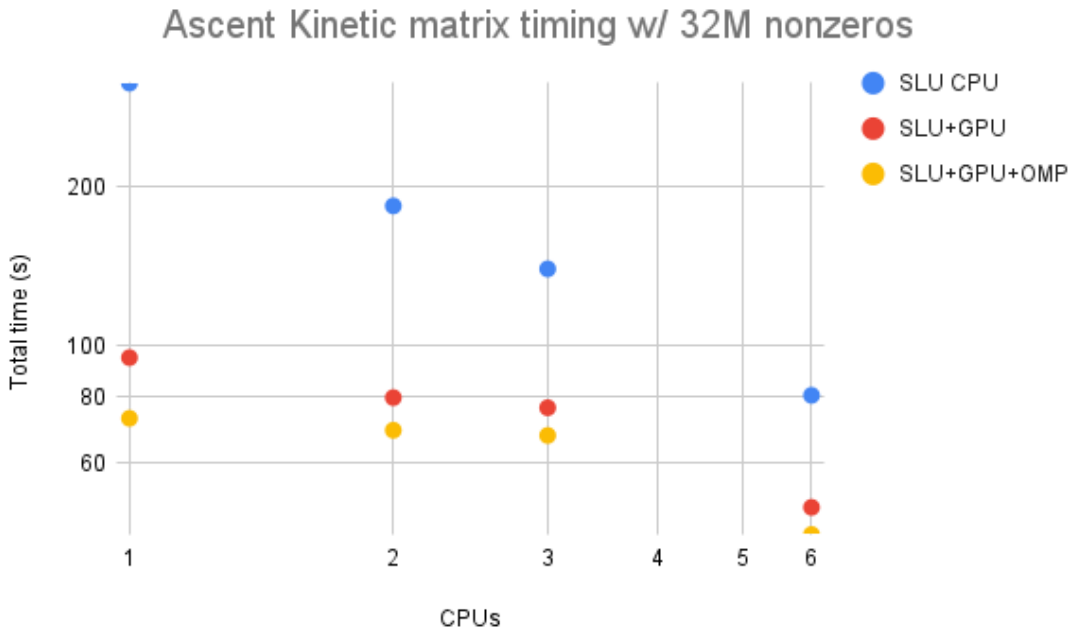

Figure 1: Ascent Results

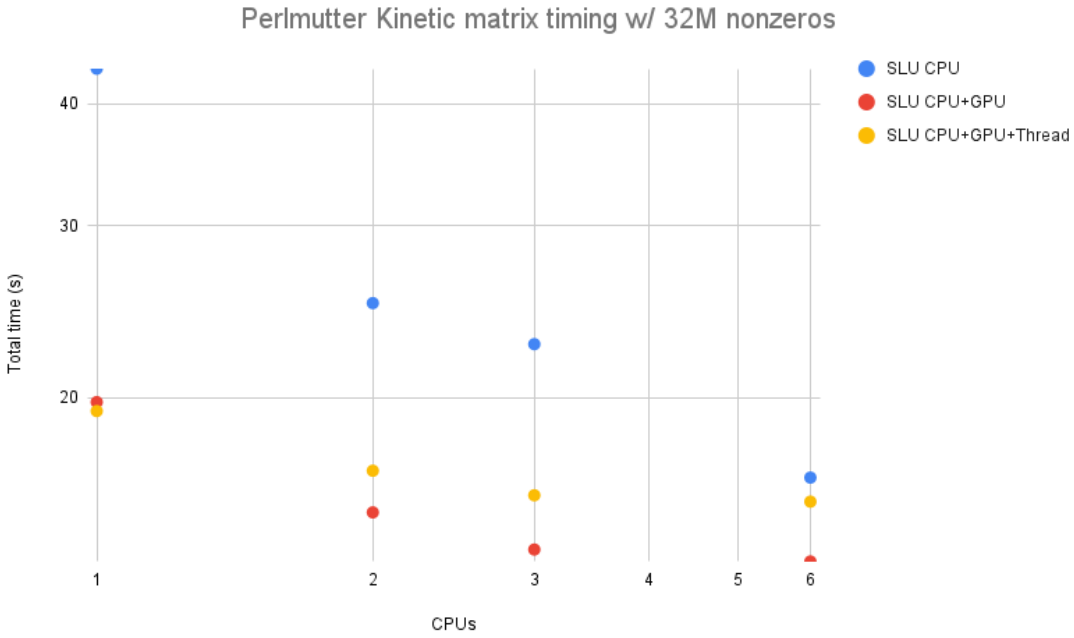

Figure 2: Perlmutter Results

Facility during a GPU hackathon, in order to provide greater perspective and learn to navigate a supercomputer.

To determine the efficacy of bootstrapping GPU's to the preconditioning process, a test was performed using the same data input of matrices, first a base case of only CPU, then one that included GPUs, and finally a case that also included multi-threading with the CPU-GPU node resource set.

Results

The results of the experiment can be divided into two sections, the first one being when computational testing was performed on the Ascent computer at the ORNL as a part of practice and testing of the difficulty of compiling the necessary libraries for the actual computational efficiency testing.

It was observed that the GPU offloading of the factoring of the preconditioning matrix outperformed the version of the code that only allocated resources to CPUs. It's also noteworthy to see how the inclusion of multi-threading again increases the factorization time and thereby total solve time.

The main results however are those found using a test version of NIMROD on Perlmutter with SuperLU, due to the unique nature of the Perlmutter supercomputer.

Again there is a significant increase in the efficiency of the factoring of the preconditioning matrix as seen by the decrease in the total time, in the case of the GPU offloading. It's interesting to note the decrease in how much more efficient the program utilizing GPU offloading compared to the CPU only code as the number of CPU tasks increases. Although this makes sense because this requires greater communication between the CPUs as the task is divided and anything that requires more communication will be less efficient for GPU offloading.

Conclusion

Overall, it can be deduced that the employment of GPU architecture found in supercomputers currently is worth investigating in the actual implementation of code for NIMROD and other computation groups. With the increase in efficiency observed with the use of a third party linear algebra package alone, it makes sense to put a greater effort toward researching how to best communicate with GPUs directly in the code and where in the numerical methods they best fit and have optimal efficiency.

Acknowledgement

All my acknowledgement goes to my mentor, Andrew Spencer. This research would not be possible without his guidance and support through it all. Also, the support from Eric Held and the rest of the NIMROD group was very much appreciated.

I also want to acknowledge my gratitude to my family. Without them, I wouldn't be who I am today. I especially want to thank my mother for always believing that I could be a scientist and that I was truly capable of changing the world.

References

[1] Gurnett, D. A., & Bhattacharjee, A. (2017). INTRODUCTION TO PLASMA PHYSICS With Space, Laboratory and Astrophysical Applications. Cambridge University Press.

[2] J. Andrew Spencer, Brett Adair, Eric D. Held, Jeong-Young Ji, Joseph R. Jepson, Accurate numerical, integral methods for computing drift-kinetic Trubnikov-Rosenbluth potentials, Journal of Computational Physics, Volume 450, 2022, 110862, ISSN 0021-9991,

[3] E.D. Held et. al, Physics of Plasmas 22, 032511 (2015).

[4] C.R. Sovinec et. al, Journal of Computational Physics, Volume 195, 355 (2004). https://doi.org/10.1016/j.jcp.2021.110862.

[5] Generalized minimal residual method. (n.d.). Wikipedia. Retrieved May 3, 2022, from https://en.wikipedia.org/wiki/Generalized_minimal_residual_method

[6] Saad, Y. (2003). Iterative Methods for Sparse Linear Systems (2nd ed.). https://doi.org/10.1137/1.9780898718003

[7]Matt Landreman, Darin R. Ernst, New velocity-space discretization for continuum kinetic calculations and Fokker–Planck collisions, Journal of Computational Physics, Volume 243, 2013, Pages 130-150, ISSN 0021-9991, https://doi.org/10.1016/j.jcp.2013.02.041.