LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-CONF-222086

ParaDiS on Blue Gene/L: stepping up to the challenge

Gregg Hommes, Athanasios Arsenlis, Vasily Bulatov, Wei Cai , Richard Cook, Masato Hiratani, Tomas Oppestrup, Moon Rhee, Meijie Tang

June 15, 2006

Supercomputing 2006
Tampa, FL, United States
November 11, 2006 through November 17, 2006

ParaDiS on BlueGene/L: stepping up to the challenge

G. Hommes, A. Arsenlis, V. V. Bulatov, W. Cai, R. Cook, M. Hiratani, T. Oppelstrup, M. Rhee and M. Tang

Lawrence Livermore National Laboratory, University of California

Abstract

This paper reports on the efforts to enable fully scalable simulations of Dislocation Line Dynamics (DLD) for direct calculations of strength of crystalline materials. DLD simulations are challenging and do not lend themselves naturally to parallel computing. Through a combinations of novel physical approaches, mathematical algorithms and computational science developments, a new DLD code ParaDiS is shown to take meaningful advantage of BG/L and, by doing so, to enable discovery class science by computation.

I. Introduction: promise and challenge of Dislocation Line Dynamics

Discovery of dislocations is one of the most fascinating developments in the history of materials physics. In 1934, three eminent scientists, E. Orowan, M. Polanyi and G. Taylor, simultaneously and independently published three papers [1-3] proposing exactly the same hypothesis to explain a ubiquitous yet perplexing observation of plastic deformation in crystals. While it was firmly established that crystals yield to external loads by shearing atomic bonds across crystal planes, the maximum loads that crystals were observed to sustain were orders of magnitude below levels that a simple simultaneous bond-shearing theory of crystal deformation would predict. The theory of dislocations explained this fundamental discrepancy by proposing that atomic shearing takes place one bond at a time, by propagating topological wrinkles – dislocation lines – along the crystal planes (this is what a smart housekeeper does to move a heavy carpet lying on the floor). Remarkably, while explaining much of the observed behaviors, dislocations remained a beautiful hypothesis for over 20 years until they were first observed with the then newly invented experimental method of Transmission Electron Microscopy [4]. Over time, the principal role of dislocations in crystal strength and ductility and numerous other properties of crystalline materials has been established firmly and incontrovertibly.

The beauty of dislocation theory is its economy of means: it distills the whole complexity of crystal strength and ductility into the motion and interaction of dislocations line defects that, even in the worst circumstances, occupy no more than about 10^{-5} of the material volume while the rest of the material remains essentially perfect and featureless. Thus, if one were to simulate, on a computer, dislocation motion in response to external loads, material strength could be directly *computed*. Such a direct simulation turns out to be a daunting task given that dislocation motion takes place on the length and time scales of nanometers and picoseconds whereas the strength response is defined by the collective motion of huge numbers of dislocation lines on the scales of tens of microns and seconds. Consequently, even though material scientists have *known* for over 70 years that crystal strength is defined by dislocation physics, it has been impossible thus far to put this knowledge to practical use and *to compute crystal strength*. As a result, the fact that dislocation physics defines material strength has had little or no impact on practical engineering applications that continue to rely on empirical knowledge and phenomenological descriptions of material, such as in designing a new bridge or a car. The direct simulation approach of

Dislocation Line Dynamics (DLD) is being advanced to bridge this disconnect between the fundamental physics of dislocation motion and engineering mechanics of material strength.

The basic premise of DLD is to represent the material subjected to deformation by a volume populated by dislocation lines and evolve this population using the known equations of dislocation theory [5,6]. DLD is an interface-tracking or, rather, a line-tracking approach. A single step of a DLD simulation consists of (1) computing forces on the properly discretized dislocation lines, (1) moving the lines in response to forces, (3) identifying instances in time and space when and where dislocation lines collide (intersect) or dissociate and (4) re-meshing the lines to better represent their evolving geometry. In principle, a DLD simulation can be used to subject the virtual crystal to arbitrary loading conditions (straining rate, temperature, pressure, etc) and to compute how much stress the material can sustain before collapsing under those conditions. Furthermore, not only the macroscopic average response of the material can be computed but also the underlying evolution of the dislocation network can be tracked in full detail. Thus, DLD is not only a versatile virtual material-testing machine but simultaneously an amazing computational *in situ* microscope.

Unfortunately, the ability of DLD approach to deliver on its promise is severely limited by the astonishing amount of calculations required to evolve large dislocation ensembles over long time intervals. Massively parallel computing appears to be the only viable approach to closing the wide gap between the levels of computational performance afforded by the existing DLD codes and that required to compute crystal strength. Development of an efficient code for DLD simulations on massively–parallel computing platforms has been the objective of the *ParaDiS* (Parallel Dislocation Simulator) project at LLNL since its inception in 2001. The *ParaDiS* team includes physicists and computer scientists working together to develop mathematical algorithms and the parallel code capable of handling the multiple challenging aspects of DLD simulations at very large scales.

Among the challenges of parallel DLD simulations, the first is the need to accurately track the constantly evolving topology of dislocation line network across the boundaries of computational sub-domains – this is a difficult task given the complexity of possible topological scenarios and the distributed ownership of topological data. Written in C, more than half the lines of *ParaDiS* code are dedicated to a consistent parallel treatment of the evolving line topology. The second computational challenge is to properly cope with the extreme spatial and temporal heterogeneity of dislocation network that evolves. Dislocations are known to become lumped into dense bundles separated by large volumes of dislocation-free material in which periods of very fast motion are interspersed with periods of relative calm. Such behaviors are similar to the complex collective behaviors encountered in computational astrophysics or plasma hydrodynamics. This commonality stems, in part, from the dominance of long-range interactions in all these problems, including DLD. This long-range interaction, in itself, presents yet another challenge for parallel implementation. Lastly, the same long-range interactions between dislocation lines become singular at close encounters, causing extreme stiffness in the equations of motion of dislocation lines.

The *ParaDiS* team and its collaborators have made significant progress addressing some of these and other difficulties. Making DLD into a powerful method for computational prediction of material strength necessitated new developments in dislocation theory, materials physics, and mathematical algorithms: most of these developments have been presented in the literature [7-11]. This paper reports on the computer science aspects of *ParaDiS* development with the particular emphasis on overcoming the challenges of DLD simulations on the massively-parallel

Blue-Gene/L platform. The next section briefly describes main mathematical algorithms implemented in *ParaDiS.* Section 3 presents and analyses the data of ultra-scale *ParaDiS* simulations on Blue-Gene/L partitions ranging from 1,024 to the full-size 131,072 processors. Section 4 contains a brief discussion of science implications of *ParaDiS* simulations on Blue-Gene/L followed by a summary in section 5.

II. *ParaDiS* algorithm

ParaDiS represents the dislocation line network as a set of nodes connected to each other by straight line segments. Because of the conservation of topological charge (the Burgers vector), the lines can not terminate anywhere but can branch at a node preserving the net charge balance at the node. Therefore, all nodes must have at least two neighbor nodes to which they are connected, and there is no upper limit on the node connectivity. Dislocation motion corresponds to motion of the nodes in space resulting in changes in the network geometry, i.e. length and orientation of the connecting segments. In addition, a variety of topological events can take place during which the connectivity of the network changes requiring the creation and/or deletion of nodes and segments. Among the critical events are (a) node dissociation in which a node with connectivity greater than four may split into two separate nodes while still preserving Burgers vector balance, and (b) collisions which occur when two unconnected dislocation lines intersect or come into contact with each other. To reduce complexity of topological operations and to enable efficient bookkeeping of network topology across the boundaries of computational domains, all such topological operations are logically reduced to combinations of events of only two types: *split_node* and *merge_node.* The *split_node* operator moves selected connections from an existing node to a newly created node, and introduces a new connection between the existing node and the new node if it is needed to preserve the balance of the topological charge of the network. An example of the *split_node* operation is shown in Figure 1. The *merge_node* operation is the inverse of *split_node*: it collapses two existing nodes into one by moving all of the connections from one node to the other, resolving any topological inconsistencies that result, and deleting one of the nodes. Any topological event in the dislocation line network can be represented as a combination of these two basic operators. Furthermore, the balance of topological charge is maintained for as long as it is preserved in each *split_node* and *merge_node* operations. Use of these two operators considerably simplifies treatment and bookkeeping of topological events in *ParaDiS*.

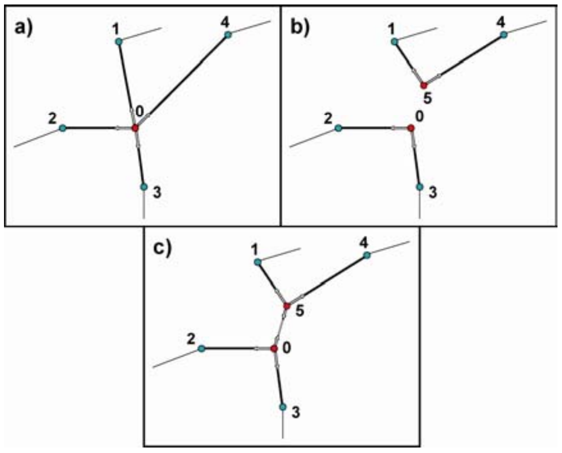

Figure 1: Topological steps taken during a *split_node* operation performed on node 0 that initially has four connections, as shown in (a). (b) New node 5 is created taking two of four connections from node 0. (c) An additional connection between nodes 0 and 5 is made, if necessary, to maintain the balance of topological charge.

The nodes move in response to forces exerted by external loads and by the interactions among all line segments in the dislocation network. The forces on a given node can be accurately computed using analytical equations supplied by the classical continuum theory of dislocations [12]. On the other hand, nodal response to these forces is ultimately defined by the details of atomic motion in the geometric center of the dislocation line around the node, the so-called *dislocation core.* We rely on dislocation theory, atomistic simulations and experiments to define equations for nodal motion in response to the nodal forces [8,9]. Computing nodal forces resulting from the interaction among the dislocation segments is the bulk of the computational expense in *ParaDiS*. The unit element of the force calculation is the evaluation of the force exerted by a single straight segment on a dislocation node. Depending on the node connectivity, one such calculation takes on the order of a few microseconds. Given that the interaction is long-ranged, decaying as ~1/r, and the number of segments N_s in the network can be very large (many millions), the expense of computing the force on every node can be overwhelming if a straightforward $O(N_s^2)$ algorithm is used. To reduce computational complexity to $O(N_s)$, we developed an efficient Fast Multipole Method (FMM) algorithm in which the effect of well separated groups of dislocations is lumped into a hierarchy of point-like sources (moments) of remote force. Our version of FMM uses a regular grid decomposition of the entire simulation volume and Taylor expansion coefficients to represent the distribution of the remote force in every grid cell of the multipole hierarchy. So far, all *ParaDiS* simulations have been performed in a cube closed into a 3D torus by periodic boundary conditions (PBC). To account for the long-range periodic image interactions, our version of FMM uses a correction procedure suggested in [13]. The calculation of nodal forces is broken into two components: (1) evaluation of forces from nearby segments using direct analytical expressions (typically more than a thousand of such segments for each node in the dislocation network) and (2) evaluation of forces exerted by all other, remote segments using the FMM algorithm.

The classical continuum theory of dislocations predicts that interaction forces among dislocations become very large (actually diverge as ~1/r) at close distances causing considerable stiffness of the nodal equations of motion. To remedy this non-physical singularity, various *ad hoc* cut-off procedures have been devised [12]. While working on *ParaDiS*, we developed a non-singular version of the continuum theory of dislocations that dispenses with the cut-offs by smearing out a dislocation line into a distribution of lines (a bundle) with a characteristic radius *a* [7]. The new non-singular theory is free of inconsistencies that plagued the application of the classical singular theory in close range dislocation interactions. Furthermore, by choosing an appropriate value for radius *a*, it is possible to significantly reduce the stiffness of the nodal equations of motion.

At the heart of our efforts to enable DLD simulations on massively parallel computers is the issue of load balancing. As stated, in the course of a straining experiment, whether real or virtual, dislocation lines multiply incessantly and gradually organize themselves into increasingly heterogeneous (lumpy) structures where large numbers of lines bundle together into tight braids or walls leaving a lot of space devoid of any dislocations in between. The scales of such heterogeneities of dislocation structures range from a few nanometers to multiple microns and possibly beyond. Given this vast range and the fact that the structure is constantly evolving during the straining simulation, it was not even clear at the outset if efficient domain decomposition and load balancing among many thousands of processors was at all possible. In *ParaDiS*, we use a recursive coordinate sectioning algorithm [14] to partition the cube-shaped simulation volume into non-overlapping spatial domains. Given a dislocation configuration, initial partitioning is performed using an estimated cost of computing forces on the nodes owned by a given processor. First, the cube is divided into slabs along X direction so that the estimated cost is equally partitioned among the slabs. Second, similar partitioning is done along Y direction within each slab by assigning equal estimated cost to each "column" within each slab. Finally, each column is partitioned along Z direction into "cells" such that each cell partition is expected to have approximately equal cost (Figure 2). Subsequent re-balancing is performed by monitoring how much time each CPU spends doing force evaluation (by far the most expensive element) on each compute cycle. Then, between the cycles, the boundaries of cells, columns and slabs are moved recursively to equalize the compute time among the neighboring domains. Our earlier experience running *ParaDiS* generally attested to the efficiency of its algorithms [15] but also exposed significant problems that had to be addressed in order to achieve sustainable load balancing among 10^5 domains. These improvements will be discussed in more details in the next section.

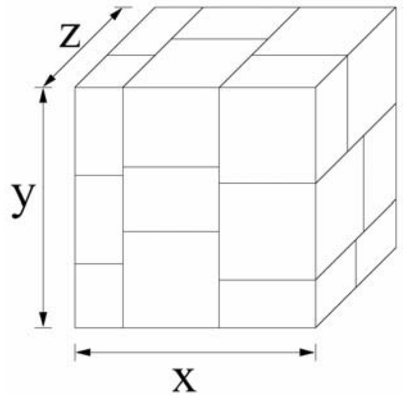

Figure 2: Decomposition of the simulation space into 3 × 3 × 2 domains along X, Y, Z axes.

III. *ParaDiS* on Blue-Gene/L: growing (together) pains

BlueGene/L (BG/L) is a massively-parallel computing system developed by IBM in partnership with the US DOE/NNSA Advanced Simulation and Computing (ASC) program [16]. The system consists of 65,536 compute nodes each node containing two processors with 512 Mb of memory per node. The system's software provides two modes of operation for application programs. In communication coprocessor mode there is one MPI task per node with one processor executing the application while much of the message passing communication is handled by the second processor. Virtual node mode allows two MPI tasks per node, one task per processor. The machine recently achieved a sustained speed of 280.6 teraOPS on the Linpack benchmark at Lawrence Livermore National Laboratory, close to its theoretical peak performance of 360 teraOPS.

ParaDiS has had a history on BG/L. That the new code might be able at some point to take advantage of the BG/L architecture was realized early, since DLD simulations have a relatively small memory footprint while commanding extremely large numbers of OPS per degree of freedom during each time step. In the summer of 2003, before the machine procurement to LLNL was approved, the code's early version, then known as *dd3d*, was successfully ported to the early 32-node hardware prototype of the BG/L system. The uniqueness of *ParaDiS* development on BG/L was that both the machine and the code grew together, from their infancy to adulthood. An "adolescent" stage of this maturation process was reported in our paper presented at SC04 [15]. By the time the draft paper was due in April 2004, the code had ran on all 4,096 (4K) nodes of BG/L then available. Over that summer, the machine grew to 16,384 (16K) nodes, and the results of *ParaDiS* scaling runs on the enlarged machine were reported at the SC04 Conference in November 2004. As the code grew and matured it was successfully executed on BG/L as the hardware grew over time from its original 32-node prototype to the full system configuration of 65,536 (64K) nodes with 131,072 (128K) processors. This process was a slow (and sometimes painful) one as each step up to a larger number of processors revealed new and unexpected behaviors and limitations in the algorithms. Over time, the code has been enhanced and refined to its current state where it has successfully scaled to the full 131,072 processor BG/L system.

This section describes the scaling performance of *ParaDiS* on BG/L. Typically, both weak and strong scaling performance data are given to gauge code performance on parallel computer architectures. Weak scaling tests assume that a change in the problem size should result in a proportional change of the computational cost. While this can be achieved by periodic replication of a small *ParaDiS* simulation, such configurations are not representative of the real system that grows increasingly heterogeneous as the simulation progresses. In this paper we present data only on more meaningful strong scaling tests of *ParaDiS* on BG/L. The problem selected for the scaling tests was randomly selected from an actual production BG/L simulation that was performed to examine the effects of extremely high straining rates (10^4 s^{-1}) on crystal strength and dislocation microstructure (see section IV). No additional optimization was performed to enhance the code performance in the scaling simulations: the scaling runs were performed using exactly the same runtime parameters as in the long production simulation. The test problem consisted of a 10 x 10 x 10 μ^3 cube containing about 10 million dislocation nodes. Scaling tests were executed at processor counts ranging from 1K to full systems runs at 128K. The 64K and 128K tests were executed in the virtual node mode and all other tests were executed in the co-processor mode.

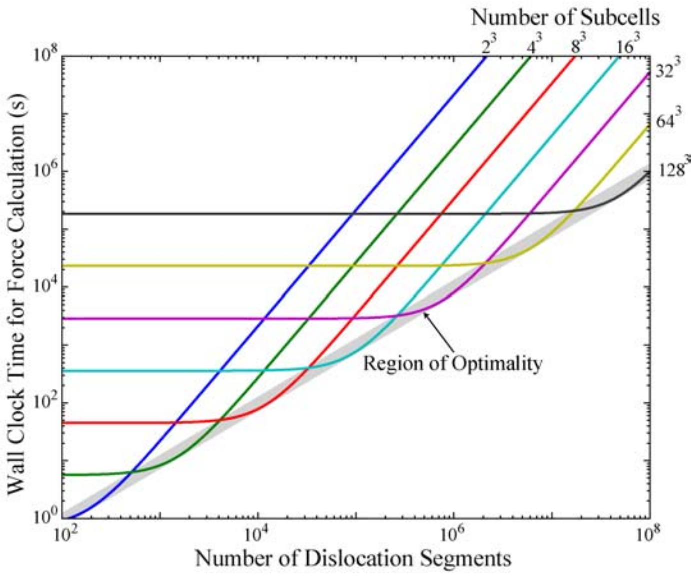

Figure 3: Estimated time required to compute forces on a single CPU of BG/L as a function of the number of dislocation segments N_s. The time scales as $O(N_s)$ provided the number of FMM subcells is chosen optimally.

Given that most of every compute cycle is taken by the calculations of forces on dislocation segments, it is critically important to minimize this cost. Assuming that the segments are distributed uniformly in the volume (a very strong assumption!), the single CPU time required to compute the forces can be approximated as a function of the number of segments, *N*, the number of FMM subcells, *K*, the order of the multipole expansion of the lumped sources, *M*, and the order of the Taylor series expansion used for interpolation of the far field stress, *S*. It turns out that, compared to all other routines, the local segment-segment force calculations (*segsegforce*) and evaluation of the Taylor expansion coefficients for the local stress due remote segments (*mktaylor*) are by far the costliest. The estimated total time spent calculating forces during a single integration time step on a single processor is approximately[1]

$$C(N,K) = \frac{27N^2}{K} \cdot C_{segsegforce} + 216K \cdot C_{mktaylor}, \qquad (1)$$

where $C_{segsegforce}$ = 13.2μs and $C_{mktaylor}$ = 553μs on BG/L for M=2 and S=4. To achieve the desired O(N) scaling, the number of multipole subcells K is selected from the powers of 8 to minimize the total cost for a given number of segments N, as shown in Figure 3.

The minimal BG/L partition size of our actual timing runs was 1,024 processors because it was impossible to fit all of the data for 9,558,824 dislocation nodes in the memory on the smaller partitions. Still, it is possible to use formula (1) to estimate the speedup (in the force calculations only) from the single processor to the full machine. Assuming that communication between CPUs is instantaneous and that 9558824 degrees of freedom are evenly distributed over the whole machine, the most optimistic (perfect scaling) estimate for the time spent calculating forces during a time-step is approximately 1.2 s. A more realistic estimate leads to a higher estimate, at 2.7 s. This latter value accounts for our deliberate decision, in the present version of *ParaDiS*, to compute force interactions multiple times if the degrees of freedom are owned by different processors rather than increase the inter-processor communication between during the force calculation. Comparing this estimate to the achieved timing of 4.8s (Table 1) yields a 56% utilization of the single processor performance on the full BG/L. This figure is consistent with another measure of load balance presented in Figure 6 to be discussed below.

Table 1: Time spent on force calculations in a single cycle of *ParaDiS* simulation on BG/L partitions from 1K to 128K processors

CPU count	Dislocation nodes	Total time (sec)	Time spent in force calculation (sec)
1024	9558824	278.71	267.75
16384	9558824	27.71	24.76
32768	9558824	15.44	12.92
65536	9558824	10.40	7.38
131072	9558824	8.91	4.85

Table 1 summarizes the total single-cycle times and the times spent on force calculations measured in the series of scaling tests. The code exhibits good overall scaling with increasing processor count showing considerable speedups up to the full machine size with 131,072 processors. This performance is outstanding given the inherently challenging character of Dislocation Line Dynamics simulations. As mentioned, the greatest of all difficulties is the naturally developing heterogeneity of the dislocation density. For example, among 64 x 64 x 64 FMM subcells used in the full machine simulation, some of the subcells located in areas where dislocations have "lumped" together in high density formations contain many hundreds of dislocation nodes while numerous other subcells are completely empty. Given this extreme spatial non-uniformity, the ability to dynamically balance the computational load has been absolutely critical to scaling to large numbers of processors.

[1] On each integration time-step the local force routine *segsegforce* is called twice.

As was described in the previous section, the mechanism used in *ParaDiS* to partition the computational load and to maintain load balance dynamically is by a hierarchical recursive sectioning algorithm combined with domain relaxation. This algorithm is able to track the constantly changing distribution of computational load with admirable consistency. Shown in Figure 4 are two planar cross-sections through the computational volume. The cross-section on the left shows where the dislocation lines intersect the cutting plane, whereas the cross-section on the right shows the distribution and shapes of compute domains that develop through load relaxation to deal with that heterogeneity at the same instant in time. Clearly, the load balancing algorithm goes a great distance trying to maintain an equal load distribution and generally does a good job tracking locations with very high dislocation densities. Remarkably, while the volumes of the largest and the smallest computational domains in this simulation differ by astounding three (!) orders of magnitude, the sustained load balance remains around 80% for most of the tested partition sizes dropping to 64% for the full 128K partition of BG/L (Figure 6).

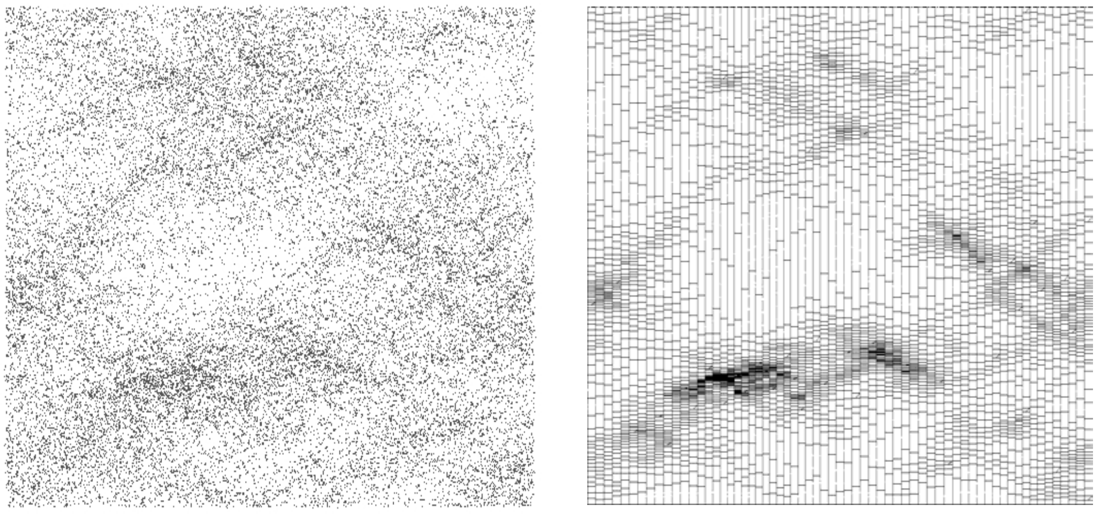

Figure 4: (a) Dislocation distribution in a single planar cross-section through the computation volume. (b) The instantaneous distribution of computational domains attained in the same region of the computation volume as in (a).

The overall rate of speedup does drop off as the processor count increases. This less than perfect speedup is more pronounced for the overall cycle time shown in Figure 5 while the speedup for the time spent on force calculations is significantly better. There are several factors accounting for these behaviors. In *ParaDiS*, work load is balanced based on the amount of time each processor spends calculating forces on the dislocation segments. Although the force calculations consume the dominant portion of computing time, there are several other relatively expensive tasks whose timing is also less predictable from cycle to cycle. As the processor count increases and *ParaDiS* works to minimize the time spent in the force calculations, the amount of time spent in these other tasks also decreases, but not as significantly. Hence, these other tasks begin to use a larger portion of the overall time with adverse effects on scaling. Another factor is that in *ParaDiS*, the majority of processor-to-processor communications are not just nearest-neighbor, but also “nearby neighbor”. As more and more processors are used, the number of nearby neighbors can increase significantly resulting in a larger fraction of the compute cycle time spent communicating. This communication overload increases with the increasing number of processors and becomes a factor at extremely large processor counts. Our use of the recursive partitioning algorithm contributes to this problem because sub-volumes assigned to different processors can not adjust their shapes independently but are constrained to relax within their respective columns and slabs. Working to re-balance the evolving load, this algorithm leads to the development of sub-volumes with large aspect ratios (Figure 4) that also tend to have large numbers of nearby neighbors. Another potential drawback of recursive partitioning manifests itself is slow overall relaxation of load imbalance – depending on the number of domains and the initial problem imbalance, this relaxation can take hundreds of cycles before achieving an optimal balance. To enhance the rate of relaxation of partition boundaries, we developed a procedure in which, rather than timing the whole cycle of force calculations for each processor, *ParaDiS* estimates the per-processor load by counting the number of times the *segsegforce* calls each processor would make and shifting the domain boundaries to balance the counts. Such "empty" cycles are computationally expedient and can be repeated many times for a given instantaneous load distribution allowing to greatly enhance the relaxation rates and to rebalance the load more efficiently.

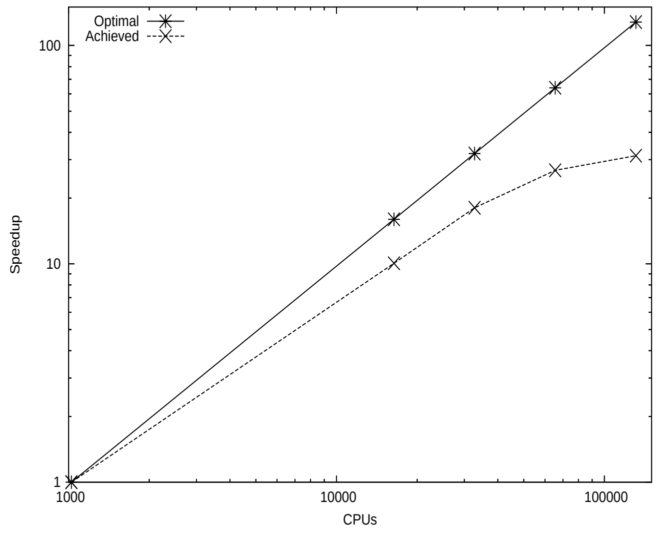

Figure 5: Relative overall speedup for strong scaling *ParaDiS* simulations on BG/L.

The overall effectiveness of this dynamic load balancing algorithm is shown in Figure 6 for all tested processor counts. In addition to a gradual reduction trend, there is a more significant drop in the load balance from 64K to 128K processors which was in large part due to the fact that the test problem was a bit too small for the full 128K partition[2]. Still, the data does indicate that, for very large and heterogeneous simulations, the current load balancing algorithm may need improvement. This is an area that remains to be explored to further improve the scalability of *ParaDiS*.

[2]To generate a large dislocation network that is representative of real crystals, we had to let our line network "grow" on BG/L. Yet, we were unable to "grow" our simulation to a size that could take full advantage of the entire BG/L before the machine was taken down to prepare it for classified service in support of the NNSA stockpile science mission.

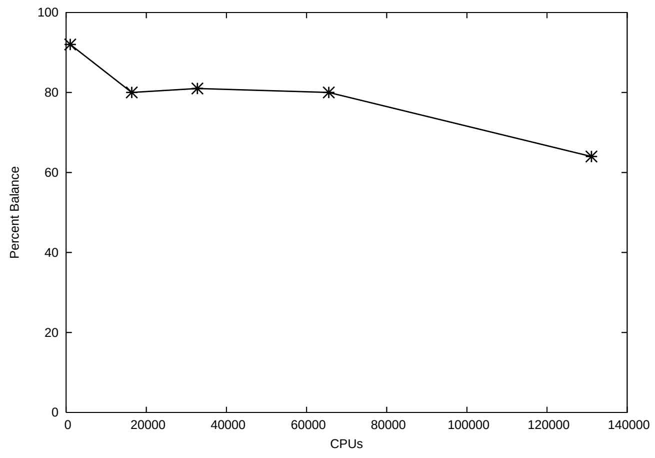

Figure 6: Percent load balance as a function of processor count. As a measure of load balance here we use the ratio of the average time spent by each processor performing its own force calculation, to the average time each processor spends computing the forces and waiting all other processors to finish their force calculations.

IV. Glimpses of new physics from *ParaDiS* simulations

Recently the new capability for DLD simulations embodied in *ParaDiS* was used to discover new mechanism of strain hardening in metals. The initial observation of anomalously strong dislocation tangles – multi-junctions - came directly from *ParaDiS* simulations. These observations were subsequently verified by direct atomistic simulations and Transmission Electron Microscopy experiments. Finally, very large scale *ParaDiS* simulations produced direct evidence for the key role of the multi-junctions in defining the anomalously large orientation dependence of strength in BCC metals – a behavior long observed and yet puzzling [17].

The simulation reported here is more than a scaling test. It predicts the strength of a crystal under an extremely high straining rate. Such simulations are needed for accurate predictions of the behavior of stockpile materials during nuclear device detonation and are an integral part of the NNSA ASC Program. Under the current moratorium on nuclear tests, simulations have grown to become one of the primary tools for certification of the aging nuclear stockpile. In this particular simulation we examined the behavior of a single crystal of molybdenum under compressive straining at the rate 10^4 s^{-1}. Initially, a few dislocations lines were inserted in the simulation volume. Soon after the straining load was applied, dislocations started to multiply at a very high rate eventually increasing the number of dislocation lines by nearly three orders of magnitude. The multiplication was so profuse that we had to successively increase the number of processors from the initial 64 CPUs (on the *Thunder* machine at LLNL) to 128, 256, 1K and so on, all the way to 64K and 128K of BG/L. As a result, the number of dislocation nodes increased from a few hundred to over 12 million, still shy of the 15-20 million dislocation nodes which was our eventual target for this simulation.

We are just beginning to analyze this huge simulation[3] and will present its results in the literature in a near future. Preliminary indications are that dislocations appear to organize themselves into a network with fractal geometry – the significance and origin of this striking behavior remain unclear. Figure 7 shows dislocations in a thin (1/16th of the total depth) cross-section of the simulation cube at an intermediate stage of the simulation (at about 8 million dislocation nodes).

[3] We are currently developing means to handle, analyze and visualize the immense amounts of data obtained in this simulation: the data for a single simulation snapshot takes about 3.5 Gb of disk space.

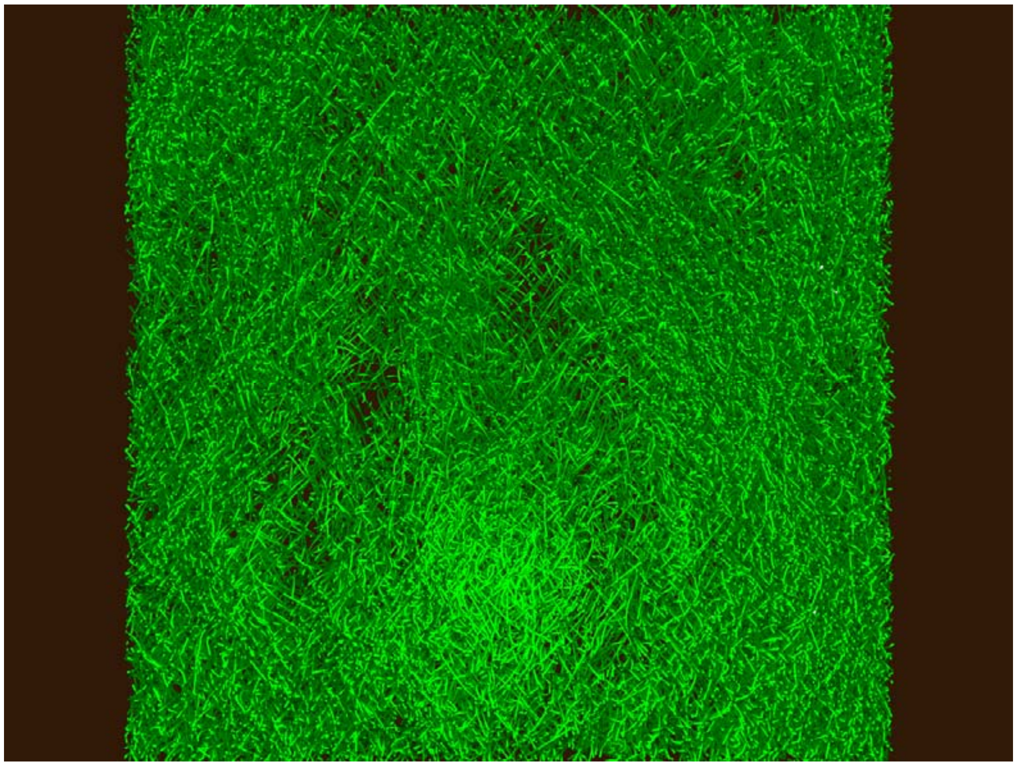

Figure 7: A thin cross-sectional view through the simulation box – dislocations are shown as thin green lines.

V. Summary

Development of predictive Dislocation Line Dynamics simulations bridges the gap between the physical mechanisms of material deformation at the atomistic scale and the practical mechanics of materials for engineering applications. This is one of the cases where new ability to compute large and long brings about new physical insights and generates valuable engineering data that is impossible or impractical to obtain experimentally. In addition to demonstrating the feasibility of DLD as a practical approach for predicting material's strength by direct computation, a broader significance of this study is in demonstrating that even in a challenging physics application that does not lend itself naturally to parallel computing, it is still possible to take meaningful advantage of a massively parallel machine of the BG/L scale. We hope that this demonstration will be encouraging for researchers in computational astrophysics, computational plasma hydrodynamics and other fields where severe computational limits stand in the way of further progress in understanding.

This work was supported over the years by the US DOE/NNSA ASC Program and LLNL LDRD office. We thank E. Chandler, K. Budil, T. Diaz de la Rubia, D. Nielsen, M. McCoy and D. Kusnezov for their encouragement and support of the *ParaDiS* code development effort. We are deeply indebted to Tim Pierce for his early courageous effort in *ParaDiS* code development and for subsequently sharing with us valuable ideas and insights. This work was performed under the Auspices of the U.S. Department of Energy by the University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

1. E. Orowan. Zur kristallplastizitat. *Zeitschrift Phys.* **89**, 605-659 (1934).
2. G. Taylor. The mechanism of plastic deformation in crystal. Part I. Theoretical. *Proc. Roy. Soc. A* **145**, 363-404 (1934).
3. M. Polanyi. Uber eine Art Gutterstorung, die einen kristall plastich machen konnte. *Zeitschrift Phys.* **89**, 660-664 (1934).
4. P. B. Hirsch, R. W. Horne, and M. J. Whelan. Direct observations of the arrangement and motion of dislocations in aluminium. *Phil. Mag.* **1**, 677-684 (1956).
5. K. W. Schwarz, Simulation of dislocations on the mesoscopic scale. I. Methods and examples. *J. Appl. Phys.* 85, 108(1999).
6. B. Devincre and L. P. Kubin. Mesoscopic simulations of dislocations and plasticity. *Mater. Sci. Eng. A* 8, 234-236 (1997).
7. W. Cai, T. Arsenlis, C.R. Wenberger, and V.V. Bulatov. A non-singular continuum theory of dislocations. *J. Mech. Phys. Solids* **54**, 561-587 (2005).
8. W. Cai, V. V. Bulatov, Ju Li, J.P. Chang and Sid Yip. Dislocation core structure and mobility", (2004) in: *Dislocations in Solids*, volume 12, edited by F.R.N. Nabarro, pp. 1-80.
9. W. Cai and V.V. Bulatov. Mobility laws in dislocation dynamics simulations. *Mater. Sci. Eng*. A **387-389**, 277-281 (2004).
10. W. Cai, V. V. Bulatov, T. G. Pierce, M. Hiratani, M. Rhee and M. Tang. Massively-parallel Dislocation Dynamics simulations", in *Mesoscopic Dynamics of Fracture Processes and Materials Strength*, IUTAM Symp. Proceedings (Osaka, 2003), 1-11.
11. W. Cai, V. V. Bulatov, J. Li, J.P. Chang and Sid Yip. Periodic image effects in dislocation modeling. *Philosophical Magazine* **83**, 539 (2003).
12. J. P. Hirth and J. Lothe, *Theory of Dislocations,* 2nd edn (Wiley, New York, 1982) 857 p.
13. M. Challacombe, C. White and M. Head-Gordon. Periodic boundary conditions and the fast multipole method. *J. Chem. Phys.* **107**, 10131-10140 (1997).
14. M. Berger and S. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Trans. Computers* **C-36**, 570-580 (1987).
15. Bulatov, V.V. *et al*. Scalable line dynamics in *ParaDiS*. *Supercomputing* (2004).

    http://www.sc-conference.org/sc2004/schedule/pdfs/pap206.pdf
16. http://www.llnl.gov/computing/hpc/resources/OCF_resources.html#bluegenel
17. V. V. Bulatov *et al.* Dislocation multi-junctions and strain hardening. *Nature* **440**, 1174-1178 (2006).