Early LLNL Application Scaling Results on BlueGene/L

(Presented at SC04, Pittsburgh PA, November 8-12, 2004)

National Nuclear Security Administration
Advanced Simulation and Computing Program

Andrew W. Cook, Jeffrey A. Greenough, Francois Gygi, Frederick H. Streitz,
Alison Kubota, Vasily V. Bulatov, Steven Louis

Lawrence Livermore National Laboratory

UCRL-TR-207656

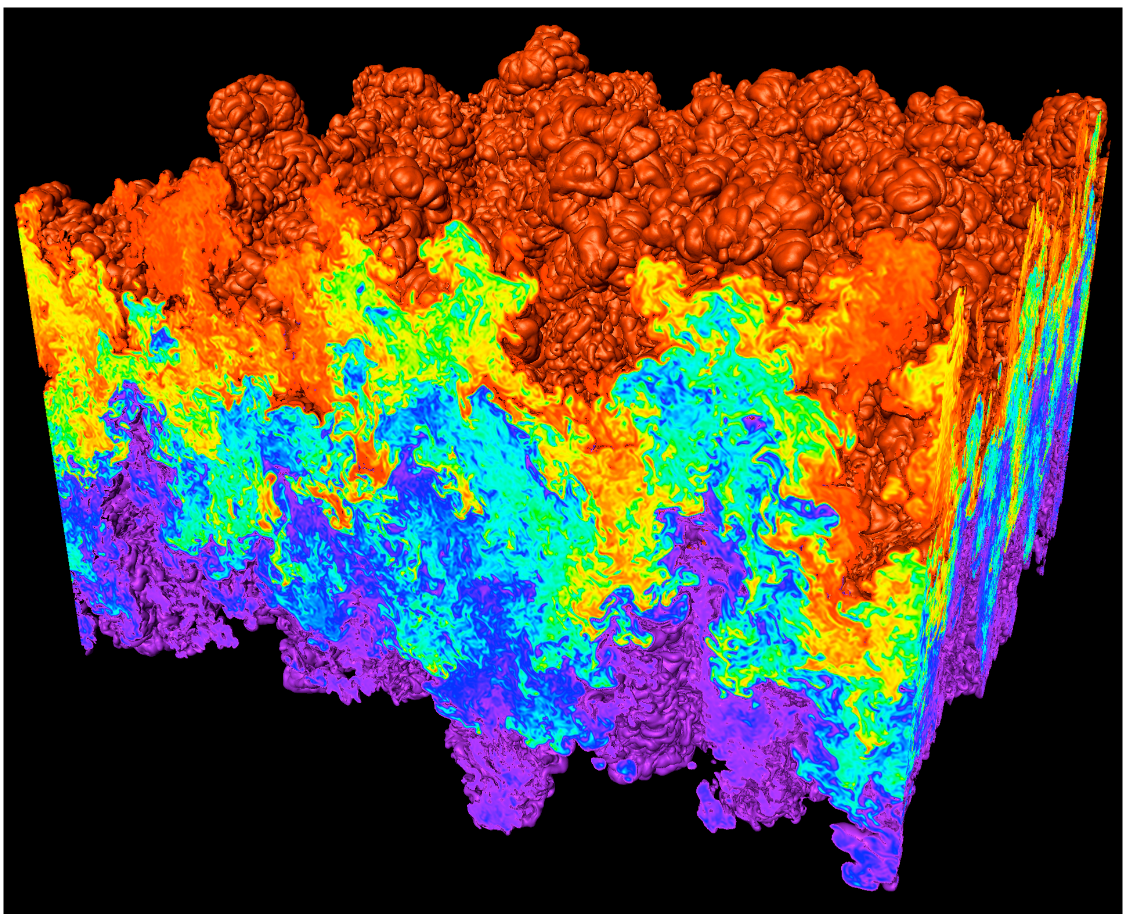
Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

MIRANDA

Code Contacts: Andy Cook, Bill Cabot and Peter Williams, LLNL

What is it?

Miranda is a high order hydrodynamics code for computing fluid instabilities and turbulent mixing. It employs FFTs and band-diagonal matrix solvers for computing spectrally-accurate derivatives, combined with high-order integration methods for time advancement; e.g., fourth-order Runge-Kutta. Fluid properties, i.e., viscosity, diffusivity and thermal conductivity, are computed from kinetic theory. The code contains solvers for both compressible and incompressible flows. It has been used primarily for studying Rayleigh-Taylor (R-T) and Richtmyer-Meshkov (R-M) instabilities, which occur in supernovae and Inertial Confinement Fusion (ICF).



Turbulent flow mixing of two fluids using large-eddy simulation of R-T instability.

What is the benefit from scaling it to large number of processors?

The grid resolution needed to support a sufficiently broad range of dynamical scales for turbulent flow is severe. Unless the resolution requirements are met, the flow cannot reach a state independent of both initial conditions and boundary conditions. Hence, virtually all past measurements of R-T and R-M growth rates have been sensitive to grid resolution (i.e., they were not converged). Very large simulations, using well in excess of 1000 grid points in each of three directions, are needed to support the large range of length scales necessary to grow R-T and R-M instabilities to full turbulence. Smaller simulations simply cannot capture true rates of growth and mixing due to initial/boundary effects.

How has the code been optimized on BG/L?

The code has been optimized for the BG/L torus by distributing the data among a Cartesian arrangement of processors. All portions of the code operating in a master-slave CPU manner have been replaced with fully parallel routines. Memory overhead has been reduced; the FFTW library has been ported and new I/O routines are currently being tested.

How has the code been tested for scaling and performance?

The most recent scaling results for Miranda are shown below:

8 x 8 x 512 grid/CPU in virtual mode:

# CPUs	Time/Step(s)	% comm. time
8192	9.84	86.3
16384	10.4	86.5
32768	8.15	82.2

8 x 8 x 1024 grid/NODE in coprocessor mode:

# NODEs	Time/Step(s)	% comm. time
512	3.16	34.3
1024	3.79	36.3
2048	3.96	39.8
4096	5.50	56.9
8192	7.02	65.3
16384	12.5	79.1

Does a demo exist? If so, what is it and what does it need?

YES – There will be a live demonstration of Miranda on a 128 node BG/L partition.

Description of demo:

Title: "Large Eddy Simulation of Rayleigh-Taylor Instability"

Importance: Rayleigh-Taylor instability is an important design consideration in Inertial Confinement Fusion. It also influences creation of heavy elements inside supernovae. A large computational domain, and hence a large number of grid points, is required to reach a state independent of both initial conditions and boundary conditions. The visualization shows turbulent flow mixing of two fluids using large-eddy simulation of Rayleigh-Taylor instability.

The demonstration uses the ViSUS high-performance streaming infrastructure to help solve the long-standing problem of fast and flexible data availability for large-scale simulations. A simulation linking ViSUS has good scalability and can stream and dump its data directly in a format that enables high performance I/O for a range queries including (i) arbitrary domain decomposition useful for restarting simulations with a varying number of processors, and (ii) progressive multi-resolution streaming useful for remote data retrieval, processing and visualization.

With ViSUS a scientist can monitor in real time the progress of scientific simulation directly from his office without transferring the data to the local disk of his workstation. A grid of 8-billion-nodes can be explored interactively on a laptop computer using a wireless connection to a remote repository. Overall the ViSUS streaming infrastructure and data model allows to minimize the problems and delays that continuous data movements introduce in a modern working environment where a scientist needs to access routinely heterogeneous computing resources distributed over local and wide area networks.

Who will run the demo at SC04?

Jeff Hagelberg, LLNL

RAPTOR

Code Contacts: Jeff Greenough, LLNL and Charles Rendleman, LBNL

What is it?

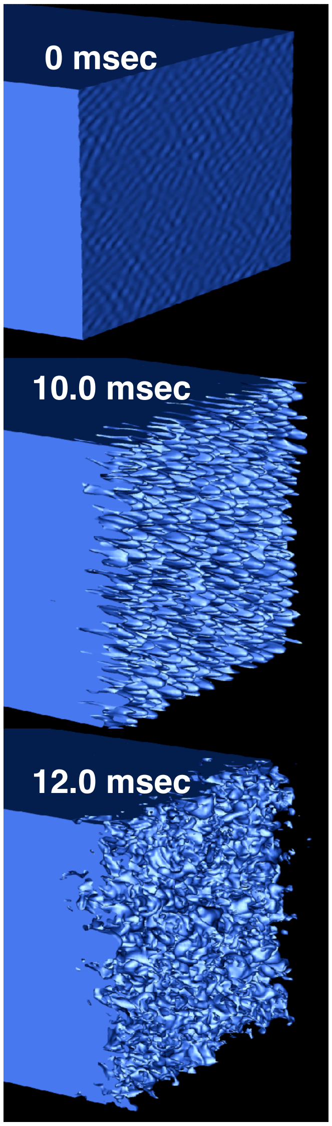
Raptor is a multi-physics Eulerian Adaptive Mesh Refinement (AMR) code used for applications at LLNL including astrophysics, Inertial Confinement Fusion (ICF) and shock-driven instabilities and turbulence. Raptor can be used to simulate purely fluid dynamics systems by solving the Eulerian equations (inviscid, non-conducting) or the Navier-Stokes equations (viscous, conducting) using a higher-order Godunov finite difference method. Raptor can also be used to simulate more complex physical systems where the fluids are coupled to the radiation field, such as in ICF or astrophysics. A fully implicit treatment is used to solve the radiation-diffusion equation coupled to the matter internal energy.

Raptor is based on the BoxLib and AmrLib general software infrastructure developed and maintained by the Center for Computational Sciences and Engineering at Lawrence Berkeley National Laboratory. BoxLib provides C++ foundation classes for templated data containers and their efficient manipulation. AmrLib adds framework support in C++ that extends BoxLib to efficiently support the demands of block-structured AMR. The entire software system, both base and framework libraries, as well as applications software and physics algorithms, has been optimized for efficient use of modern large-scale parallel computing platforms.

What is the benefit from scaling it to large number of processors?

In computational fluid dynamics (CFD), more resolution means that more of the physical length scales of interest are accurately represented by the numerical representation. Resolution is achieved by using more grid points per unit length in physical dimensions. Simulations also provide access to all of the data in the computational domain, e.g. point-wise density, momentum and energy, whereas experimental facilities are limited to either integral measures or single value point-wise measurements.

Simulations at full scale on BG/L will offer the computational power to gain an order of magnitude more resolution in simulations of three-dimensional shock-driven systems. Two of the systems to be investigated numerically on BG/L are modeled after the research shock tubes at the University of Arizona (low Mach number facility) and the University of Wisconsin-Madison (high Mach number facility) as well and laser driven systems like the National Ignition Facility (ultra-high energy density facility).



Simulation of Richtmyer-Meshkov Instability

When a perturbed density interface, separating two different materials, is traversed by a shock wave, vorticity is baroclinically generated at the interface by mis-alignment of the pressure and density gradients. The vorticity field then evolves due to the induced velocities distorting the density interface. This development is called the Richtmyer-Meshkov instability*. In the accompanying picture, we see an iso-surface rendering of the late time structure (t = 10 milliseconds) of a perturbed interface initially separating air (on the left and shown in blue) and SF6 (on the right and removed to aid visualization). The initial perturbation (t = 0.0 msec) is a spectrum of modes peaked at a wavelength of 1/16 the the transverse length. The incident shock Mach number is 1.3. The initial perturbation spectrum is imprinted on the interface and persists to long time with no appreciable changes.

After the nonlinear interface is re-accelerated by a counter-propagating shock wave, the interface transitions to turbulence (t = 12 msec). There are now a wide range of lengthscales present in the flow.

*Although it is termed an instability, it is not an instability in the classical sense.

Figures excerpted from UCRL-PRES-205229.

How has the code been optimized on BG/L?

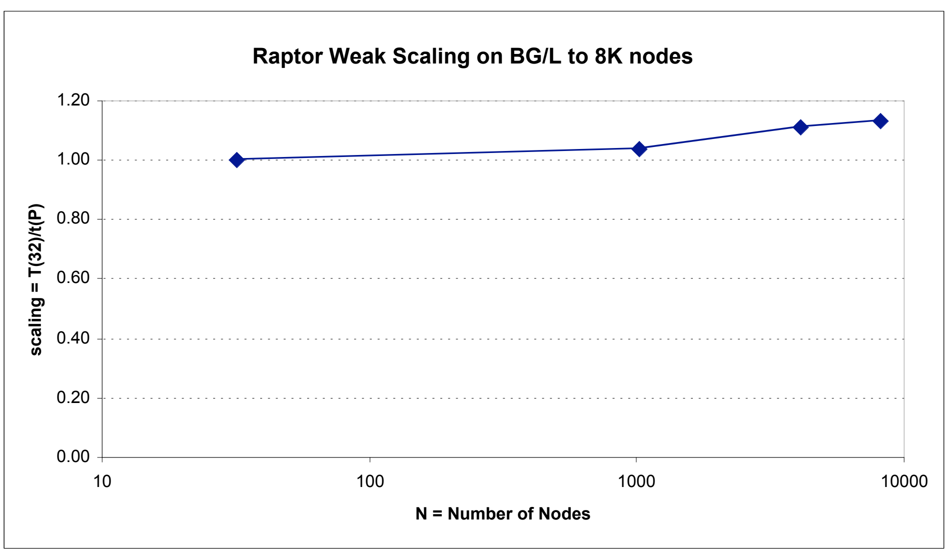
To date, the primary optimizations made to the Raptor code in porting it to BG/L have been modifications required by the xlC (IBM C++) compiler. Our testing on BG/L has shown that some optimizations are required to run at full machine scale and these optimizations are now in progress.

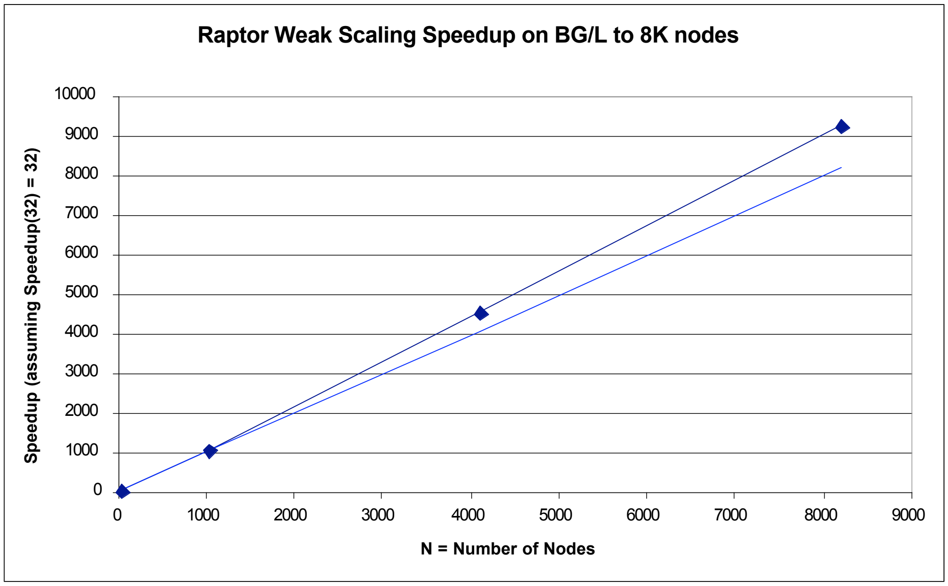
When using AMR methods on systems that evolve in time like our CFD applications, the grid hierarchy must be regenerated as required to maintain high resolution at the structures of interest. The current algorithm duplicates the mapping of data blocks to processors (distribution mapping) on all compute nodes and uses an approximately N^2 algorithm to create a new grid hierarchy (regridding). The new algorithm (development/implementation in progress) is based on a set of simplifying assumptions on the size of the data blocks and how they are laid out on any level of refinement. The new algorithm's benefits are to eliminate the memory overhead of representing the distribution mapping on the compute nodes and to provide an order N re-gridding algorithm.

How has the code been tested for scaling and performance?

1. Tests have been performed while running Raptor with just a single level of data across the computational domain, i.e. no adaptivity was used. On the Yorktown system, using a data block size of 32^3 with one block per compute node, we observe a time per step (seconds per step) of 1.95 sec on up 1024 processors. This is similar timing to that obtained on MCR (LLNL) using two 2.2GHz Pentium 4 CPU's per node and an Elan 3 interconnect.

2. Testing on an 8k compute node partition (Coprocessor mode) at IBM Rochester, with help from James Sexton at IBM, has also yielded successful tests with up to 8 32^3 data blocks per node. Weak scaling results are shown in the following two charts. For each point in the plot, there was a single 32^3 data block per node. A speed-up of over 10% compared to the initial 32-node result is shown at full 8k partition size. Plotting the data another way, as in the second chart below, shows better than linear scaling up to 8k compute nodes. The blue line shows what would be perfect scaling and the data points show that the actual speedup exceeds this.





3. To measure Raptor's strong scaling on up to the full BG/L machine at 64k compute nodes, runs were made on 8K nodes, by successively increasing the problem size per node from 1×32^3, 2×32^3, 4×32^3 and 8×32^3 data blocks. The next section gives a summary of the raw data used in here. One analysis that is possible from this is to make strong scaling projections to BG/L systems up to 64K nodes. This is done by noticing that starting from the 8×32^3 run on 8K nodes, if the number of nodes were to be increased to 16K, 32K, and 64 K, the problem size per node would be 4×32^3, 2×32^3, 1×32^3, respectively.

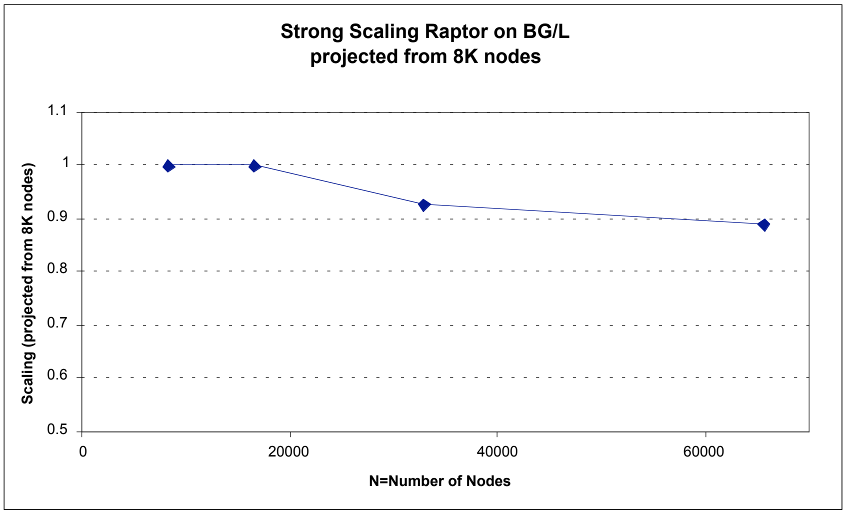
   Hence in the ideal case of perfect scaling
   t(N=B*8192,8/B) = t(N=8192,8)/B where B is the number of 32^3 blocks. The scaling performance can then be obtained as the ratio of this ideal time to the actual run time with B blocks. Thus,

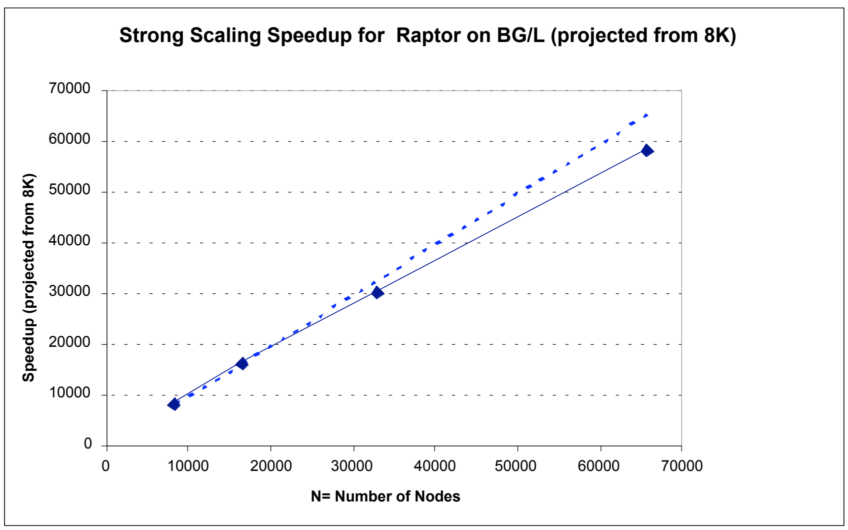
$$\text{Scaling}(N=B*8192,8/B) = t(N=8192,8)/[B*t(N=8192,B)]$$

   and the speedup is given by

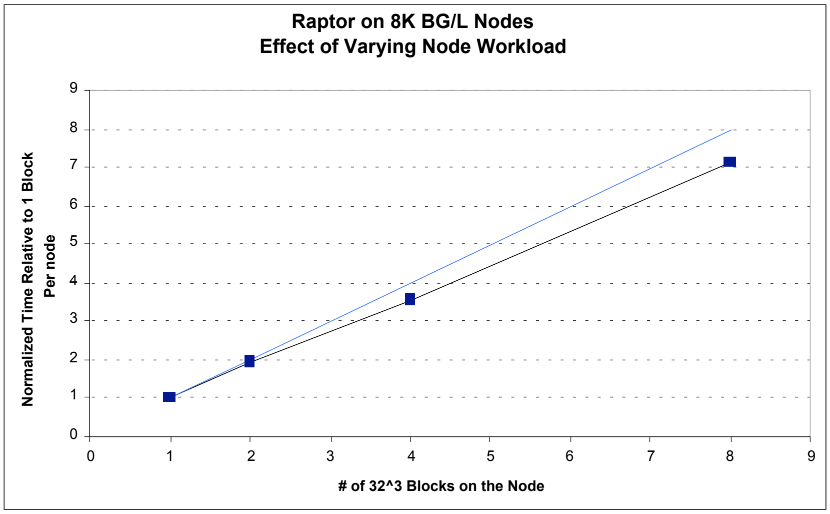
$$\text{Speedup}(N=B*8192,8/B) = N*\text{Scaling}(N=B*8192,8/B).$$

   These are plotted in the charts below and give an indication of how the runs should scale on larger systems, provided that the MPI performance is dominated by surface to volume ratio and not by other quantities such as MPI collective functions, which are neglected here.





4. The raw performance data for 1 x 32^3, 2 x 32^3 , 4 x 32^3 and 8 x 32^3 data blocks per nodes (successive doubling of node workload) on the 8k partition in Coprocessor mode is show below. Notice that the time to compute for B blocks increases in a sub-linear fashion by comparing the data to the solid line which shows perfect linear scaling. Data below this line exhibits better scaling. This means that doubling the workload per node requires less than a factor of two more work. More specifically the time increases by only 1.7, instead of the full factor of two for each doubling. This may come from using BG/L in Coprocessor mode where the second CPU is effectively used to help manage MPI communications overhead.



Does a demo exist? If so, what is it and what does it need?

YES – There will be a live demonstration of Raptor on a 128 node BG/L partition.

Description of demo:

Title: Numerical Simulation of Richtmyer-Meshkov (RMI) Instability"

Importance: RMI is an important instability in any physical flow containing shock waves and density interfaces, such as astrophysics, Inertial Confinement Fusion and high-speed combustion. The simulation shows a perturbed material interface (light gas on the left and heavy gas on the right) being accelerated by a shock wave and then evolving in time. The shock wave (not shown by the visualization software) moves from left to right across the interface. The underlying grid consists of 128 (x) by 64 (y) by 64 (z) cells decomposed into 128 data blocks that are distributed across the 128-node BG/L hardware.

Vorticity (curl of the velocity vector) is deposited at the interface due the misalignment of the pressure gradient (shock wave) and the density gradient (perturbed interface) as the shock wave traverses the interface. The vorticity field induces material motion that causes growth of the perturbations and mixing of the light and heavy gases. The visualization shows the small-scale initial perturbations growing to larger amplitude. In addition, three-dimensional "mushroom" structures coalesce into larger scale structures. As time goes on, the coalescing process slows. This is due to the fact that the turbulent kinetic energy contained in the developing interface is decaying with time.

Who will run the demo at SC04?

Jeff Hagelberg, LLNL

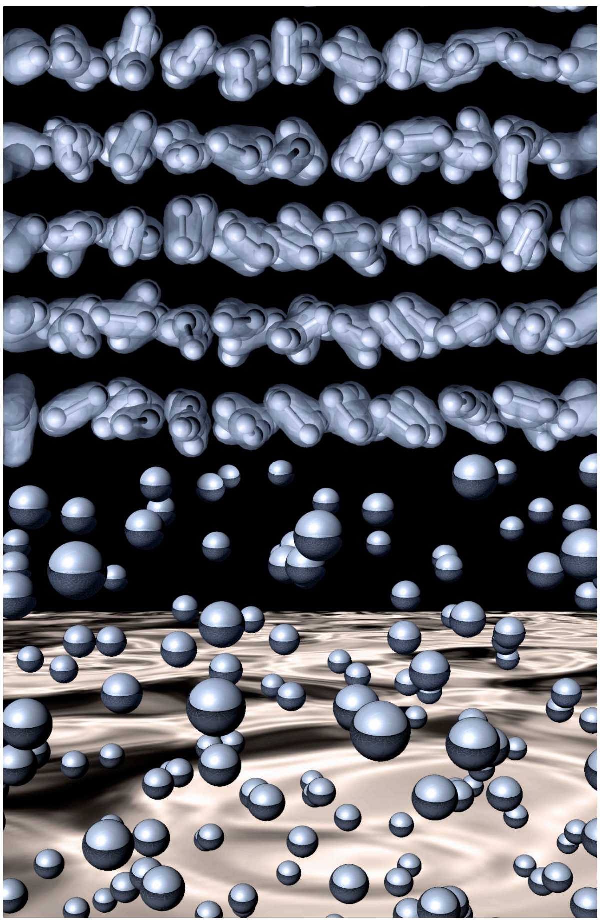
Qbox

Code Contacts: Francois Gygi and Erik Draeger, LLNL

What is it?

Qbox is a massively parallel C++ implementation of First-Principles Molecular Dynamics. It is under development at the LLNL Center for Applied Scientific Computing (CASC) by Francois Gygi and collaborators. Qbox implements the plane-wave pseudo-potential method within Density Functional Theory (DFT), and is routinely used on many LLNL platforms for simulations of condensed matter subjected to extreme conditions, as well in applications to nanotechnology and biochemistry.

Qbox features an advanced XML-based interface that allows it to interact easily with other simulation codes. This can be used to run multi-scale simulations, coupling different physical models of matter, such as DFT and the more accurate Quantum Monte Carlo (QMC) method.



The figure shown at the left was recently used on the cover of the October 7, 2004 edition of the journal Nature, and illustrates the transition from a molecular solid (top) to a quantum liquid (bottom) expected to occur in hydrogen under high pressure. Livermore scientists used the ab initio molecular dynamics code GP (precursor of Qbox) to discover a new melt curve of hydrogen, resulting in the possible existence of a novel superfluid - a brand new state of matter.

LLNL researchers present the results of ab initio calculations of the hydrogen melt curve at pressures up to 2 million atmospheres. The measurement of the high-pressure phases of hydrogen has been the focus of numerous experiments for nearly a century. However, the phase boundary that separates the solid and the liquid has remained relatively unknown.

The team's calculations not only predict a maximum in the melt line, but also provide a microscopic model showing its physical origin in changes in the intermolecular interaction - significantly different from earlier models. Based on their new understanding for the physics behind the melting of hydrogen, the researchers are able to propose new experiments to measure the solid-liquid phase boundary. [For more information, please go to www.nature.com]

What is the benefit from scaling it to large number of processors?

Qbox is currently used in production runs on a 23 TFlops, 4000-processor platform installed at LLNL. Scalability to full machine size has been demonstrated in simulations involving over 3000 atoms. BG/L offers the exciting possibility of running coupled simulations on a scale never before possible, thus considerably enhancing the accuracy of first-principles simulations.

Among the problems that will become tractable with BG/L, some are of particular relevance to the DOE/NNSA Stockpile Stewardship Program. They include the computation of solid-liquid phase transformations of heavy metals at high pressure and high temperature. These simulations are intractable today, even using terascale computers. Other problems of interest include the study of the growth mechanisms of nanoparticles, and enzymatic reactions in biomolecules.

How has the code been optimized on BG/L?

Written in ISO-C++, Qbox makes extensive use of efficient parallel numerical algebra libraries, as well as optimized numerical kernels that exploit the double FPU units of the BG/L PowerPC processors.

How has the code been tested for scaling and performance?

Scalability tests on BG/L show that Qbox can achieve a 3x speedup when solving a given problem on 16384 nodes instead of 4096 nodes. This represents a 75% parallel efficiency. Further optimization is under way.

Does a demo exist? If so, what is it and what does it need?

Not at this time.

Description of demo:

N/A

Who will run the demo at SC04?

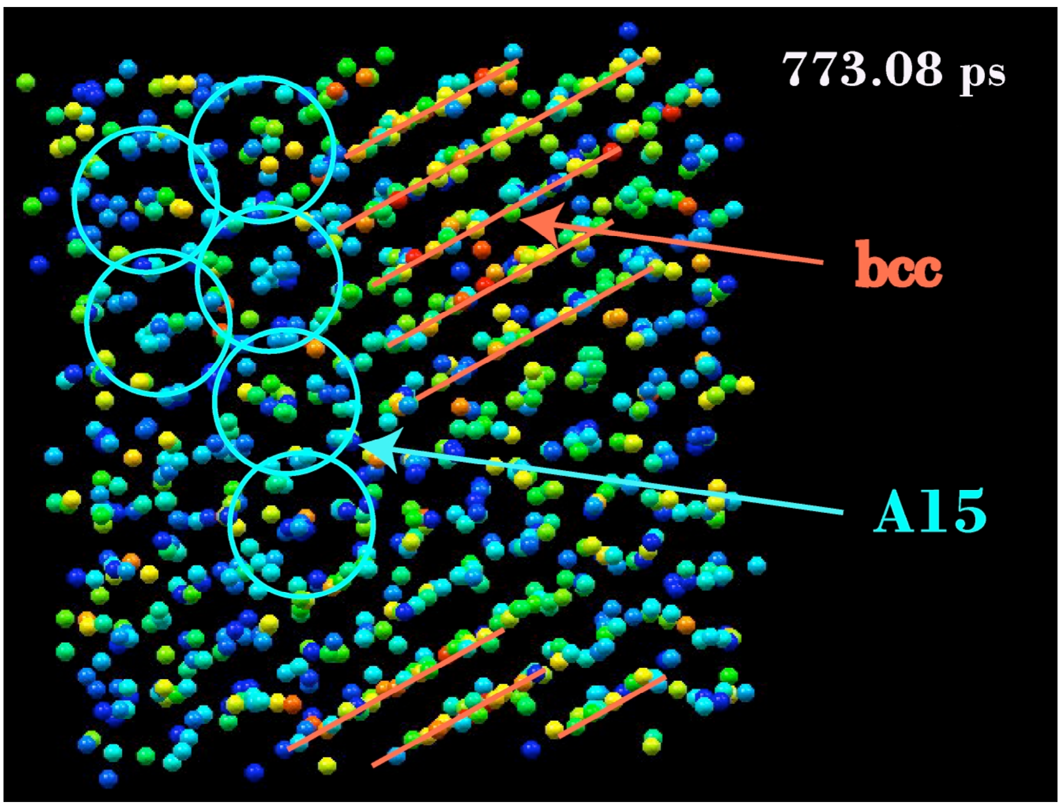
N/A

ddcMD

Code Contacts: Fred Streitz, Jim Glosli and Mehul Patel, LLNL

What is it?

ddcMD is a scalable, general purpose code for performing classical molecular dynamics (MD) simulations using the highly accurate MGPT potentials. These semi-empirical potentials, which are based on a rigorous expansion of many body terms in the total energy, are needed in order to investigate quantitatively the dynamic behavior of transitions metals and actinides under extreme conditions.



A ddcMD simulation showing unexpectedly slow solidification of a metal undergoing high-temperature compression

What is the benefit from scaling it to large number of processors?

To date, accurate atomic scale simulation of materials behavior has been constrained by the maximum size that can be modeled, as un-physically small simulation cell sizes introduce artificial “size effects” into the dynamics. Scientists must either draw inferences from such small simulations, or make sufficient approximations to the underlying physics (i.e., use a "cheaper" potential) to enable larger simulations. By scaling the simulation to tens of thousands of processors, we will be able to model for the first time the dynamic behavior of transition metals and actinides with results independent of system size, while using the most accurate semi-empirical potentials available. These size independent results are needed in order to develop meso-scale and continuum level models of behavior.

How has the code been optimized on BG/L?

We optimized our algorithm for updates of domain decomposition, which previously had caused a linear scaling with the number of processors. This adverse scaling would have been impossible to diagnose without the large number of processors available for testing at BG/L. Currently the only inefficiency that affects the scaling is load leveling on the processors, which appears to limit us to approximately 90% efficiency. (i.e., about 10% of the total elapsed time is spent in communication.)

How has the code been tested for scaling and performance?

We performed runs using a fixed problem size per processor (weak scaling limit) on up to 16,384 processors (with assistance from Jim Sexton at IBM). The results (shown below for two different problem sizes) demonstrate a very weak dependence on number of cpus. We see no impediment at this time to scaling our simulations beyond even the 8 million atoms used for these test runs – close to the largest simulation of MGPT atoms to date.

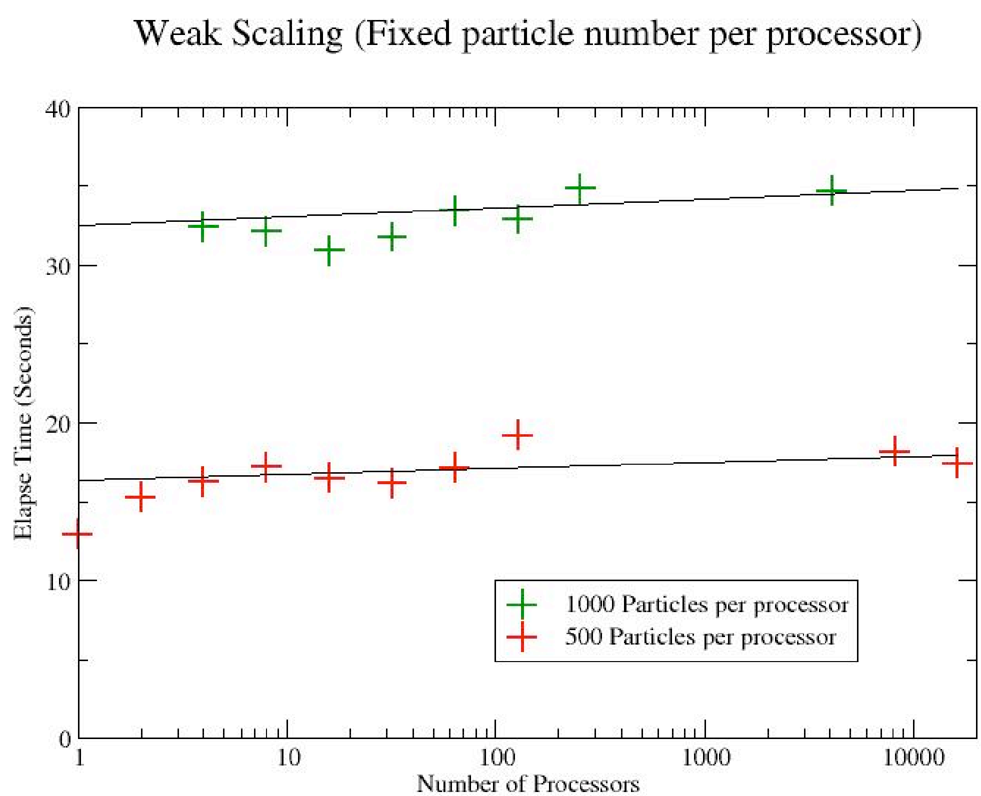
The most recent weak scaling results in the following chart for ddcMD are shown below:

1000 particles per processor:

# CPUs	Elapsed Time
4	32.5
8	32.1
16	31.0
32	31.9
64	33.4
128	32.9
256	35.8
4096	34.7

500 particles per processor:

# CPUs	Elapsed Time
1	12.8
2	15.3
4	17.3
16	16.5
64	17.2
128	19.2
8192	18.2
16384	17.5



Does a demo exist? If so, what is it and what does it need?

Not at this time.

Description of demo:

N/A

Who will run the demo at SC04?

N/A

MDCASK

Code Contacts: Alison Kubota and Tom Spelce, LLNL

What is it?

MDCASK simulates the motion of large collections of individual atoms using the classical laws of Newtonian mechanics and electrostatics. The basic features of the code, as in any "classical" (as opposed to "quantum mechanical") molecular dynamics (MD) code, are (1) an algorithm for the integration of the equations of motion, (2) an inter-atomic potential, and (3) boundary conditions and constraints.

For a given problem the code first defines initial positions for the atoms (such as in lattices for crystalline solids), calculates the forces on each atom using the inter-atomic potential and atom positions, updates the velocities and then uses the new velocities to obtain new positions for the atoms. This cycle is repeated to evolve the system over time. MDCASK is capable of using a wide variety of inter-atomic potentials that allows for the simulation of metals, semiconductors, insulators, glasses, etc. Each atomic material and spatial configuration type requires its own potential. These potentials are derived from atomic theory and quantum mechanical, "ab initio" calculations.

What is the benefit from scaling it to large number of processors?

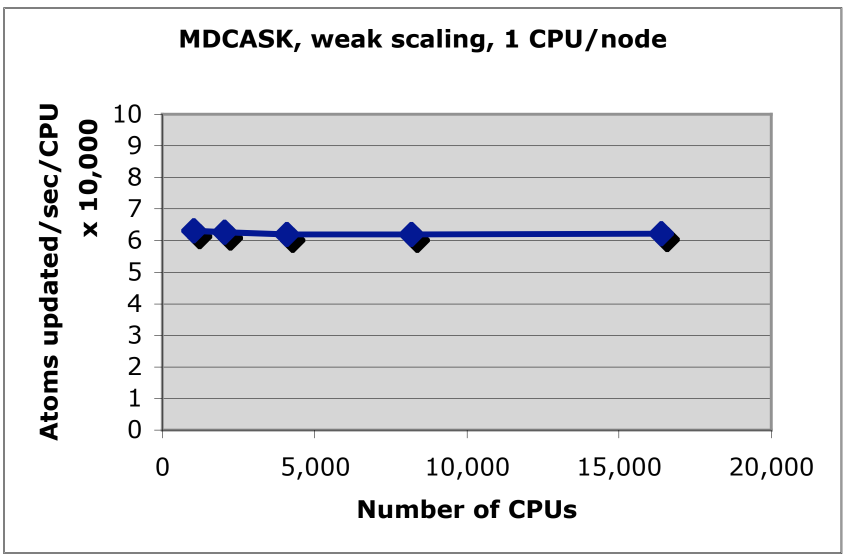
It is the specifics of atomistic behavior that gives rise to phenomenon at the meso- and macroscopic scale that are in turn responsible for the wide range of material properties important for science and industry. Larger computer systems such as BG/L allow scientists to span the gap between the microscopic scale of individual atoms to the meso-scale, thereby providing critical validation of meso- and macroscopic models of material properties. There are also some phenomena, such as the competition between different possible lattice structures during re-solidification that require very large collections of atoms to properly represent. At the largest processor counts, it will also be possible to perform full, 3D simulations of hydrodynamic instabilities important to the ICF program without any of the approximations inherent in more commonly used fluid dynamics simulations.

How has the code been optimized on BG/L?

In some sense, it was BG/L that was designed to optimally meet the needs of classical MD simulation codes such as MDCASK. Using a "spatial" decomposition to divide a large collection of atoms into small blocks, MDCASK inherently has a very high ratio of computation to communication. BG/L provides a very high bandwidth, low latency communication network to nearest neighbors, which allows the code to scale efficiently to tens of thousands of processors. Single CPU performance has increased four-fold via optimization techniques available in the IBM compilers. Further performance tuning is underway which will enable the use of IBM's optimized math libraries.

How has the code been tested for scaling and performance?

A "weak scaling" test has been conducted in which a constant workload per processor (of about 250,000 atoms) was tested in powers of two from one processor to 16K(16,384) processors. Over this very large range of processor counts, the runtime remains constant. This excellent scaling behavior is a powerful validation that the design objectives of BG/L to service the needs of classical MD simulation codes such as MDCASK have been met in full. Continued perfectly linear scaling to 64K(65,536) processors is expected when the full BG/L system is installed at LLNL in June 2005. Scaling tests to ascertain the speedup possible from using both processors on a node are now in process.



Does a demo exist? If so, what is it and what does it need?

Not at this time.

Description of demo:

N/A

Who will run the demo at SC04?

N/A

ParaDiS

Code Contacts: Vasily Bulatov and Gregg Hommes, LLNL

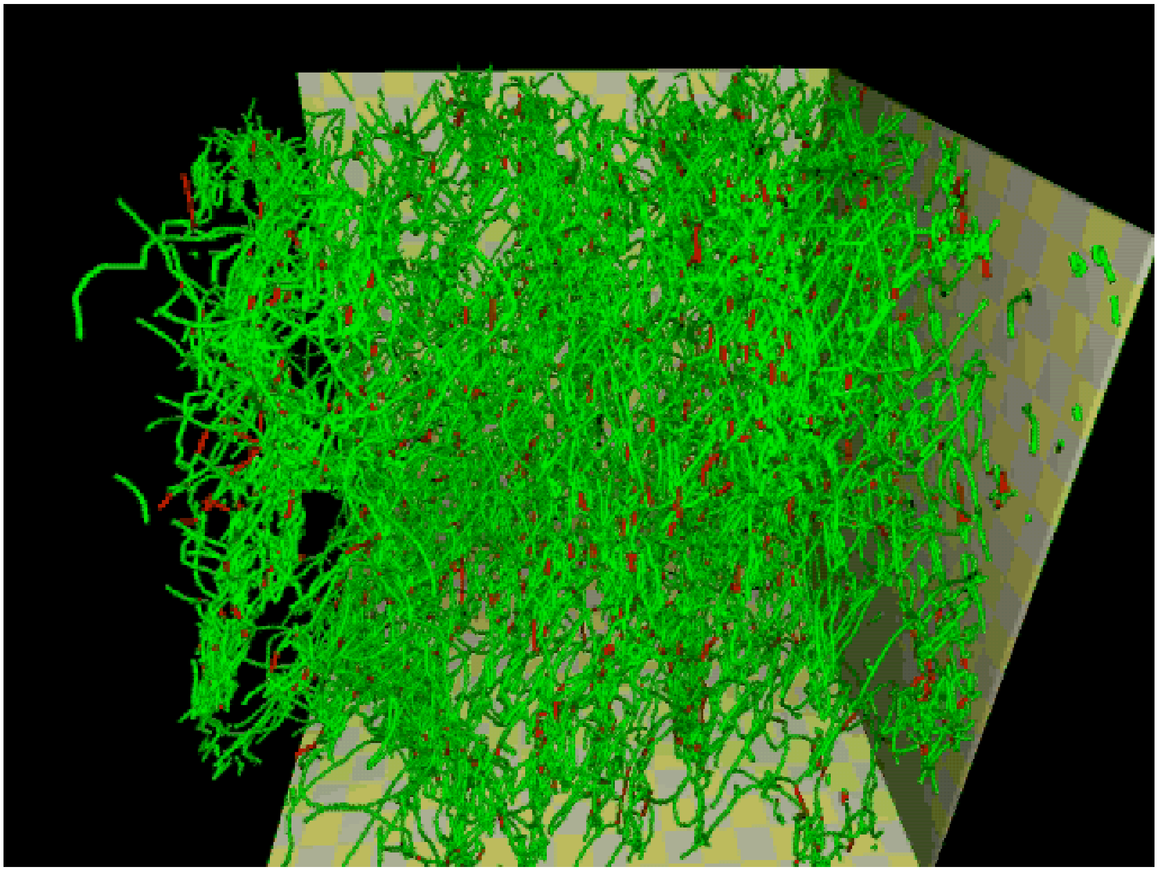
What is it?

ParaDiS (for ***Para****llel* ***Dis****location* ***S****imulator*) is a new code developed at LLNL for direct computation of plastic strength of materials by tracking simultaneous motion of millions of dislocation lines. Simulations using ParaDiS are closing the computational performance gap long recognized to prevent physicists and materials scientists from understanding the fundamental nature of self-induced strengthening (or hardening) and the origin of intricate patterns which dislocation spontaneously form under mechanical straining. The code is primarily written in C and uses the MPI library for communication among the processors.

ParaDiS relies on a line-tracking model that only considers the defects and not the rest of the material. Compared to the various mesh-based approaches, the line-tracking model cuts down the number of degree of freedom dramatically but this saving comes at a price: it now takes considerable effort to track the constantly evolving topology of the dislocation network. The resulting bookkeeping can quickly become horrendously complicated making it nearly impossible to write a working parallel code. ParaDiS achieves an important breakthrough in topology handling by using a minimal set of (irreducible) topological operators for all of its network bookkeeping.

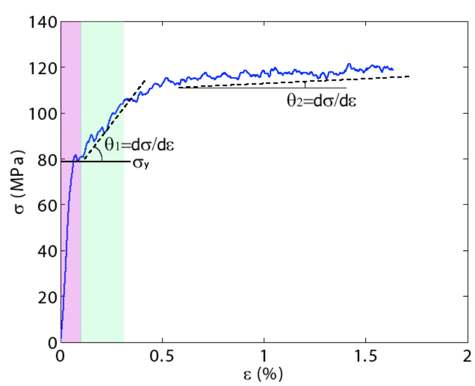
Another serious challenge is a natural tendency of dislocation lines to cluster in space (owing to the long-range interactions among the lines) and develop highly heterogeneous distributions of degrees of freedom making it difficult to achieve a good load balance. To maintain scalability ParaDiS recursively partitions the problem domain first in X, then in Y, and finally in Z dimensions. At regular intervals, ParaDiS re-evaluates the computational load and shifts the domain boundaries to maintain a good balance.

Because dislocation interaction is long ranged, any two line segments interact with each other in a ParaDiS simulation. For computational efficiency, all segment-segment interactions are partitioned into *local* and *remote* contributions, based on proximity of the interacting segments. The *local* interactions are computed explicitly for each local segment pair, while the effect of all *remote* segments in a single cell are lumped together into a super-segment contribution, using a Fast Multipole algorithm. Still, evaluation of forces among dislocation segments typically takes more than 80% of compute time.



A snapshot of a line dynamics simulation in ParaDiS.

Shown above is a snapshot of a typical line network configuration obtained in a ParaDiS simulation. Through a series of such configurations, dislocation lines move around, interact, multiply and recombine. Simulations of this kind provide wealth of information about various scenarios of dislocation behavior both at the level of individual dislocations and at the level of large collectives of interacting dislocations. Simultaneously, the overall strength of the simulated material is directly computed as a function of strain (see figure at left). ParaDiS simulations are now being used for direct comparison with experimental data obtained under the same straining conditions. Once validated against experiments, ParaDiS simulations will be used to accurately predict the behavior of materials under conditions not accessible to experimental measurements, such as under very high pressure, temperature and strain rates.



Plastic strength σ (in MPa) of a simulated material as a function of strain ε.

What is the benefit from scaling it to large number of processors?

Among various challenges worthy of large-scale computational attack, understanding the nature of strain hardening and dislocation patterning in metals is arguably the most famous and holds a special, nearly sentimental value among researchers in the area of material strength. If direct line dynamics simulations can be shown to accurately reproduce dynamic hardening transitions that occur naturally during crystal deformation[1], even skeptics will be convinced that the microscopic physical theory of crystal strength has arrived in the form of line dynamics. With this in mind, we decided to gear ParaDiS towards a single large-scale "hero" simulation that will cover the length and time scales sufficient to observe the hardening transitions that occur naturally, as a result of collective motion and rearrangement of dislocations. Careful estimates show that to be able to naturally account for hardening and dislocation patterning and avoid "small volume" artifacts, the model should include from 1M to 100M dislocation segments. Furthermore, the evolution of such large dislocation groups will have to be traced over millions of time steps, to reach the strain levels at which the hardening transitions are observed. Line dynamics capabilities available up to now at LLNL and elsewhere stop short of these target performance figures by some 2-3 orders of magnitude. Massively parallel computing is the only viable pathway to closing this performance gap.

In our recent simulations we identified an interesting strategy for planning and executing the simulations. In the course of deformation, dislocations multiply increasing their numbers by 2-3 orders of magnitude. For this reason, it is possible and sufficient to start with a relatively small model and let it grow on a small machine. Then, following a steadily growing number of dislocations, the job should be moved to progressively larger machines. This sort of progression worked very well when we scaled our simulation up from 12 to 100, then to 200 and, eventually, to 1,500 CPUs on MCR. The same strategy is now applied on the growing Blue Gene /L machine, which is planned to have 131,072 processors when delivered to LLNL in 2005. On Blue Gene/L ParaDiS will span, in a single simulation, length scales from atomistic (nanometers) to visible (multiple microns) and time scales from picoseconds to seconds and beyond. This computational resource will be sufficient to directly compute, for the first time ever, the plastic strain of a material up to the strain levels achieved in real life material applications.

How has the code been optimized on BG/L?

Primary optimizations have simply been through the use of compiler options, although significant modifications were made to allow the code to execute with the limited memory available on the nodes of BGL. No explicit task mapping was done to map the MPI tasks to the BGL torus, however the runs were executed in a configuration that allocated the ParaDiS tasks in the most optimal fashion.

---

[1] Strain hardening is responsible for some well-known facts of everyday life, such as why an aluminum paper clip eventually breaks after bending it back and forth several times.

How has the code been tested for scaling and performance?

To measure strong scaling, a set of runs was done in which an identical problem was run an identical number of steps at processor counts of 4K and 8K nodes of BGL in co-processor mode. At first glance, the scaling numbers were poorer than expected, however, further analysis revealed that the results were being skewed by the dynamic load-balance capability of the ParaDiS code. Given a specific initial problem, the load-balance of the problem decreases as the number of processors is increased. The ParaDiS code dynamically adjusts to even out the load balance and settle into the optimal work distribution. However, the scaling tests have revealed that the current load-balancing mechanism requires longer to converge on the optimal distribution as the number of processors is increased. Thus, comparing runs of the same number of time steps at different processor counts resulted in skewed results.

The initial scaling runs show a speedup just over 1.5x when doubling the processor count from 4096 to 8192 while the load-balance at 8192 processors was significantly lower than that on the 4096 processor run. Subsequent runs have been performed on Thunder (an LLNL Linux cluster) in which similar load-balance between runs was achieved. These runs showed a 7.85x speedup with an 8x increase in processors from 256 to 2048. Based on these results we expect to see much improved scaling results on BGL with runs long enough to achieve more optimal load-balancing.

Does a demo exist? If so, what is it and what does it need?

Not at this time. A movie will be provided.

Description of demo:

N/A

Who will run the demo at SC04?

N/A