Abstract

Understanding the nature of turbulent flows remains one of the outstanding questions in classical physics. Significant progress has been recently made using computer simulation as an aid to our understanding of the rich physics of turbulence. Here, we present both the computer science and the scientific features of a unique terascale simulation of a weakly compressible turbulent flow that includes tracer particles. (Terascale refers to performance and dataset storage use in excess of a teraflop and terabyte, respectively.) The simulation was performed on the Lawrence Livermore National Laboratory IBM Blue Gene/L™ system, using version 3 of the FLASH application framework. FLASH3 is a modular, publicly available code
designed primarily for astrophysical simulations, which scales well to massively parallel environments. We discuss issues related to the analysis and visualization of such a massive simulation and present initial scientific results. We also discuss challenges related to making the database available for public release. We suggest that widespread adoption of an open dataset model of high-performance computing is likely to result in significant advantages for the scientific computing community, in much the same way that the widespread adoption of open-source software has produced similar gains over the last 10 years.

Introduction

Turbulent flows are ubiquitous in nature, arising at scales as small as tabletop experiments of fluids to scales as large as the entire universe. Understanding the nature of turbulent flows remains one of the outstanding questions in classical physics. Significant progress has been recently made using computer simulation as an aid to our understanding of the rich physics of turbulence. Here, we present both the computer science and the scientific features of a unique terascale simulation of a weakly compressible turbulent flow that includes tracer particles. (Terascale refers to performance and dataset storage use in excess of a teraflop and terabyte, respectively.) The simulation was performed on the Lawrence Livermore National Laboratory IBM Blue Gene/L™ system, using version 3 of the FLASH application framework. FLASH3 is a modular, publicly available code designed primarily for astrophysical simulations, which scales well to massively parallel environments. We discuss issues related to the analysis and visualization of such a massive simulation and present initial scientific results. We also discuss challenges related to making the database available for public release. We suggest that widespread adoption of an open dataset model of high-performance computing is likely to result in significant advantages for the scientific computing community, in much the same way that the widespread adoption of open-source software has produced similar gains over the last 10 years.
as interstellar gas clouds, and they play a fundamental role in the mixing, transport, and combustion of fluids. However, despite its importance, turbulent behavior largely remains an unsolved problem, particularly when one is interested in computing detailed properties of a flow, such as the turbulent drag over an airplane wing, or the rate of combustion in a turbulent flame.

Over the last 20 years, scientific knowledge of turbulent flows has grown tremendously, partly because of the use of computers both in the laboratory and in simulation. Beginning in December 2005, the ASC (Advanced Simulation and Computing) Alliances Center for Astrophysical Thermonuclear Flashes at the University of Chicago was one of six groups in all fields of computation and a second is dedicated to scientific applications. The turbulence was also tracked with

The simulation of the FLASH Center (ASC Alliances Center for Astrophysical Thermonuclear Flashes) is the largest weakly compressible, homogeneous, isotropic simulation in existence [1–3]. Approximately 1 week of CPU (central processing unit) time on 65,536 processors in coprocessor mode was used to complete the simulation, which was performed on a uniform mesh of 1,856^3 grid cells, with fully periodic boundary conditions. In coprocessor mode, one BG/L core is used for computation and a second is dedicated to communication.) The turbulence was also tracked with

parallelism while scaling to many tens of thousands of processors.

**Architecture**

The most recent major release of the FLASH code is version 3. The FLASH3 architecture is defined by two entities: the units and the setup tool. FLASH3 is neither a single application nor a single binary. Rather, it is a collection of units combined by the setup tool into an application tailored for specific simulations. The unit is the basic functional entity in FLASH3; it provides well-defined functionality and conforms to a structure that facilitates interactions with other units. All units must conform to a set of inheritance and encapsulation rules. For example, a unit may have subunits to collect self-contained subsets of its functionality into easily identifiable entities. A unit may also have interchangeable implementations of varying complexity, including child implementations that inherit from and override the functionality of the parent.

FLASH units can be broadly classified into five groups: infrastructure, physics, monitor, driver, and simulation (Figure 1). The infrastructure units are responsible for all the housekeeping tasks of the code such as managing the grid that describes the physical domain, making runtime parameters available as needed, and organizing all the I/O (input and output) from the code. The physics units implement algorithms to solve the mathematical equations describing specific physical phenomena. Physics units include hydro (for hydrodynamics) and EOS (the equation of state) [6]. The stir module is responsible for driving turbulent simulations to maintain a steady-state condition against viscous dissipation, which is
structured adaptive grid. The FLASH code scales very well (i.e., experiences a nearly linear speedup with increasing numbers of processors and with problem size) and was chosen in the spring of 2004 as one of the primary applications for the new BG/L platform. The BG/L machine demonstrates the utility of the massively parallel computing paradigm of large numbers of relatively slow and simple processors with limited memory. It provides teraflop-scale computation through duplication of hardware, rather than through hardware complexity. In its largest incarnation at LLNL, the parallelism is an order of magnitude greater than any other available platform. The configuration of BG/L represents a marked deviation from the prevailing practices in high-performance computing and presents opportunities and challenges to application developers and users. Because each BG/L processor offers lower performance and less memory than processors on other platforms, scientific applications attempting to effectively use this platform must be able to exploit fine-grained implementation in the problem setup directory. For example, users can provide their own criteria for refining the mesh by including the appropriate code files with compatible function interfaces in the simulation directory.

**Algorithms**

The large-scale turbulence simulation described in this paper used a uniform Cartesian grid that divides the physical domain into uniformly spaced computational cells. Periodic boundary conditions are applied to all physical variables at the domain edges. Individual computational cells are grouped into equal-sized cubical subdomains, which are then mapped one-to-one to different processors. We refer to these subdomains as blocks. A perimeter of nonphysical guard cells surrounds each block of data, providing it with data either from the neighboring blocks (if the block is in the computational domain interior) or from boundary conditions (if the block lies on the edge of the computational domain).

The hydrodynamics solver used in the simulation is a directionally split piecewise parabolic method (PPM) [7] that solves the Euler equations with an explicit second-order accurate forward time difference [4]. Time-advanced fluxes at cell boundaries are computed using the numerical solution to the Riemann problem at each boundary. Initial conditions for each Riemann problem are determined by assuming the non-advanced solution to be piecewise-constant in each cell. Use of the Riemann solution has the effect of introducing explicit nonlinearity described further below. The *monitoring units* Logfile and Profiler track the progress of an application, while the *driver unit* implements the time-advancement methods and controls the simulation by managing the interaction between the included units. The *simulation unit* is of particular significance because it specifies a given application by defining the initial conditions and simulation-specific runtime parameters.

Each unit has one or more text files called config files. These specify the (sub)unit requirements, such as physical variables, applicable runtime parameters, and the interaction with other code units. The setup script starts building an application by first examining the config file in the simulation unit, which specifies units essential to the simulation. The setup tool then parses the config files of required units to find their physical variables, runtime parameters, and necessary libraries in order to create a cohesive set of files defining an application. Finally, the user compiles the code pieces to produce an executable application for the problem. A user can replace any native FLASH routine by providing an alternative variable with a given variance, weighted by a “driving” factor $\sqrt{1 - f^2}$ for which $f = \exp(-\Delta t/\tau_{\text{decay}})$, and then causes decays in the previous value by an exponential factor $f$. Since the OU process represents a velocity, the variance is chosen to be the square root of the specific energy input rate divided by the decay time $\tau_{\text{decay}}$. In the limit for which the timestep $\Delta t \rightarrow 0$, the algorithm represents a forcing term that is a linearly weighted summation of the old state with the new Gaussian random variable.

Evolving the phases of the stirring modes in Fourier space makes imposing a divergence-free condition relatively straightforward. At each timestep, the solenoidal components of the velocities are projected out, leaving only the noncompressional modes to add to the velocities. The velocities are then converted to physical space by a direct Fourier transform—adding the trigonometric terms explicitly. The transform is, therefore, trivially parallelized. Because the stirring is done in the low modes, the driving involves a fairly small number of modes, and so this decomposition is more efficient than a complete fast Fourier transform (FFT). The FFT would require large numbers of modes (six times the number of cells in the domain), the vast majority of which would have zero amplitude.

The simulation also evolved the movement of massless tracer particles, which are pointlike objects characterized by positions $x_i$ and velocities $v_i$. The characteristic quantities of each particle are defined by the position of the particle and are determined by interpolation from the
into the difference equations of flow and permits the
calculation of sharp shock fronts and contact
discontinuities without introducing significant
nonphysical oscillations into the hydrodynamics. The
flow variables are represented with piecewise parabolic functions, making this scheme a second-order accurate method.

Because theories of turbulence generally assume a
steady state, and because turbulence is inherently a
dissipative phenomenon, we have chosen to drive the
simulation to sustain a steady state. This driving must be
done carefully in order to avoid introducing artifacts into the
turbulent flow. We use a relatively sophisticated
stochastic driving method originally introduced by
Eswaran and Pope [8]. The turbulent velocity fluctuations
are described by Fourier-transforming from the spatial
domain. For each “stirred” mode of the velocity field, an
acceleration is applied at each timestep. The field consists
of three complex phases, with each acceleration mode
evolved by an Ornstein–Uhlenbeck (OU) random
process, which is analogous to Brownian motion in a
viscous medium. An OU process is a time-correlated,
zero-mean, constant-root-mean-square process. Each
subsequent step in the process adds a Gaussian random

grid mesh. The particles move with the flow relative to the
mesh and can travel from block to block, requiring
communication patterns different from those used to
transfer boundary information between processors for
mesh-based data. The implementation in FLASH
directionally splits the movement of particles out of a
block. Consider a particle that moves from \((x_1, y_1, z_1)\) to
\((x_i, y_i, z_i)\). If the new coordinate \(x_i\) is outside the current
block of the particle, it is moved to the appropriate
neighbor along the first dimension. The particle is now
owned by the neighbor, and when examining movement
along the second dimension, the neighboring block treats
it identically to its own particles. This process is repeated
for all the dimensions, until the particle terminates
movement in the correct block. The direction-splitting
halves the number of explicit data exchanges between
processors per timestep from 26 (potentially one exchange
with every neighbor including those along the corners) to
13. The computation required for a particle is performed
by the same processor that evolves the block in which the
particle resides. No effort is made to separately load
balance the particle computations because experience has
shown that the cost of load balancing the particles
outweighs the gain of maintaining a good load balance
massive amounts of data, and the I/O management and
post-processing can easily become a significant challenge.

Because turbulent flow simulations benefit from high
resolution at all locations, the simulation described here
did not require use of the adaptive mesh capabilities of
the FLASH code [9]. Use of a uniform grid reduced the
global communication complexity of the Eulerian part of
the solution, allowing it to scale almost perfectly, as
shown in the weak scaling plot of Figure 2. A weak
scaling study is one in which researchers vary the problem
size and the number of processors simultaneously, such
that the problem size per processor remains constant.
Figure 2 is a plot of the time required to advance the
solution by 50 timesteps; both axes are in log scale. The
two lines show the overall evolution time for the
turbulent flow simulation and the time taken by the tracer
particles to evolve forward in time by advancing their
position and velocity vectors. The number of processors
along the \(x\)-axis grows from 1 to 32,768, and the amount
of work grows in proportion to the number of processors.
The Lagrangian tracer particles, however, do not show
the same perfect scaling as that observed for the grid
evolution, even though their communications pattern is
nearest-neighbor, like that of the Eulerian grid. As
mentioned above, no separate effort is made to load
balance the particles. The complex nature of the turbulent
flow changes the particle distribution during the

Figure 2

Weak scaling of the evolution of the Eulerian mesh and the
Lagrangian particles with the FLASH3 code, for the driven
turbulence problem.

for what is a small fraction of the overall execution time.
The effect of this choice on scaling is discussed below. The
time integration used for the Lagrangian particles in this
simulation uses a predictor-corrector scheme that yields a
second-order accurate solution for uniform timesteps.
Challenges of scale
As mentioned, the BG/L machine deviates from the current parallel computing norm of relatively few but very powerful processors with large memory. This standard model derives from cluster-based computing, which is driven by single-processing-element (PE) performance. The BG/L machine, on the other hand, uses slower processors with limited memory and relies upon finer-grain parallelism to achieve performance. This shift in the computing paradigm presents major challenges for codes that are memory intensive and have components with global communication patterns.

Implementation of multiphysics codes such as FLASH3 usually face trade-offs between modularity, efficiency of memory, and CPU usage. For example, in order to keep various physics solvers in the code independent of one another, and their data management well modularized, multiple copies of some datasets may exist in the code. This problem could be alleviated with dynamic memory management, but this solution causes loss of compiler optimizations and, therefore, loss of computational efficiency. Also, while a single timestep evaluation typically involves only nearest-neighbor communications, some operations are necessarily global, such as updating the timestep selected by the Courant condition. Additionally, scientific calculations generate the idea of anomalous scaling to probe our BG/L turbulence run dataset to gather clues to these issues.

Results
A fundamental mathematical tool commonly used by turbulence researchers to examine the scaling properties of turbulent flow is the $p$th-order structure function $S_p(r)$, which is closely related to the autocorrelation function of the velocity field:

$$S_p(r) = \langle (\vec{v}(\vec{x} + \vec{r}) - \vec{v}(\vec{x}))^p \rangle \propto r^{\zeta_p}.$$ 

Here $\langle \rangle$ denotes an average of spatial locations $\vec{x}$ over all space and $\zeta_p$ are the scaling exponents. The proportionality on the far right-hand side applies over a range of separations $r$ in the inertial regime and is a direct consequence of self-similarity.

A naive computation of $S_p(r)$ involves an average over all spatial points. However, instead of calculating $S_p(r)$ directly, we generate probability distribution functions (PDFs) of each component of the velocity increment $\Delta v = \vec{v}(\vec{x} + \vec{r}) - \vec{v}(\vec{x})$ of each interval $r$ for each dimension for each dimensional velocity component. Because this approach involves only the calculation of the separations
Over the past 20 years, extensive research has been performed on turbulent flows. One of the key insights attained in experiment and in numerical simulation is that Kolmogorov's original theory is incomplete: The hypothesized self-similarity is not observed because the dissipation within the fluid does not occur homogeneously, but instead it is intermittent in both space and time. The resulting scaling laws, which differ from Kolmogorov's theory, are said to describe anomalous scaling.

Kolmogorov's theory relates to turbulence in the same manner as Newton's laws relate to mechanics. They both provide highly successful, though incomplete frameworks whose enormous influence is widely felt. A number of phenomenological theories have been advanced as possible heirs to Kolmogorov's legacy, although it remains unclear which of these (if any) is correct [11, 12]. Working directly from the Navier–Stokes equations presents challenging barriers to standard mathematical analysis techniques; thus, theorists must instead work from plausible assumptions.

Because our techniques of treating hydrodynamic flows away from the strictly incompressible limit is different from the majority of work done in the turbulence community, we also must compare our results with experiment, theory, and previous simulations in order to convince ourselves that weak compressibility does not have a strong influence upon the scientific conclusions we draw from our dataset. In the following section, we use

\[ S_p(r) = \int d\Delta v_p P(\Delta v_p) \Delta v_p^p. \]

Here, we denote the probability distribution function of the velocity increment \( \Delta v_p \), on the increment \( r \) by \( P(\Delta v_p) \).

The Lagrangian tracer particle data can be analyzed in a similar fashion, making use of the \( p \)th-order structure function with respect to time:

\[ S_p(\tau) = \langle \{ \tilde{v}(t + \tau) - \tilde{v}(t) \}^p \rangle \propto \tau^x. \]

While the Lagrangian dataset is significantly smaller, problems in the calculation do arise because each timestep is contained in a different file, making I/O much more costly due to the frequency of accesses than observed for the Eulerian dataset. A similar method of probability distribution function generation is utilized as was described above. The structure-function data generated from this set was then used to verify the Lagrangian tracer particle data.

A useful technique for determining the scaling exponents \( \zeta_p \) was first discovered by Benzi et al. [13], who noted that an extended self-similar region appears when, instead of plotting \( S_p(\tau) \) for different values of \( r \), a plot is used in which the vertical and horizontal axes are broadly extended over one to two more decades in length scale, which permits much more precise determinations of the scaling law exponents.

One can easily demonstrate that if Kolmogorov's original theory were strictly correct, and if dissipation occurred homogeneously, the exponents \( \zeta_p \) would simply equal \( p/3 \). In the inertial regime, the flow is self-similar and no other characteristic length scales are introduced. Therefore, the scaling exponents \( \zeta_p \) follow directly from simple dimensional analysis: Assuming a characteristic velocity \( v \) over a spherical region of size \( r \), the dissipation per unit mass \( \varepsilon \) within \( r \) must scale as \( \varepsilon \propto v^3/r \). If \( \varepsilon \) is homogeneous and the flow is steady, one infers that \( v \propto r^{3/2} \). Dimensional analysis then reveals that the exponents \( \zeta_p \) are simply equal to \( p/3 \).

We can directly test Kolmogorov's original theory, as well as more recent theories, by applying ESS to the data from the BG/L run. The results are shown in Figure 3. Here, we observe the scaling exponents \( \zeta_p \) versus \( p \) (derived from fits of the structure function) as a function

![Figure 3](image-url)

**Figure 3**

Comparison of anomalous exponents for theory and experiment. Anomalous scaling exponents (for \( p = 10 \)) are shown for the BG/L run (points with error bars) and are compared against predictions of Kolmogorov's original 1941 theory (red straight line), as well as the predictions of She–Leveque (blue curved line). When plotted at the resolution of this graph, the experimental results of Benzi...
structure functions. This technique is referred to as extended self similarity (ESS). The scaling exponents derived from ESS are identical to the \( f_p \) above, because the third-order structure function \( S_3(r) \) is exactly proportional to \( r \), as was first rigorously demonstrated by Kolmogorov in his famed 4/5th law. ESS has an advantage because the self-similar scaling regime is of separation \( r \) along \( x \)—one of the three principal components of the longitudinal structure function. As is easily seen, Kolmogorov's 1941 theoretical prediction [10] deviates systematically from both Benzi's experimental analysis [13] and our simulation. In contrast, the more recent theory of She and Leveque [12] is in close agreement with both experiment and simulation. We conclude that despite being weakly compressible, the results of our simulation agree very well with previous incompressible results and suggest that our dataset can be used to explore a wide variety of issues in turbulent flows.

**Visualization**

Current visualization efforts have focused on exploration of the dataset. In order to meet analysis needs, this effort has utilized a combination of open-source solutions (e.g., the parallel visualization tools ParaView and VisIt) and more specific solutions. Processing has been performed to augment the dataset with post-simulation content such as scalar magnitudes of the local vorticity and divergence vectors. This data is now being stored as part of the dataset that will be made publicly available. Visualization challenges have demonstrated the need for large computational resources beyond the initial simulation, for data-filtering and data-selection tools to reduce the data to appropriate amounts for viewing, and for investment in remote visualization solutions. The generation of the derived data fields for vorticity and divergence required 1.5 days on a 64-node cluster. Visualizing all 16 million Lagrangian particle traces yields no useful insight. Instead, filtering and cutting (i.e., slicing) the data and using nonlinear shading functions to highlight features of interest are crucial (Figure 4 and Figure 5). Future efforts will involve the integration of workflow tools for the construction of analysis pipelines and the availability of specific tools to enable users to visualize the data.

**Open dataset model**

The public release of the dataset will perhaps be one of the most far-reaching consequences of this effort. Performing simulations on machines with teraflop or petaflop performance is by necessity a major challenge that few research efforts have both the technical capacity and the computational resources to address. Even when successful, the appropriate datasets are not typically released publicly, which limits their impact on the few researchers who have access to the data. Here, we draw inspiration from the open-source movement and advocate an open model for high-performance computing datasets—i.e., that large-scale high-performance
smaller machines, such as the widely available small-scale Linux clusters. As a concrete example, consider a traditional high-performance computing, time-dependent simulation of a set of partial differential equations discretized explicitly onto a uniform mesh of $N^D$ cells in $D$ dimensions. In this case, because of the Courant limitation on the timestep, the number of CPU operations to complete the simulation will scale as $N^{D+1}$ even though the number of operations to be performed for a local analysis scales only as $N^D$. On the basis of this analysis, it is easy to see that local post-processing of a dataset will be decreased roughly by a factor of $N$ over the full simulation. In state-of-the-art three-dimensional (3D) simulations, which is a 244-processor AMD Opteron** computing cluster used for local processing of the data. High-performance computing consists of a hierarchy of computation, ranging from the fastest machines in the world used by a handful of researchers to smaller machines accessible to a much wider community. The public release of the FLASH Center dataset will amplify the impact of our turbulence effort significantly by allowing us to share our results with many levels on the computational hierarchy. Moreover, if our data dissemination effort is successful, it will provide both a fruitful model and concrete software tools that will encourage computational scientists to release their datasets—i.e., that large-scale high-performance computing datasets should be made openly accessible. While dissemination and analysis of high-performance computing datasets pose numerous technical challenges, we argue that these technical challenges can be addressed and overcome and are outweighed by the potential scientific gains to be achieved by sharing the datasets.

Historically, the movement known today as open source originated with the IBM user group SHARE in the 1950s and was later championed by Richard Stallman and the Free Software Foundation beginning in the early 1980s. However, it first received widespread attention following Eric Raymond’s highly influential book *The Cathedral and the Bazaar* [14], which inspired the Netscape company to release a version of its browser software in 1997. The open-source model of distributing source code has had an enormous impact on computing in both the scientific and the business world. At the heart of the open-source paradigm is the elimination of barriers to the exchange of source code, thereby creating a free marketplace of ideas wherein communication and resource sharing are encouraged. In the years following Raymond’s essay, the number of open-source code development projects quickly increased, and companies such as IBM, Hewlett-Packard, Oracle, and CollabNet have incorporated the open-source paradigm into their business models. Perhaps even more far reaching is the wide array of other endeavors inspired by the open-source paradigm—including examples ranging from open publication projects such as Wikipedia**, the Creative Commons, and the Science Commons to open educational curricula, such as that facilitated by the OpenCourseWare project at MIT, as well as numerous other projects.

We reiterate our belief that the same open-minded thinking that gave rise to the open-source model of distributing source code can also be profitably applied to large-scale high-performance computing datasets. In other areas of science in which experimental and observational datasets have been made open (i.e., available) to researchers, enormous strides have been made. Particularly successful examples include the Human Genome Database, the Sloan Digital Sky Survey, and the Particle Physics Data Grid. The impact of these projects has been enormous in their respective fields. More than 3,000 astrophysics articles refer to the Sloan Digital Sky Survey alone. However, despite the significant advantages to be gained by sharing their data, the high-performance computing community has lagged behind their colleagues in opening access to their datasets. We believe this is in part due to some lingering fundamental misconceptions about the nature of simulation and analysis. Once these issues are recognized, it becomes clear that opening access to high-performance computing datasets presents a major opportunity for the growth of computational sciences in the near future.

The first common misconception is that large-scale computational datasets, which required the largest supercomputers in the world to compute, would also necessarily require the largest supercomputers to store and to analyze. However, this is not necessarily the case. For a wide class of problems that are CPU bound (and not memory bound), the results of large-scale simulations performed on the largest machines can be analyzed on
simulations, \( N \) approaches 2,000 to 4,000, and the number of CPUs utilized during the full simulation may be as large as \( 10^5 \). However, by simply scaling these numbers, it is clear that the post-processing analysis of this same dataset can be completed in a roughly equal amount of wall-clock time on just a few hundred processors. Significantly, as we suggested, analyses of large-scale datasets can be completed on small-scale clusters. In fact, the analysis shown in this paper was completed on several small-scale Linux clusters at the University of Chicago, all with fewer than 256 processors.

Another common misconception is that distribution of large-scale datasets could possibly use a significant amount of bandwidth and take very long times to transfer. However, this is also not a fundamental deterrent. For instance, one solution is simply to sidestep the transfer of the datasets and collocate storage and computation, allowing open access to the system to all interested parties. We believe that this is the natural solution for terabyte and larger datasets, and we have adopted the collocated model in establishing a new datastore system at the University of Chicago. Our turbulence dataset will be served to the community from the Computation Institute (CI) large datastore at the University of Chicago. The system is a scalable high-performance storage resource that has 75 TB of storage configured in an 8:2 RAID array, allowing up to 48 drives to fail with no impact on performance, stability, or reliability. The system can deliver a sustained throughput of 3 GB/s. It can also be scaled to 480 TB of storage. Five I/O servers are connected to the storage system by five fiber channels and then connected to outside the datastore via 1-Gb/s Ethernet connections to the 10-Gb/s I/O servers.

I-WIRE is a dark fiber communications infrastructure interconnecting organizations that include Argonne National Laboratory, the University of Illinois, the University of Chicago, the Illinois Institute of Technology, Northwestern University, and the Illinois Century Network Chicago hub. Collocated with the storage resource is the CI TeraPort compute resource, which is a modular, application-specific tool for astrophysical simulations that scales well to massively parallel environments. The BG/L configuration of more than 64,000 processors with limited memory and computing power posed special challenges. Specialized I/O routines were developed, and efficient particle-tracking schemes were implemented. Despite the demanding conditions, preliminary analysis of the results indicates a dataset of very high quality. Results of extended self-similarity analysis on the data show support for new theories of turbulent structure and match well with experimental evidence. The massive dataset will be released to the public, thereby further expanding the impact of this simulation.

**Acknowledgments**

This work is supported in part at the University of Chicago by the U.S. Department of Energy under Contract B523820 to the ASC/Alliances Center for Astrophysical Thermonuclear Flashes. All or portions of this manuscript have been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory under Contract No. DE-AC02-06CH11357 with the U.S. Department of Energy. We gratefully acknowledge the generous computational resources supplied by the DOE/ASC program at the LLNL, as well as additional resources supplied by Argonne National Laboratory and the IBM T. J. Watson Research Center.

In addition, we thank a number of highly talented individuals who supported our effort, and without whom this project would not have been possible. At the LLNL, we thank Steve Louis, Adam Bertsch, Richard Hedges, Jean Shuler, Dick Watson, Mike McCoy, and the LC Computing Staff. At Argonne National Laboratory, we thank Ed Jedlicka and Susan Coghlan. At IBM, we thank Bob Walkup, James Sexton, and Sameer Kumar.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of the University of Chicago, Linus Torvalds, Wikimedia Foundation, Robert T. Fisher  
DOE ASC Center for Astrophysical Thermonuclear Flashes, and Department of Astronomy and Astrophysics, The University of Chicago, Chicago, Illinois 60637 (rtfisher@uchicago.edu). Dr. Fisher received his Ph.D. degree in physics from the University of California at Berkeley in 2002 and was a postdoctoral fellow at Lawrence Livermore National Laboratory. He is currently a member of the DOE/ASC FLASH Center astrophysics group and a research scientist in the department of astronomy and astrophysics at the University of Chicago. His research interests include the fundamental physics of turbulent flows, type Ia supernovae, star formation, numerical algorithms, and high-performance computing.
Inc., or Advanced Micro Devices, Inc., in the United States, other countries, or both.

References

Received March 15, 2007; accepted for publication March 28, 2007; Internet publication December 5, 2007

Leo P. Kadanoff  Department of Physics and Department of Mathematics, The University of Chicago, Chicago, Illinois 60637 (leop@uchicago.edu). Mr. Kadanoff received his Ph.D. degree in physics from Harvard in 1960. He is a professor of physics at the University of Illinois, Brown University, and the University of Chicago, where he presently has emeritus status. He is President of the American Physical Society. His research interests include the fundamental physics of turbulent flows, interface motion, applied mathematics, and the critical analysis of scientific computing.

Don Q. Lamb  DOE ASC Center for Astrophysical Thermonuclear Flashs, and Department of Astronomy and Astrophysics, The University of Chicago, Chicago, Illinois 60637 (lamb@calder.mail.astro.sunysb.edu). Lamb is the Louis Block Professor in astronomy and astrophysics and a faculty member at the Enrico Fermi Institute at the University of Chicago. He is the director of the DOE/ASC Alliance FLASH Center. He is a Fellow of the American Physical Society and the American Academy of Arts and Sciences. His research interests include type Ia supernovae, gamma-ray bursts, galaxy clusters, and high-performance computing.

Anshu Dubey  DOE ASC Center for Astrophysical Thermonuclear Flashs, and Department of Astronomy and Astrophysics, The University of Chicago, Chicago, Illinois 60637 (dubey@flash.uchicago.edu). Dr. Dubey received her Ph.D. degree in computer science from Old Dominion University in 1993. She joined the University of Chicago as a Research Associate in astronomy and astrophysics in 1993, and then she moved to the FLASH Center in 2002. At the FLASH Center, she leads the code group responsible for design, development, and support of the FLASH code.

Tomasz Plewa  DOE ASC Center for Astrophysical Thermonuclear Flashs, and Department of Astronomy and Astrophysics, The University of Chicago, Chicago, Illinois 60637 (tomek@uchicago.edu). Dr. Plewa is a Senior Research Associate at the University of Chicago. He obtained his Ph.D. degree in theoretical astrophysics from the Warsaw University and spent several years at the Max-Planck-Institut für Astrophysik, Garching, where he worked on the development of adaptive methods in application to astrophysical flows. His research focuses on problems related to core-collapse and thermonuclear supernovae, mixing processes, laboratory astrophysics, code validation, and development of high-performance multiphysics simulation tools.

Alan Calder  DOE ASC Center for Astrophysical Thermonuclear Flashs, and Department of Astronomy and Astrophysics, The University of Chicago, Chicago, Illinois 60637 (acalder@mail.astro.sunysb.edu). Mr. Calder received his Ph.D. degree from Vanderbilt University in 1997. He subsequently held research positions at NCSC and the University of Chicago and is now a Senior Research Scientist at SUNY, Stony Brook. His research interests include the physics of supernovae, large-scale computing, and validation of simulation codes with respect to experimental data.

Fausto Cattaneo  DOE ASC Center for Astrophysical Thermonuclear Flashs, and Department of Astronomy and Astrophysics, The University of Chicago, Chicago, Illinois 60637 (richp@flash.uchicago.edu). Mr. Rich received his M.S. degree in computer science from the University of Chicago in 2006. He is currently a Scientific Programmer in the code group of the ASC FLASH Center. His research interests include high-performance computing, parallel architectures, and terascale data analysis and post-processing.

Paul M. Rich  DOE ASC Center for Astrophysical Thermonuclear Flashs, The University of Chicago, Chicago, Illinois 60637 (richp@flash.uchicago.edu). Dr. Rich received his Ph.D. degree from Vanderbilt University in 1997. He subsequently held research positions at NCSC and the University of Chicago and is now a Senior Research Scientist at SUNY, Stony Brook. His research interests include the physics of supernovae, large-scale computing, and validation of simulation codes with respect to experimental data.
Peter Constantin  Department of Mathematics, The University of Chicago, Chicago, Illinois 60637 (const@cs.uchicago.edu). Dr. Constantin received his Ph.D. degree in mathematics in 1981 at the Hebrew University of Jerusalem. His research interests include turbulence and turbulent mixing, dynamical systems, fluid mechanics, and nonlinear analysis.

Michael E. Papka  DOE ASC Center for Astrophysical Thermonuclear Flashes, and Department of Computer Science, The University of Chicago, Chicago, Illinois 60637 (papka@flash.uchicago.edu). Mr. Papka is the Deputy Associate Laboratory Director for Computing and Life Sciences at Argonne National Laboratory and professor of computer science at the University of Chicago. His research interests include applied analysis in fluid dynamics, plasmas, and astrophysics.

Shimon M. Asida  DOE ASC Center for Astrophysical Thermonuclear Flashes, The University of Chicago, Chicago, Illinois 60637 (asida@phys.uchicago.edu). Dr. Asida is a lecturer at the Racah Institute of Physics of the Hebrew University of Jerusalem where he received his Ph.D. degree in physics in 1998. He was a postdoctoral fellow at the University of Arizona at Tucson and had a sabatical at the FLASH Center at the department of astronomy and astrophysics at the University of Chicago. His research interests include unstable flows, convection in stars, and type Ia supernovae modeling.

Chad C. Glendenin  DOE ASC Center for Astrophysical Thermonuclear Flashes, The University of Chicago, Chicago, Illinois 60637 (chad@uchicago.edu). Mr. Glendenin received a bachelor’s degree from Yale University and a master’s degree from the University of Chicago. He is currently a member of the DOE/ASC Alliance FLASH Center visualization group and a Ph.D. student at the University of Chicago department of geophysical sciences.

Katie Antypas  DOE ASC Center for Astrophysical Thermonuclear Flashes, The University of Chicago, Chicago, Illinois 60637 (kantypas@lbl.gov). Ms. Antypas received her M.S. degree in computer science from the University of Chicago. She is currently a high-performance computing consultant at NERSC at the Lawrence Berkeley National Laboratory. She previously worked in the code group at the DOE/ASC Alliance FLASH Center as a scientific programmer.

Daniel J. Sheeler  DOE ASC Center for Astrophysical Thermonuclear Flashes, The University of Chicago, Chicago, Illinois 60637 (sheeler@flash.uchicago.edu). Mr. Sheeler received his M.S. degree in computer science and his B.A. degree in physics both from the University of Chicago. He currently works as a Scientific Programmer on the ASC FLASH project in the department of astronomy and astrophysics at the University of Chicago.

Ian Foster  Department of Computer Science, The University of Chicago, Chicago, Illinois 60637 (foster@cs.uchicago.edu). Dr. Foster received his Ph.D. degree in computer science from Imperial College, London, in 1988. He is currently Director of the Visualization group at the University of Chicago.

Pramod K. Joshi  DOE ASC Center for Astrophysical Thermonuclear Flashes, The University of Chicago, Chicago, Illinois 60637 (pramodk@flash.uchicago.edu). Dr. Joshi is a Lecturer in the Computer Science and Engineering Program at the University of Washington.

Crossref Reference(s)
Kathleen Conrooy

0.00 - University of Chicago

Do you know this publication? Be the first to give feedback

Add a comment

ADD COMMENT  CANCEL

Data provided are for informational purposes only. Although carefully collected, accuracy cannot be guaranteed. The impact factor represents a rough estimation of the journal’s impact factor and does not reflect the actual current impact factor. Publisher conditions are provided by RoMEO. Differing provisions from the publisher’s actual policy or licence agreement may be applicable.