



Large-Scale 3D Phase Field Dislocation Dynamics Simulations On High-Performance Architectures

The International Journal of High Performance Computing Applications
000(00) 1–13
© The Author(s) 2010
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1094342010382534
hpc.sagepub.com



Abigail Hunter¹, Faisal Saied², Chinh Le² and Marisol Koslowski¹

Abstract

In this paper we present the development and performance of a three-dimensional phase field dislocation dynamics (PFDD) model for large-scale dislocation-mediated plastic deformation on high-performance architectures. Through the parallelization of this algorithm, efficient run times can be achieved for large-scale simulations. The algorithm's performance is analyzed over several computing platforms including Infiniband, GigE, and proprietary (SiCortex) interconnects. Scalability is considered on data sets up to 2,048³, along with the efficiency on up to 2,048 processors. Results show that scalability improves as the size of the data set increases and that the overall performance is best on the Infiniband interconnect. In addition, a performance model has been developed to predict run times and efficiency on large sets of data running on multiple processors. This performance analysis shows that this parallel code is capable of harnessing the greater computer power available from petascale systems.

Keywords

1 Introduction

Dislocation dynamics simulations have advanced the understanding of plastic deformation in crystalline materials including several characteristic aspects of plastic deformation such as strength and strain hardening. Plastic deformation of crystalline materials involves complex non-local, non-linear mechanisms with disparate length scales such as long-range dislocation–dislocation interactions, the core structure of the dislocations, and their irreversible interactions with obstacles such as grain boundaries and second phase particles. Owing to these long-range interactions, predictive numerical simulations of plastic deformation in crystalline materials are extremely expensive (Hirth and Lothe, 1968; Bulatov and Cai, 2006; Dimiduk et al., 2006).

Grain refinement in crystalline materials leads to an enhancement of several materials properties including yield and fracture strength and superior wear resistance. These improved material responses benefit the reliability of micro and nano devices, including microelectromechanical systems (MEMS) and nanoelectromechanical systems (NEMS) (Rebeiz, 2003; Walraven, 2003) with components made of nanocrystalline materials. On the other hand, this reduction in grain size is also responsible for new deformation mechanisms that are not present in their coarse-grained counterparts. Some of these mechanisms include size-dependent plastic deformation and plastic strain recovery (Budrovik et al., 2004; Rajagopalan et al., 2007; Uchic et al., 2004).

Numerical simulations should be able to explore these fascinating mechanisms and reveal their key features. One of the challenges in numerical simulations of plastic deformation and one of the reasons these simulations remain limited is that its length and time scales do not lie inside the envelope of traditional multiscale approaches that lead from atomistic processes to a macroscopic response. The characteristic length scales remain in the nanometer range while time scales are in the range of seconds to hours. New models and computer algorithms are necessary to study these complex deformation mechanisms. Therefore, most of the recent efforts in this field are focused on the development of numerical tools to predict plastic deformation in nanocrystalline materials.

Numerical methods describing the evolution of crystalline plasticity can be generally differentiated in two groups, the methods that explicitly track discrete dislocation lines and the methods that follow the evolution of the dislocation density. Discrete dislocations methods explicitly track the

¹School of Mechanical Engineering, Purdue University, West Lafayette, In 47906, USA

²Rosen Center for Advanced Computing, Purdue University, West Lafayette, In 47906, USA

Corresponding author:

Abigail Hunter, school of mechanical Engineering, Purdue university, West Lafayette, In 47906, USA

Email: hunter/@purdue.edu

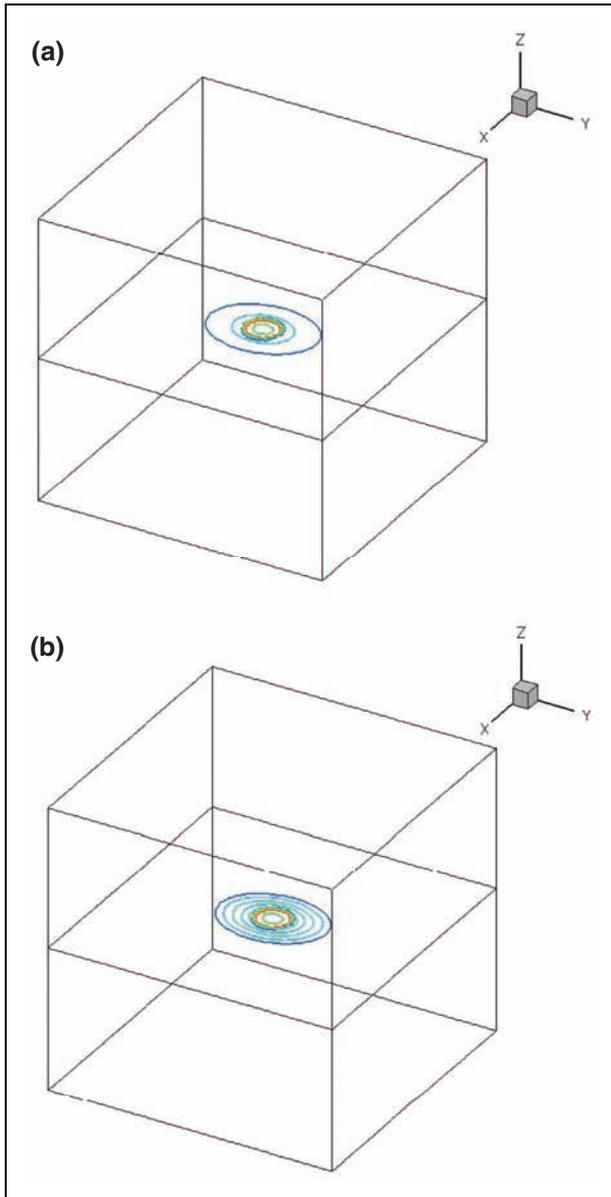


Figure 1. A 128^3 three-dimensional cube showing a single dislocation expanding under an applied shear stress at (a) 100 and (b) 300 time steps.

evolution of dislocation lines including their motion, interaction, and multiplication (Kubin and Canova, 1992; Devincere and Roberts, 1996; Zbib et al., 1998; Bulatov et al., 2006). The alternative approach follows the evolution of the dislocation density based on evolution equations derived from dislocation interactions (Groma, 1997; El-Azab, 2000; Zaiser et al., 2001; Limkumnerd and Sethna, 2006). Even though these models do not track individual dislocation lines, the non-local interactions are accounted for and used to derive the evolution equations for the dislocation density.

On the other hand, phase field dislocation models constitute a new approach in which plasticity is described in terms of the strain on each crystallographic slip system, and at the same time may resolve scales such that individual dislocations can be followed during simulations

(Finel and Rodney, 2000; Wang et al., 2001; Koslowski et al., 2002). A great advantage of phase field dislocation models is that the evolution of the dislocation fields follows from minimization of the energy of the dislocation ensemble and their interaction with other crystalline defects. Once the mechanisms of interaction of dislocations with defects and their energetics is described, these mechanisms can be integrated into the phase field description.

Some examples of successful integration of these processes are pinning of dislocations at obstacles (Koslowski et al., 2004b,a), interaction of dislocations with interfaces in thin films (Wang et al., 2003; Koslowski, 2007; Hunter and Koslowski, 2008), polycrystals (Wang et al., 2001), dislocation structures in grain boundaries (Koslowski and Ortiz, 2004), and creep (Sullivan et al., 2009). In the present paper we describe an extension of the two-dimensional phase field dislocation model developed by Koslowski et al. (2002) to a three-dimensional phase field dislocation dynamics (PFDD) model with capabilities to represent all 12 slip systems present in face center cubic (fcc) crystals. In addition, the parallelization of this numerical tool and its performance is studied over several architectures including Infiniband, GigE, and SiCortex interconnects.

The following section introduces the PFDD model, including the energetics used to describe the evolution of the dislocations. Section 3 describes the numerical implementation and parallelization of the PFDD model. Section 4 presents results that demonstrate the scalability and performance of the code over different architectures and problems sizes. In addition, Section 4 discusses the impact of different architectures and problem sizes on run times, and shows how this data can be used to predict run times for large simulations on massively parallel computers.

2 3D PFDD

Plastic deformation of crystalline materials is driven by the nucleation and motion of dislocations. Dislocations are line defects present in crystalline materials in high densities (10^{12} – 10^{17} m^{-2}) and are characterized by their slip system given by a slip plane and a slip direction. The PFDD model was developed to predict the evolution of dislocations in fcc materials and, therefore, includes 12 slip systems. In the phase field model, dislocations in each slip system, α , are represented by means of a scalar function, $\xi^\alpha(x)$. This function is scalar valued and each integer jump represents a dislocation line. This is illustrated in Figure 2, which shows a single dislocation loop expanding and overcoming four surrounding obstacles. Initially, an obstacle will pin the dislocation until a large enough stress builds up allowing the dislocation to overcome and move past the obstacle.

The energy of the system can be written as the sum of two terms Koslowski et al. (2002):

$$E = E^{int} + E^{ext}. \quad (1)$$

The first term represents elastic dislocation–dislocation interactions and the second describes the interaction of an

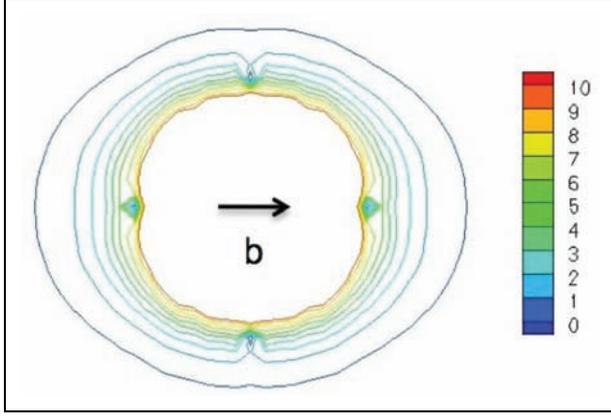


Figure 2. Two-dimensional slice of a simulation showing dislocations interacting with four obstacles. The legend shows that each integer jumps corresponds to a dislocation line.

external stress with the dislocations. The elastic dislocation–dislocation interaction is (Koslowski et al., 2002):

$$E^{\text{int}} = \frac{1}{2} \int \hat{A}_{mnuv}(\mathbf{k}) \hat{\beta}_{mn}^{\text{P}}(\mathbf{k}) \hat{\beta}_{uv}^{\text{P}*}(\mathbf{k}) \frac{d^3 \mathbf{k}}{(2\pi)^3}, \quad (2)$$

where, here and subsequently, a superposed $\hat{\cdot}$ denotes the Fourier transform of a function, \int denotes the principal value of the integral and

$$\hat{A}_{mnuv}(\mathbf{k}) = c_{mnuv} - c_{kluv} c_{ijmn} \hat{\mathbf{G}}_{ki}(\mathbf{k}) k_j k_l. \quad (3)$$

In this expression, c_{mnuv} is the tensor of elastic constants, $\mathbf{G}(\mathbf{x})$ is the Green’s tensor of linear elasticity, and β^{P} is the plastic distortion (Mura, 1987), which can be described as follows:

$$\beta_{ij}^{\text{P}}(\mathbf{x}) = \sum_{\alpha=1}^N \sum_{n_{\alpha}=-\infty}^{\infty} \zeta_{n_{\alpha}}^{\alpha}(\mathbf{x}) \delta_{n_{\alpha}} m_i^{\alpha} b_j^{\alpha}, \quad (4)$$

where it is assumed that plastic slip is confined to parallel slip planes in a crystal structure. In the previous equation, N is the number of slip systems (12 in a fcc material), α represents the slip plane family determined by the direction of the Burgers vector, \mathbf{b}^{α} , and the slip plane normal, \mathbf{m}^{α} , and $\delta_{n_{\alpha}}$ is a Dirac distribution supported on the slip plane, n_{α} . In Equation (4), the sum of the slip, $\zeta_{n_{\alpha}}^{\alpha}(\mathbf{x})$, over all of the planes in a family, α , can be replaced by $w^{-1} \zeta^{\alpha}(\mathbf{x})$, where w is the interplanar distance:

$$\beta_{ij}^{\text{P}}(\mathbf{x}) = \frac{b}{w} \sum_{\alpha=1}^N \zeta^{\alpha}(\mathbf{x}) m_i^{\alpha} s_j^{\alpha}. \quad (5)$$

Here, \mathbf{s}^{α} is the slip direction of the Burgers vector, \mathbf{b}^{α} , and $w^{-1} \zeta^{\alpha}(\mathbf{x})$ represents a 3D density of slip. Substituting Equation (5) into Equation (2):

$$E^{\text{int}} = \int \sum_{\alpha=1}^N \sum_{\alpha'=1}^N \hat{B}_{\alpha\alpha'}(\mathbf{k}) \zeta^{\alpha'}(\mathbf{k}) \frac{d^3 \mathbf{k}}{(2\pi)^3}, \quad (6)$$

where

$$\hat{B}_{\alpha\alpha'}(\mathbf{k}) = \frac{b^2}{w^2} \hat{A}_{mnuv}(\mathbf{k}) m_m^{\alpha} s_n^{\alpha} m_u^{\alpha'} s_v^{\alpha'}. \quad (7)$$

The energy of interaction of dislocations with an external applied stress can be written as (Koslowski et al., 2002):

$$E^{\text{ext}} = \int \sigma_{ij}^{\text{app}} \beta_{ij}^{\text{P}} d^3 x, \quad (8)$$

where σ_{ij}^{app} is the externally applied stress. Additional terms can be incorporated into the energy equation to account for the core energy of the dislocations, the interaction of dislocations with other defects and the formation of partial dislocations Wang et al. (2001); Koslowski et al. (2002); Shen and Wang (2004).

The displacement jump corresponding to one dislocation sliding in the slip plane should be an integer multiple of the Burgers vector. Therefore, the phase field, $\zeta(\mathbf{x})$, should take integer values. There are several potential forms satisfy this condition, in the past we have used a piecewise quadratic potential Koslowski et al. (2002). Here we enforce this penalty with a sinusoidal function Wang et al. (2001); Hirth and Lothe (1968).

$$E^{\text{core}} = \int \sum_{\alpha=1}^N A_1 \sin^2(\pi \zeta^{\alpha}(\mathbf{x})) d^3 x, \quad (9)$$

where A is the unstable stacking fault energy and can be obtained from atomistic simulations.

The time evolution of the dislocation ensemble follows the Ginzburg–Landau equation Wang et al. (2001); Koslowski (2007):

$$\frac{\partial \zeta^{\alpha}(\mathbf{x}, t)}{\partial t} = -L \frac{\delta E}{\delta \zeta^{\alpha}(\mathbf{x}, t)}, \quad (10)$$

where the kinetic coefficient, L , determines the velocity of the dislocations and $E = E^{\text{int}} + E^{\text{core}} + E^{\text{ext}}$. Through application of Equations (6)–(9), the term on the right-hand side of Equation (10) can be written as

$$\frac{\delta E}{\delta \zeta^{\alpha}(\mathbf{x}, t)} = \int \sum_{\alpha'=1}^N \hat{B}_{\alpha\alpha'}(\mathbf{k}) \zeta^{\alpha'}(\mathbf{k}) e^{i\mathbf{k}\cdot\mathbf{x}} \frac{d^3 \mathbf{k}}{(2\pi)^3} + \tau^{\alpha}(\mathbf{x}) + A\pi \sin(2\pi \zeta^{\alpha}(\mathbf{x})), \quad (11)$$

where $\tau^{\alpha} = \sigma_{ij} s_i^{\alpha} m_j^{\alpha}$ is the resolved shear stress in the slip plane, α .

The PFDD model is used to study the evolution of complex dislocation structures and the resulting macroscopic stress–strain response. As an example, we carry out simulations of dislocations evolving under an external applied shear stress in a confined geometry.

Figure 3 shows the resulting dislocation configurations. The walls perpendicular to the x direction are impenetrable to dislocations and represent the interface in a passivated thin film. Figure 3(a) and (b) show dislocation pile ups against these impenetrable walls. We consider film thicknesses of 8, 16, and 32 nm and we use Nickel material constants, $b = 0.249$ nm, $\mu = 75$ GPa, $\nu = 0.2$ and $A = 0.6 \text{ J m}^{-2}$ (Lee et al., 2010). The computational cells are 32^3 , 64^3 , and 128^3 , respectively. Simulations were performed with the code running on the Steele cluster on up to 64 processors for up to 8,000 time steps at 20 different

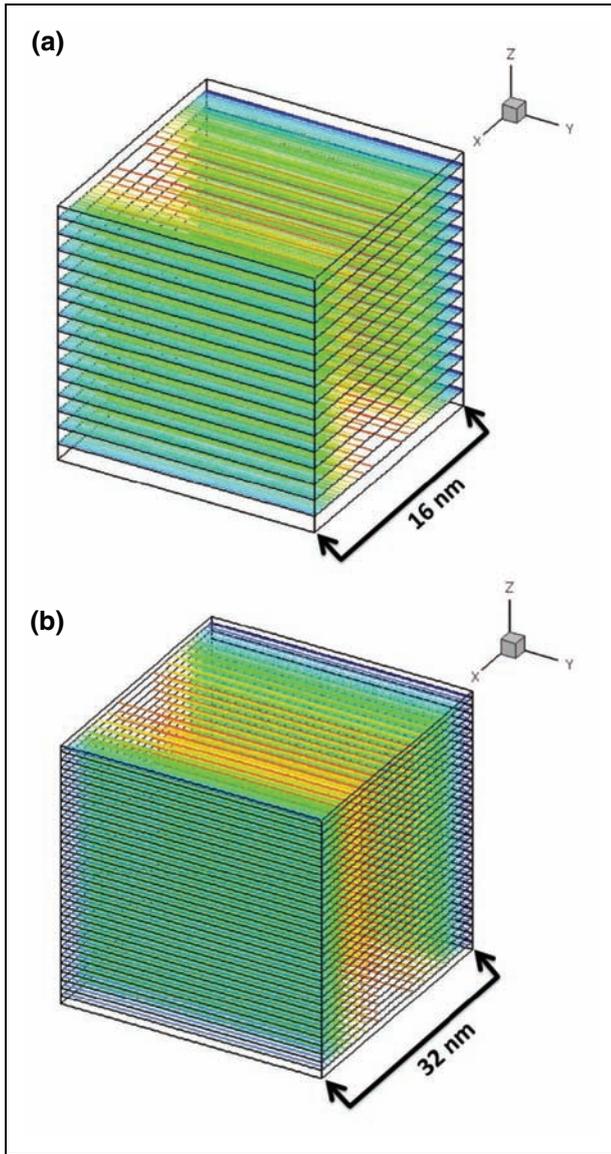


Figure 3. Final dislocation configurations for two different layer thicknesses for a Nickel passivated layer.

applied shear stresses to simulate quasi-static monotonic loading.

From the stress–strain curves we are able to compute yield stress dependency on the thickness. The yield stress is computed using the 0.2% offset yield point. Figure 4 shows the yield stress dependency on the layer thicknesses. As expected the yield stress increases as the thickness decreases following a Hall–Petch relation Hall (1951); Petch (1953):

$$\sigma_y \sim h^{-0.6}, \quad (12)$$

where σ_y is the yield stress and h is the film thickness. The exponent is in very good agreement with experimental observation. On the other hand, the values of the yield stress are affected by the initial dislocation density and the microstructure (Hunter and Koslowski, 2010). Parallelization becomes more important for these applications as dislocation density and characteristic size becomes larger and more complex geometries are represented.

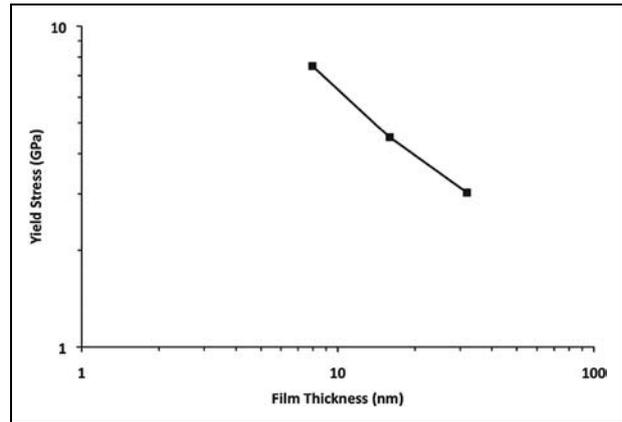


Figure 4. Yield stress dependency on characteristic length for a thin layer subject to shear loading.

3 PFDD Numerical Implementation

This section gives an outline of the parallel algorithm used to implement the PFDD approach presented in the previous section. A flowchart outlining the serial implementation of the PFDD approach is shown in Figure 5.

Initial preprocessing steps include applying loading conditions to the system, setting initial conditions, and calculation of the dislocation–dislocation interaction matrix given by Equation (7). The dislocation–dislocation interaction matrix is one of the most costly calculations completed by the algorithm. However, one advantage is that this matrix is calculated only once during each simulation. This matrix is the size of the grid multiplied by the number of activated slip systems squared. For even a small system of 32^3 and 12 slip systems, the dislocation–dislocation interaction matrix will take up 36 MB of memory. As larger and larger simulations are needed, the size and memory required to calculate this matrix quickly grows and the need for parallelization can be seen more directly.

The following step in the algorithm is to solve the Ginzburg–Landau equation, Equation (10). The right-hand side of Equation (10) is computed as an integral in the frequency domain, therefore we calculate the Fourier transform of each phase field using a fast Fourier transform (FFT) algorithm. The FFT is a very effective approach for computing the discrete Fourier transform in $O(N \log N)$ operations (Cooley and Tukey, 1965). Several iterations are completed until the system deforms in such a way that the internal and external energies are balanced and convergence is reached.

To obtain the discrete Fourier transform of our data we use the multidimensional, complex-to-complex FFTW 2.1.5 subroutine (Frigo and Johnson, 1998). FFTW 2.1.5 is a free, open-source parallel C subroutine library that computes the discrete Fourier transform in one or more dimensions. FFTW uses a 1D decomposition to slice the 3D cube in the x -direction, so each processor is given a slab/portion of the data. In turn, the rest of the code is parallelized to accommodate this kind of decomposition for consistency. In addition, the FFTW 2.1.5 subroutine allows for data transposition. This can have a significant impact on

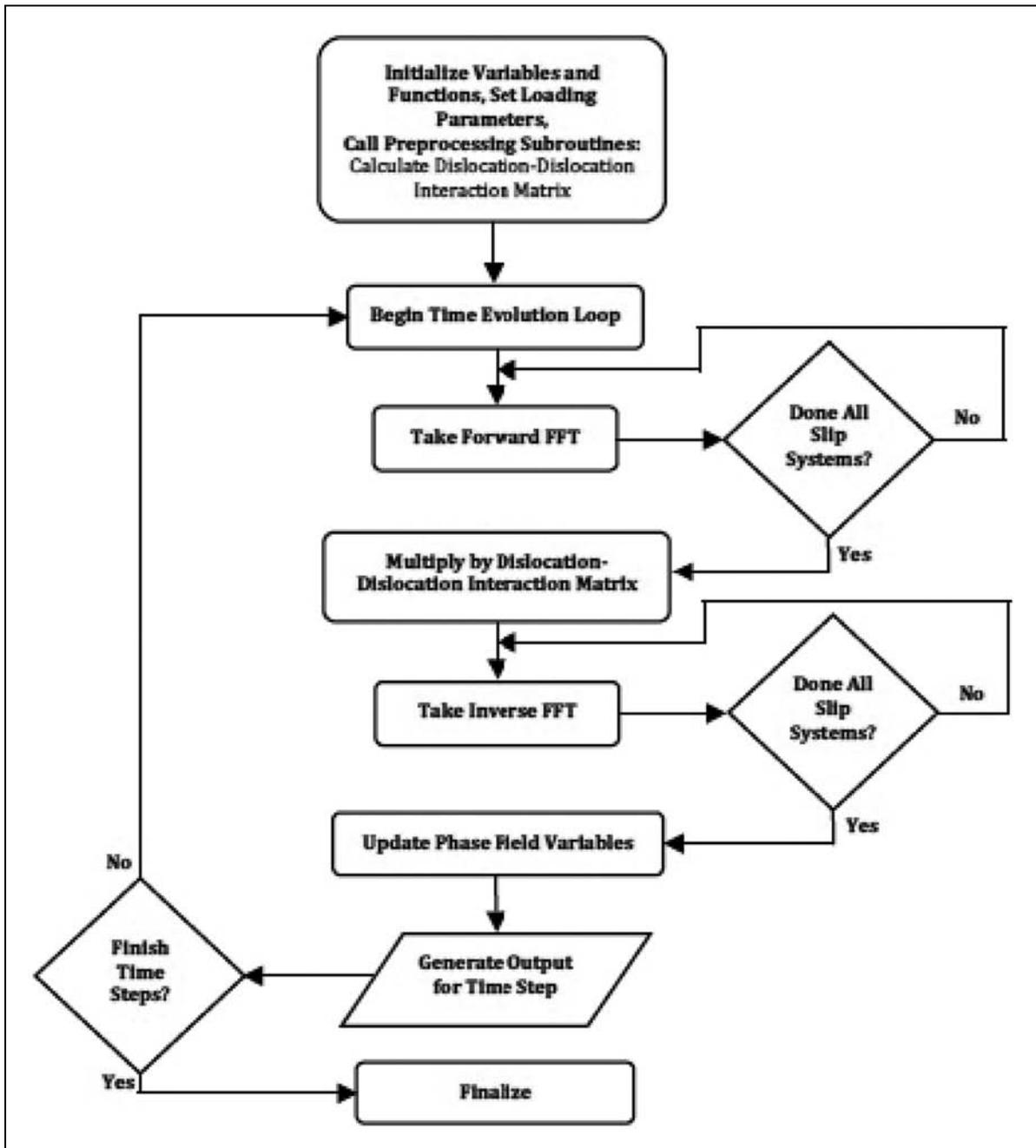


Figure 5. Flow chart showing major components of the PFDD code.

the amount of communication needed to complete the Fourier transform. This aspect of the subroutine performance was taken into consideration and analyzed. Further discussion can be found in Section 4.4.1. The majority of the parallelization is achieved using local variables in the loop structures within the algorithm. Consequently, very little MPI communication is added outside what is required for the forward and inverse FFTs, all of which is contained within the FFTW subroutine. Figure 6 shows the 1D decomposition for both normal order and transposed order FFTs.

4 Performance Analysis

The performance of the 3D PFDD model was analyzed on three different interconnects including GigE, Infiniband,

and SiCortex. Runs for the GigE and Infiniband interconnects were completed on the Steele Cluster at Purdue University with Dual 2.33 GHz Quad Core Intel E5410 processors with eight cores per node. The SiCortex runs were performed on the Moffett Cluster which is also at Purdue University and has 633 MHz SiCortex 5832 processors with six cores per node. Each run simulated a single dislocation loop expanding on a single slip plane for 100 time steps.

In the following section, results detailing the parallel 3D PFDD model's performance are presented. This includes investigation of the impact of the interconnection network on the algorithm's performance, scalability analysis, discussion of the efficiency of the major components of the code, and the MPI overhead measured using the MPI

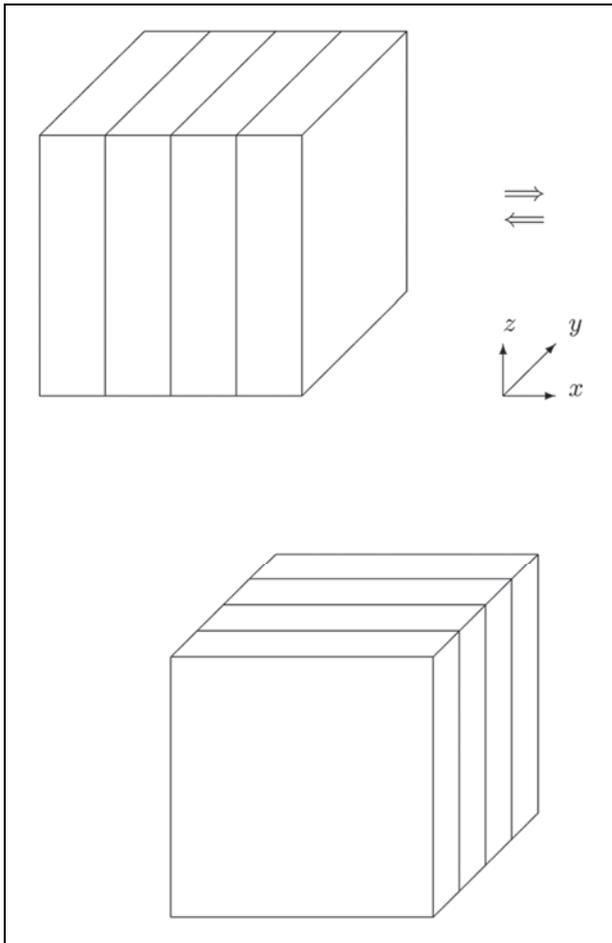


Figure 6. One-dimensional domain decomposition used in FFTW: normal order (left) and transposed order (right).

Profiler (mpiP) software. In addition, a performance model was applied to the data to predict performance for various cases.

4.1 Impact of the Interconnection Network on Performance

The efficiency of each interconnect is evaluated using two primary criteria: the total run time and scalability over increasing data sizes. Not only is it desired for the code to attain fast run times, but future simulations will require larger data sets and a greater number of time steps in order to analyze more realistic and complex systems. Therefore, the capacity of the parallel implementation to efficiently handle large amounts of computation on large numbers of processors is of great interest. The hardware used to run the code can vastly affect the run time and scalability efficiency. This is obvious from the results displayed in Figure 7(a) and (b), which show run time and scalability data for the different interconnects, respectively.

Figure 7(a) presents run times for various dimension sizes on all three interconnects for three different grid sizes (64^3 , 128^3 , and 256^3). Overall, the best runs times came from runs done using the Infiniband interconnect. The GigE runs produced faster run times only on single node

runs (using up to eight processors). This may be an indication of more efficient memory bandwidth usage in the GigE interconnect. However, for large data sets, run times on a single node are considerably higher than run times on multiple nodes. This is true in the case of both the Infiniband and SiCortex interconnects. The GigE interconnect does not show a decreasing slope as the number of processors increases. In fact, in all of the runs, this curve tends to level out early on in comparison to the other two interconnects. This is an indication of poor parallelization on this interconnect.

Simulations completed using the SiCortex interconnect have longer run times in every case. The SiCortex interconnect has a much slower processing time when compared with the Infiniband and GigE architectures, so this result is expected. Conversely, the SiCortex interconnect has fast communication between processors, which results in much better scalability. This is shown in Figure 7(b). This figure compares the speed-up efficiency of the three interconnects over several data sizes. In addition, these results are all compared with an ideal speed-up case, which helps to understand the true efficiency of the parallelization implementation. The SiCortex interconnect shows the best parallelization out of the three interconnects as expected. The Infiniband interconnect also shows moderately good parallelization, with the scalability increasing as the data set size increases. The results from the GigE interconnect show very poor scalability as expected.

After taking into account both run time and scalability, the Infiniband interconnect produced the best results, and the GigE interconnect produced the worst. General trends include better scalability for larger data sets, and lower run times on larger numbers of processors. Both of these trends are desired results, and show that through parallelization of the 3D PFDD model, better computational efficiency can be achieved. Results produced and presented for the remainder of this performance analysis are found using the Infiniband interconnect (unless otherwise stated).

4.1.1 Multidimensional FFT. The 3D PFDD model performs both a forward and inverse multidimensional FFT over all slip systems every time step using the complex-to-complex FFTW function (Frigo and Johnson, 1997). This function accounts for most of the communication time in the code via calls to MPI Alltoall. Therefore, it is important to analyze how the FFTW function behaves within the code. Figure 8 shows FFTW run times over multiple processors, and over the Infiniband and GigE networks on the Purdue Steele cluster for a 256^3 data set. In addition, the total run times of the 3D PFDD program are shown.

Within a Steele cluster node, the forward and inverse FFTW Infiniband and GigE curves are identical from one to eight cores, as expected, since the communication is intra-node and does not use the interconnect. Intra-node communication in FFTW is carried over the shared memory bandwidth, and not the Infiniband or GigE networks. Bandwidth is the ability to feed data to the cores from the local memory. Generally, memory bandwidth saturates at four cores and will have a more noticeable impact on a

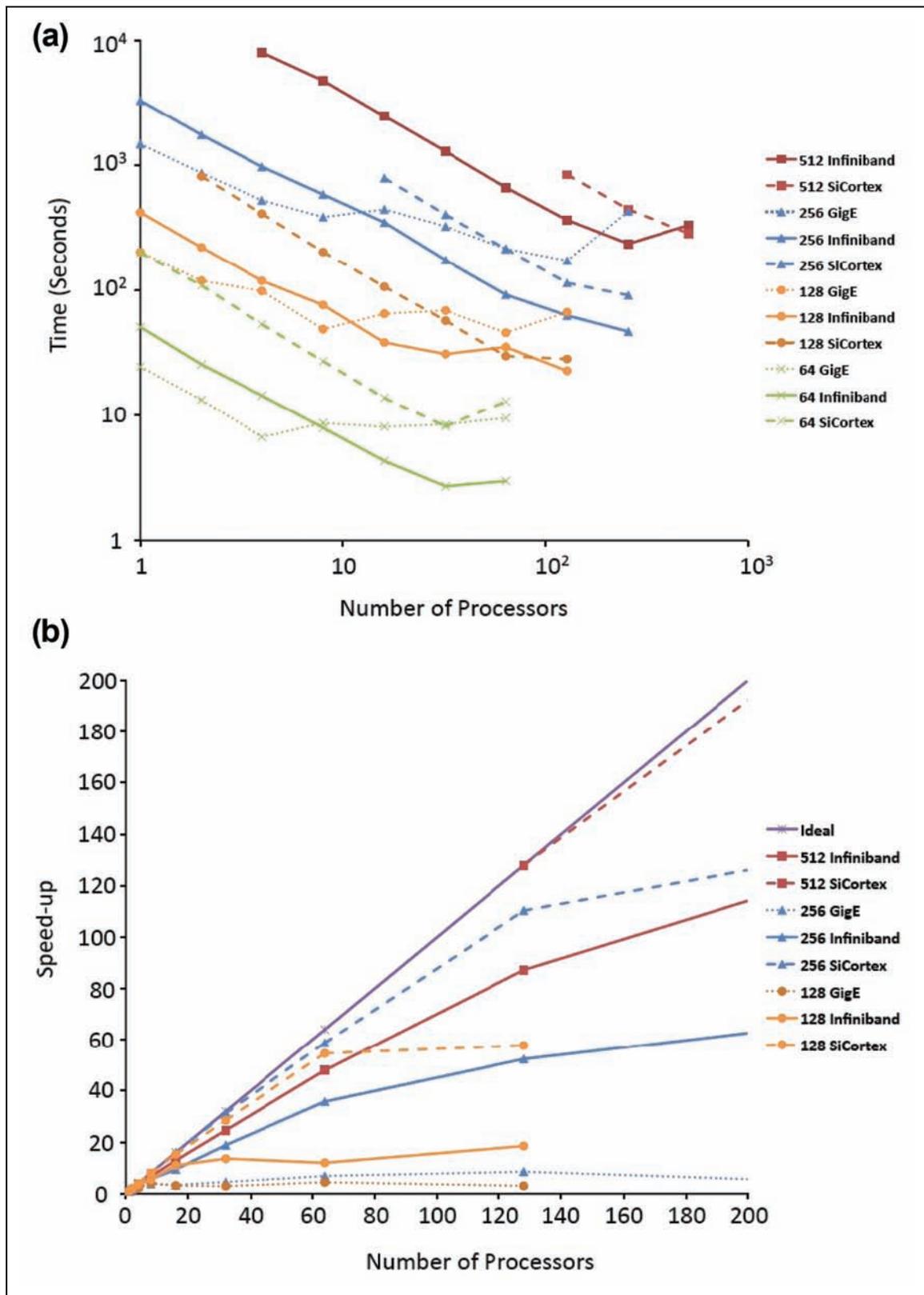


Figure 7. Total run times (a) and speed-up (b) of the parallelized code on GigE, Infiniband, and SiCortex interconnects, for three different problem sizes.

single node than on multiple nodes. In Figure 8, the slope of the run time curves begins to level off between four and eight cores. This is representative of the bandwidth saturation at four cores.

The forward and inverse FFTW GigE curves exhibit a positive slope going from 8 to 16 cores, showing that communication across nodes via GigE is slower than shared memory communication within node. Between 16 and

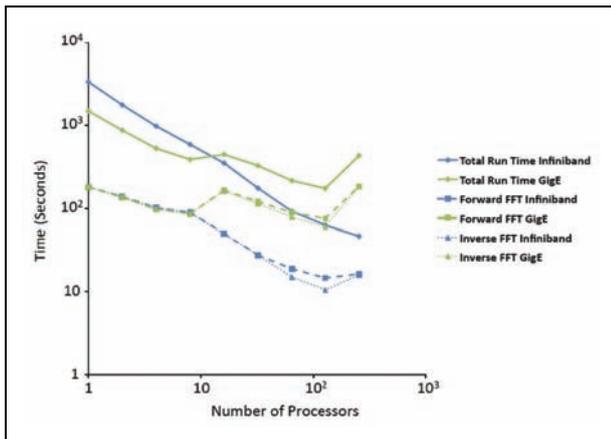


Figure 8. Run time results for forward and inverse calls to FFTW for the GigE and Infiniband interconnects.

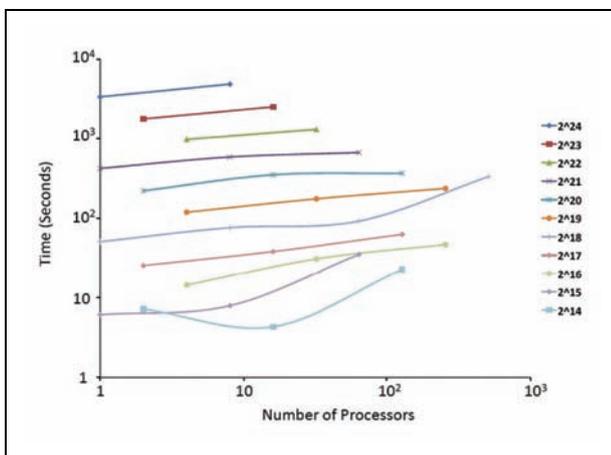


Figure 9. Weak scaling of code for different data sizes.

128 cores, the slope of the curve remains negative and relatively constant showing that this is the range of good FFTW scalability for the GigE interconnect. Between 128 and 256 cores, the slope of the curve turns positive. On the Infiniband interconnect, the slopes of the forward and inverse FFTW curves sharply decrease starting at eight cores. Unlike the GigE interconnect, communication across nodes via Infiniband interconnect is faster than the shared memory within a node. There is an increase in slope between 128 and 256 processors, although it is not as drastic a change as seen with the GigE network. The network makes such a large impact on the FFTW run times because FFTW is a MPI communication intensive function, and requires a fast interconnect in order to achieve efficient run times. In addition, this can be seen by the considerably faster run times observed on the Infiniband interconnect.

4.2 Results for Weak Scaling

Scalability was discussed above for several different architectures. For a clear picture of how the code scales, its components are analyzed separately, in particular the FFT component which accounts for over 50% of the total run time, and where most of the MPI communication occurs.

This leads to questions about how communication time scales over large data sets and number of processors in comparison with computation time.

A rough idea of the overall scalability of the code may be seen in Figure 9, which shows weak scaling for increasing data size per processor. In weak scaling, the problem size per processor is kept constant as the number of processors increases, while in strong scaling the problem size remains constant as the number of processors is increased. Most of the run time data presented in this paper uses strong scaling unless otherwise stated, as in Figure 9.

In Figure 9, curves showing small data sizes per processor exhibit a positive slope. For data sizes equal to or greater than 2^{19} , the curves become nearly horizontal, which corresponds to nearly ideal scaling. Thus, the program scales well for large data sizes. This is expected because the computation time in large data sets is far greater than the MPI communication time. For small data sizes, scalability is limited by the MPI communication in the parallel implementation, the majority of which results from the FFTs.

4.3 Scalability of Major Code Components

A run time breakdown of the components within the parallel 3D PFDD algorithm for two large data sets are shown in Figure 10. These timings were observed on the Lawrence Livermore National Laboratory (LLNL) Hera computing cluster, which has an Infiniband interconnect with 16 AMD Opteron quad core processors per node running at 2.3 GHz.

Comparison of the total run time curves in Figure 10(a) and (b) shows that between 256 cores and 512 cores the curve in Figure 10(b) follows closely the ideal curve while the curve in Figure 10(a) deviates from the ideal curve. Thus, the 3D PFDD algorithm is more scalable for the larger data set ($2,048^3$) than for the smaller data set ($1,024^3$). The 3D PFDD model is scalable up to 2,048 cores for the $2,048^3$ size data set, whereas for the $1,024^3$ size data set scalability no longer holds at some point between 512 and 1,024 cores.

The components that take the most time include the calls to FFTW and calculation of the dislocation–dislocation interaction matrix. The dislocation–dislocation interaction matrix shows nearly ideal parallelization as the number of processors increases. This is because the subroutine that calculates this matrix requires little to no MPI communication between processors. Since it is entirely based on arithmetic, the parallelization of this portion of the parallel implementation is exceptional.

For small numbers of cores, the interaction matrix calculation is the largest component and decreases linearly in the log–log plot as the number of cores increases. Conversely, the FFTW time is small when the number of cores is small, decreases with increasing processors, and finally turns positive near the maximum number of processors becoming greater than the calculation time for the dislocation–dislocation interaction matrix. This trend is very clear in Figure 10(a). In Figure 10(b), the FFTW time has not increased significantly at 2,048 cores and while the total run time shows poor scalability, it is still decreasing with increasing number of cores.

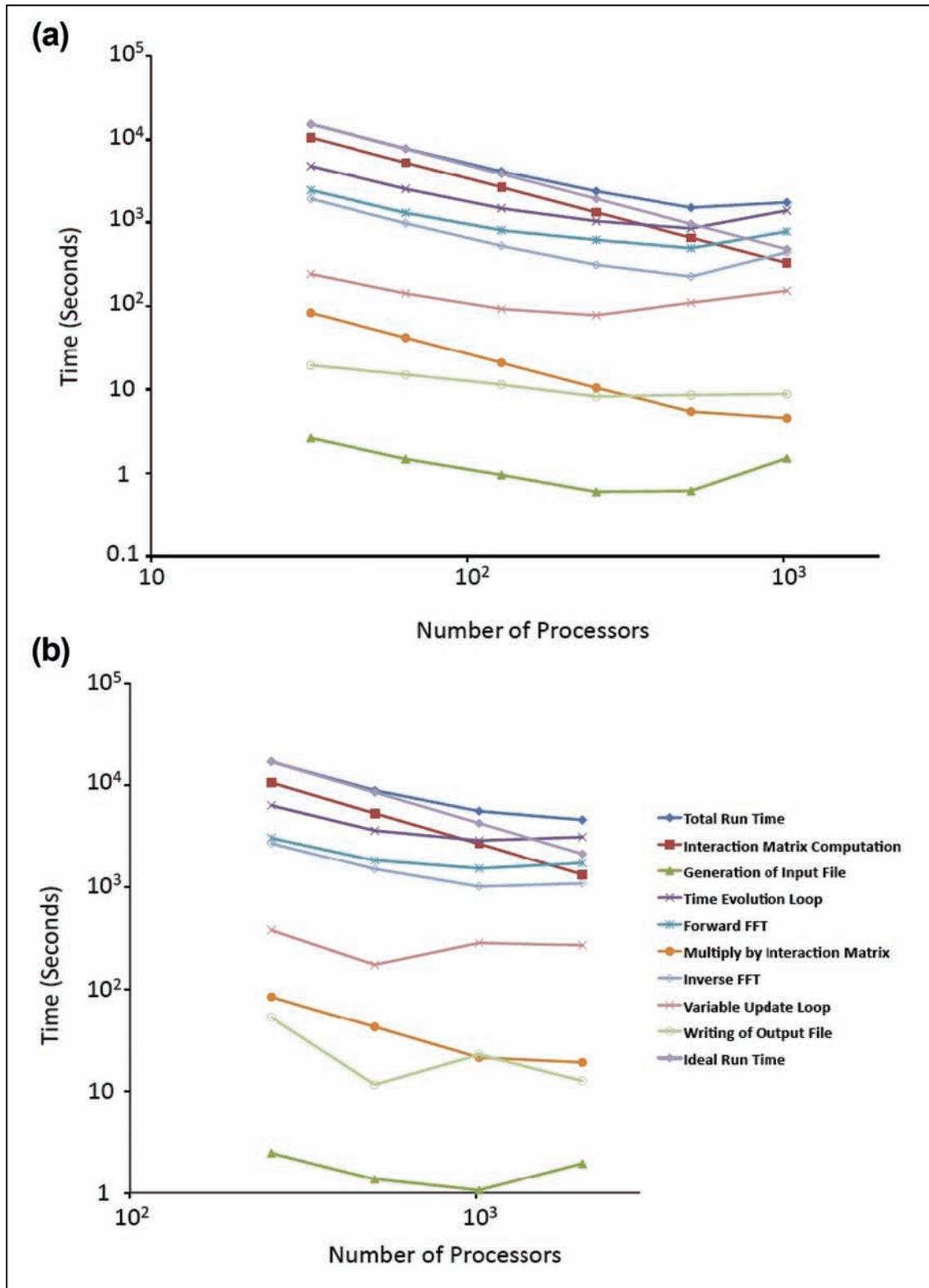


Figure 10. Run time breakdown of all components of the parallel code for (a) $1,024^3$ and (b) $2,048^3$.

These results show that not only does the model's overall scalability improve with increasing data sizes, but benefits can be seen in the individual components of the code. This would lead to the conclusion that the 3D PFDD

algorithm will continue to scale well for data larger than $2,048^3$ on more than 2,048 cores. This is important for simulations of more realistic devices and components that may require large, complex data sets.

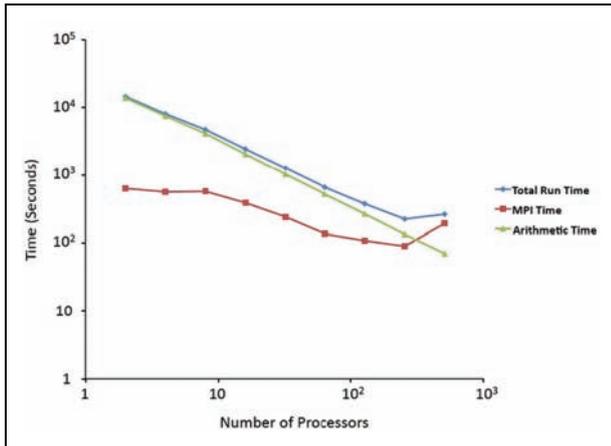


Figure 11. Comparison of MPI communication, arithmetic, and total run times. The MPI time was measured using mpiP.

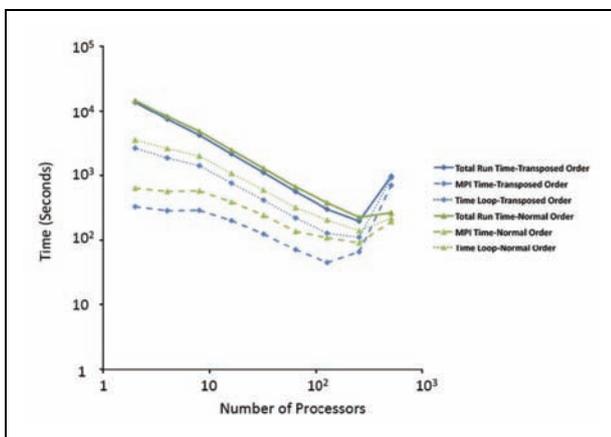


Figure 12. Run time results comparing normal order and transposed order FFTW calls.

4.4 MPI Communication

MPI communication was monitored using the computing tool mpiP (Vetter and Chambreau, 2010). mpiP reports the total time the code spent completing all MPI communication commands along with a breakdown of which commands were called and how much time each took to complete. mpiP confirmed that the majority of the communication time is due to Alltoall calls within the FFTW function. From the total run time and the communication time, the arithmetic time can be calculated. Figure 11 shows the total, communication and arithmetic times and scalability can be compared.

Data collected using the mpiP tool shows quantitatively that the communication is the limiting factor of the code's scalability. The computation time scales linearly as the number of processors increases. Conversely, the MPI communication time decreases only slightly with an increasing number of cores and is lowest on large numbers of processors. However, at the largest number of processors used, there is a sharp increase in the communication run time. As mentioned before, when the data in each processor is made up of only one plane of the 3D grid, the communication between processors outweighs the arithmetic time and causes both the communication and total run times to increase.

4.4.1 Normal Order versus Transposed Order FFT. The FFTW function can return the data in either a normal order (the same as the way it was read into the function) or a transposed order form. The transposed order requires half as many MPI Alltoall calls, which cuts the communication time in half. However, the transpose option requires modification of the code, specifically in the way the dislocation–dislocation interaction matrix is created. The run time difference between normal order and transposed order FFTW calls are compared in Figure 12.

Figure 12 compares times from three components of the code: the total 3D PFDD run time, the time taken to complete the time evolution loop, and the MPI communication time. The MPI communication time for the transpose order FFTW is half that of the normal order FFTW resulting in a smaller time evolution loop run time for the transpose order. However, the difference between the transpose and normal order total run times of the PFDD algorithm remains small. The difference in the MPI communication time is offset by the time taken for preprocessing outside of the time evolution loop, in particular the calculation of the dislocation–dislocation interaction matrix. Although this matrix requires little to no MPI communication, its large size requires a great amount of computation time relative to MPI communication time. As the number of cores increases to 512, the computation time per node decreases while latency increases and scalability is no longer valid.

It must be noted that these runs were completed for 100 time steps. Since the majority of MPI communication is contained within the FFTW subroutine which is called twice every time step, it is reasonable to expect that the amount of MPI communication will increase with increasing time steps. In addition, the dislocation–dislocation interaction matrix is calculated once at the beginning of the code and is independent of the number of time steps. As the number of time steps grows, the computation time for the dislocation–dislocation interaction matrix will remain the same while the MPI communication time grows. At large numbers of time steps, the MPI communication will dominate over this computation time and the reduction in communication time from the transposed order FFTW will be reflected in the overall run time as well as in the time evolution loop. This could be important for certain applications of the PFDD approach, such as in comparison with molecular dynamic simulations, which require large numbers of time steps for convergence.

4.5 Performance Model

All of the different factors that influence the PFDD algorithm's run time can be described by an equation of the form (Dmitruk et al., 2001)

$$T \rightarrow t_c \frac{5N^3 \log_2(N^3)}{2P} + t_a 3 \frac{N^3}{P} + t_w 2 \frac{N^3}{P} + t_s 2P, \quad (13)$$

where N represents the grid size in one dimension, P is the number of processors, t_c represents the FFT computation

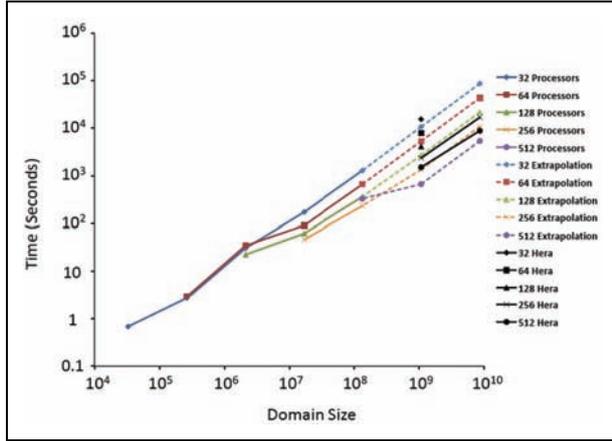


Figure 13. Actual and predicted run times calculated from the performance model.

time per word on a single processor, t_a is the time it takes for a memory-to-memory copy of a word, t_w is the time it takes for a single word to transmit between nodes, and t_s represents the start-up or latency time for a message.

The run time of a code is composed of two major components: computation time and communication time. The first term in Equation (13) corresponds to the time needed to compute the FFTs, and the second term reflects the time needed to rearrange data. These terms both correlate to the computation time of the code. The third term accounts for the time it takes to send and receive information between nodes, and the fourth term represents the latency time for a message. These last two terms represent the MPI communication time.

As the number of processors, P , increases, the first three terms of Equation (13) decrease at a rate inversely proportional to P , while the fourth term, the latency component, increases proportionally to P . For large numbers of processors, the latency component becomes larger than the sum of the first three terms, and the total run time increases. This explains the sharp increase in slope in run time curves shown in Figure 10. This is generally seen when the number of processors is at a maximum and each processor has only one slab of the 3D grid, which results in a great increase in communication between processors. This effect is reflected in the latency term in Equation (13). In addition, since the latency term correlates to MPI communication, the increase in slope with the large number of processors can be seen in FFTW run times since this function is communication intensive (see Figures 8, 11, and 12). However, the slope increase will not be seen in computation-based components, such as calculation of the dislocation–dislocation interaction matrix (see Figures 10 and 11).

Using data obtained from the runs presented above, a predictive model can be formulated to determine how long larger runs will take, and how many processors are needed to maximize efficiency. Results from runs completed on the Infiniband interconnect at Purdue University along with least squares regression were used to compute the parameters in

$$T(N, P) = 8.92 \times 10^{-7} \frac{N^3 \log_2 N^3}{P} + 2.92 \times 10^{-4} \frac{N^3}{P} - 2.64 \times 10^{-4} P \quad (14)$$

which is based on Equation (13). Equation (14) was used to predict run times for the 3D PFDD with large data sets. These predictions were then compared to runs completed on LLNL Hera computing cluster. Figure 13 shows the actual run times from Purdue’s Steele cluster and from LLNL’s Hera cluster as well as predicted times calculated from Equation (14). The predictions provide a good estimation of total run times.

5 Conclusions

We have presented a 3D phase field dislocation dynamics model that enables efficient simulations of plastic deformation in crystalline materials. The code is implemented in parallel and makes use of the FFTW library, a multidimensional FFT (Frigo and Johnson, 1997). In addition, the performance analysis shows that this parallel algorithm is portable across different architectures.

We have performed numerical simulations on up to 2,048 cores and the algorithm shows high computational efficiency and scalability in large data sets. The performance model developed allows us to estimate with very good accuracy run times for large data sets as well as the number of processors needed to achieve high efficiency. This new algorithm will increase the capacity to accomplish more advanced simulations of plastic deformation with more realistic microstructures in domain sizes not attainable with serial dislocation dynamics implementations. This will greatly aid the study of intricate dislocation structures and their interactions with grain boundaries, voids, and interfaces.

Acknowledgements

This material is based upon work supported by the Department of Energy, National Nuclear Security Administration under Award Number DE-FC52-08NA28617 and Department of Energy Basic Energy Sciences under award DE-FG02-07ER46398.

References

- Budrovik, Z., Van Swygenhoven, H., Van Petegem, S., and Schmitt, B. (2004). Plastic deformation with reversible peak broadening in nanocrystalline nickel. *Science* **304**: 273–276.
- Bulatov, V. V. and Cai, W. (2006). *Computer Simulations of Dislocations*. Oxford University Press, Oxford.
- Bulatov, V. V., Hsiung, L. L., Tang, M., Arsenlis, A., Bartelt, M. C., Cai, W., Florando, J. N., Hiratani, M., Rhee, M., Hommes, G., Pierce, T. G., and de la Rubia, T. D. (2006). Dislocation multi-junctions and strain hardening. *Nature* **44**: 1174–1178.
- Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Math. Comput.* **19**(90): 297–301.

- Devincre, B. and Roberts, S. G. (1996). Three-dimensional simulation of dislocation-crack interactions in b.c.c. metals at the mesoscopic scale. *Acta Materialia* **44**(7): 2891–2900.
- Dimiduk, D. M., Koslowski, M. and LeSar, R. (2006). Preface to the viewpoint set on: Statistical mechanics and coarse graining of dislocation behavior for continuum plasticity. *Scripta Materialia* **54**(5): 701–704.
- Dmitruk, P., Wang, L.-P., Matthaeus, W. H., Zhangc, R., and Seckela, D. (2001). Scalable parallel FFT for spectral simulations on a Beowulf cluster. *Parallel Comput.* **27**: 1921–1936.
- El-Azab, A. (2000). Statistical mechanics treatment of the evolution of dislocation distributions in single crystals. *Phys. Rev. B* **61**(18): 11956–11966.
- Finel, A. and Rodney, D. (2000). Phase field methods and dislocations. In MRS Fall Meeting, Boston, MA.
- Frigo, M. and Johnson, S. G. (1997). The Fastest Fourier Transform in the West. Technical Report MIT-LCS-TR-728, Laboratory for Computing Sciences, MIT, Cambridge, MA.
- Frigo, M. and Johnson, S. G. (1998). FFTW: An adaptive software architecture for the FFT. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1381–1384.
- Groma, I. (1997). Link between the microscopic and mesoscopic length-scale description of the collective behavior of dislocations. *Phys. Rev. B* **56**(10): 5807–5813.
- Hall, E. O. (1951). The deformation and ageing of mild steel. *Proc. Phys. Soc. London Sect.* **64**(381B): 747–753.
- Hirth, J. P. and Lothe, J. (1968). *Theory of Dislocations*. McGraw-Hill, New York.
- Hunter, A. and Koslowski, M. (2008). Direct calculation of material parameters for gradient plasticity. *J. Mech. Phys. Solids* **56**(11): 3181–3190.
- Hunter, A. and Koslowski, M. (2010). Influence of the grain size distribution on plastic deformation of polycrystalline Ni. In preparation.
- Koslowski, M. (2007). Scaling laws in plastic deformation. *Philos. Mag.* **87**(8–9): 1175–1184.
- Koslowski, M., Cuitiño, A., and Ortiz, M. (2002). A phase-field theory of dislocations dynamics, strain hardening and hysteresis in ductile single crystals. *J. Mech. Phys. Solids* **50**: 2597–2635.
- Koslowski, M., LeSar, R. and Robb, T. (2004a). Dislocation structures and the deformation of materials. *Phys. Rev. Lett.* **93**(26): 265503.
- Koslowski, M., LeSar, R. and Thomson, R. (2004b). Avalanches and scaling in plastic deformation. *Phys. Rev. Lett.* **93**(12): 125502.
- Koslowski, M. and Ortiz, M. (2004). A multi-phase field model of planar dislocation networks. *Modell. Sim. Mater. Sci. Eng.* **12**(6): 1087–1097.
- Kubin, L. P. and Canova, G. (1992). The modelling of dislocation patterns. *Scripta Metal. Material.* **27**: 957–962.
- Lee, D. W., Kim, H., Strachan, A. and Koslowski, M. (2010). Dislocation mobility informed from atomistic simulations to the mesoscale. In preparation.
- Limkumnerd, S. and Sethna, J. P. (2006). Mesoscale theory of grains and cells: crystal plasticity and coarsening. *Phys. Rev. Lett.* **96**: 095503.
- Mura, T. (1987). *Micromechanics of Defects in Solids*. Kluwer Academic Publishers, Dordrecht.
- Petch, N. J. (1953). Cleavage strength of polycrystals. *J. Iron Steel Inst.* **174**(1): 25–28.
- Rajagopalan, J., Han, J. and Saif, T. A. (2007). Plastic deformation recovery in free standing nanocrystalline aluminum and gold films. *Science* **315**: 1831–1834.
- Rebeiz, G. (2003). *RF MEMS, Theory, Design and Technology*. John Wiley and Sons, Inc., Hoboken, NJ.
- Shen, C. and Wang, Y. (2004). Incorporation of γ -surface to phase field model of dislocations: simulating dislocation dissociation in fcc crystals. *Acta Material.* **52**(3): 683–691.
- Sullivan, T., Koslowski, M., Theil, F. and Ortiz, M. (2009). On the behavior of dissipative systems in contact with a heat bath: Application to andrade creep. *J. Mech. Phys. Solids* **57**(7): 1058–1077.
- Uchic, M. D., Dimiduk, D. M., Florando, J. N. and Nix, W. D. (2004). Sample dimensions influence strength and crystal plasticity. *Science* **305**(5686): 986–989.
- Vetter, J. and Chambreau, C. (2010). mpiP: Lightweight, scalable MPI profiling. <http://mpip.sourceforge.net/>
- Walraven, J. A. (2003). Failure mechanisms in MEMS. In IEEE International Test Conference, Vol. 1, pp. 828–833.
- Wang, Y., Jin, Y., Cuitiño, A. and Khachaturyan, A. (2001). Phase field microelasticity theory and modeling of multiple dislocation dynamics. *Appl. Phys. Lett.* **78**(16): 2324–2326.
- Wang, Y., Jin, Y. and Khachaturyan, A. (2003). Phase field microelasticity modeling of dislocation dynamics near free surface and in heteroepitaxial thin films. *Acta Material.* **51**: 4209–4223.
- Zaiser, M., Miguel, M. C. and Groma, I. (2001). Statistical dynamics of dislocation systems: The influence of dislocation-dislocation correlations. *Phys. Rev. B* **64**(22).
- Zbib, H. M., Rhee, M. and Hirth, J. P. (1998). On plastic deformation and the dynamics of 3d dislocations. *Int. J. Mech. Sci.* **40**.

Bios

Abigail Hunter is a Mechanical Engineering Ph.D. student at Purdue University. She graduated with honors from the University of Utah with a Bachelors of Science degree in Mechanical Engineering in 2006. Her research interests include computational science and engineering with a specific focus on micro- and nano-scale material phenomena.

Faisal Saied graduated from Yale University in 1990 with a Ph.D. in Computer Science. Since 2003, he has been a senior research scientist at Purdue University, in the Computing Research Institute. Prior to that he had led the Performance Engineering Group at the National Center for Supercomputing Applications, at the University of Illinois at Urbana-Champaign, from 1997 to 2003. His research interests include high-performance computing and applications, computational science and engineering, parallel numerical algorithms, and performance analysis.

Chinh Le obtained his bachelor degree with high honors in electrical engineering from Case Institute of Technology in

1969 and his Ph.D. in theoretical physics from the University of British Columbia in 1975. He is a senior scientist for high-performance computing at the Rosen Center for Advanced Computing. Prior to that he was manager for scientific applications at Vanderbilt University Computing Center. His research interests include distributed computing and numerical algorithms.

Marisol Koslowski is an assistant professor in the School of Mechanical Engineering at Purdue University. She received her M.S in 1999 and her Ph. D. in Aeronautics in 2003 from the California Institute of Technology. Her research interests include computational solid mechanics, mechanical properties of micro- and nano- structured materials.