Title:

369 TFLOP/S MOLECULAR DYNAMICS SIMULATIONS ON THE ROADRUNNER GENERAL-PURPOSE HETEROGENEOUS SUPERCOMPUTER

Author(s):

Sriram Swaminarayan (CCS-2)
Timothy C. Germann (T-12)
Kai Kadau (T-17)
Gordon C. Fossum (IBM Corporation)

Intended for:

Gordon Bell Prize Finalist submitted to SC08 Supercomputing Conference (Austin, TX, November 15-21, 2008)

Los Alamos
NATIONAL LABORATORY
EST. 1943

369 Tflop/s Molecular Dynamics Simulations on the Roadrunner General-Purpose Heterogeneous Supercomputer

Sriram Swaminarayan
CCS-2, Mail Stop B296
Los Alamos National Laboratory
Los Alamos, NM 87545 USA
+1-505-667-8647

Timothy C. Germann
T-12, Mail Stop B268
Los Alamos National Laboratory
Los Alamos, NM 87545 USA
+1-505-665-9772

Kai Kadau
T-14, Mail Stop G756
Los Alamos National Laboratory
Los Alamos, NM 87545 USA
+1-505-665-0354

Gordon C. Fossum
IBM Corporation, Mail Stop 9065G018
11501 Burnet Road
Austin, TX 78758
+1-512-838-1629

ABSTRACT

We present timing and performance numbers for a short-range parallel molecular dynamics (MD) code, SPaSM, that has been rewritten for the heterogeneous Roadrunner supercomputer. Each Roadrunner compute node consists of two AMD Opteron dual-core microprocessors and four PowerXCell 8i enhanced Cell microprocessors, so that there are four MPI ranks per node, each with one Opteron and one Cell. The interatomic forces are computed on the Cells (each with one PPU and eight SPU cores), while the Opterons are used to direct inter-rank communication and perform I/O-heavy periodic analysis, visualization, and checkpointing tasks. The performance measured for our initial implementation of a standard Lennard-Jones pair potential benchmark reached a peak of 369 Tflop/s double-precision floating-point performance on the full Roadrunner system (27.7% of peak), corresponding to 124 MFlop/Watt/s at a price of approximately 3.69 MFlops/dollar. We demonstrate an initial target application, the jetting and ejection of material from a shocked surface.

1. INTRODUCTION

The power and heat dissipation challenges encountered with steadily increasing microprocessor transistor densities and clock frequencies have motivated the current trend towards multi-core processors. Dual-core and quad-core commodity processors are becoming commonplace, with continued increases in cores per socket, such as Intel's 80-core Teraflops Research Chip [1], on the horizon. Special-purpose co-processors, or accelerators, have already made significant progress in this direction, most notably the graphics processing units (GPUs) driven by the personal computer and gaming industries. The Cell Broadband Engine™, developed jointly by Sony Computer Entertainment, Toshiba, and IBM, combines a general-purpose PowerPC core (PPE) with eight co-processors, termed Synergistic Processing Elements (SPEs) [2]. First deployed in the Sony PlayStation 3 game console, the Cell's potential usage spans a variety of compute-intensive multimedia, financial, and scientific applications.

In a collaboration between Los Alamos National Laboratory (LANL) and IBM, the first Cell-based supercomputer was assembled at IBM Poughkeepsie in mid-May 2008 and delivery to LANL began in July and is expected to complete in November. Named Roadrunner, it has a theoretical peak performance of 1.3 PFlop/s, achieved a sustained 1.026 Pflop/s on the Linpack benchmark. To smooth the transition of legacy MPI-based codes designed for parallel computer clusters, typically Linux-based, to the novel multi-core and accelerated architectures that are expected to dominate the next generation of supercomputers, Roadrunner has been designed as a Cell-accelerated cluster of AMD Opteron x86 processors.

This additional level of heterogeneity means that the user is faced with managing 3 types of cores (PPE, SPE, and Opteron), each with associated memory and a nonuniform connectivity, leading to interesting programming challenges if one hopes to achieve maximal efficiency. Although the initial porting of codes using a 'function offload model' is more manageable, a more drastic code redesign is often necessary to take full advantage of the potential performance offered by the Cell processors, which contribute over 96% of the aforementioned 1.3 PFlop/s theoretical peak.

We demonstrate that such a redesign is practical and can be worth the effort, achieving ~28.3% of the theoretical peak performance for our redesign of the SPaSM (Scalable Parallel Short-range Molecular dynamics) code. The initial design of SPaSM, by Beazley, Lomdahl, and coworkers [3,4], took place roughly 15 years ago and was targeted at the Thinking Machines Connection Machine 5 (CM-5). Memory and floating-point performance provided the chief design constraints, resulting in an algorithm with frequent interprocessor communication of small messages to conserve memory and avoid redundant computations. This algorithm has stood up well over the years, on architectures ranging from Beowulf clusters [5] to the 212,992-core IBM/LLNL BlueGene/L [6,7], demonstrating nearly perfect weak scaling up to 10^{12} atoms, and an excellent strong scaling for system sizes of at least 10^4-10^5 atoms per processor. However, production simulations are almost always limited by wall-clock time rather than memory constraints, and the steady advance of processor performance by Moore's Law is diminishing the importance of floating-point performance relative to data movement.

Motivated in the longer-term by this general trend, but in the immediate term by the assembly of Roadrunner, we have introduced a major redesign of the SPaSM algorithm. This paper is organized as follows. In Section 2 we briefly summarize the Roadrunner architecture and performance characteristics. Section 3 provides background on molecular dynamics simulations and the historical SPaSM algorithm design. We turn in Section 4 to its (re-)implementation on Roadrunner , first discussing our initial "evolutionary" port to a hybrid Opteron/Cell architecture, and finally the "revolutionary" redesign leading to greatly improved performance, presenting specific performance results. We also discuss two side benefits of this redesign: allowing for much more efficient load balancing and checkpointing. Section 5 describes one of the initial scientific problems we performed on Roadrunner at Poughkeepsie this summer, related to the Richtmyer-Meshkov instability at a shocked surface, a problem that is important for inertial confinement fusion (ICF) design but that is difficult to address using conventional fluid dynamics calculations. Beyond this initial example, the work described here paves the way to tackle a wide variety of other scientifically relevant problems, in solid state physics, fluid dynamics, biology, and even stochastic agent-based modeling that can target areas such as infectious disease spread and other social network-related problems.

2. ROADRUNNER ARCHITECTURE

Roadrunner is a petascale hybrid supercomputer built for Los Alamos National Laboratory (LANL) by IBM. To enable rapid utilization by the broadest range of multiphysics computer codes at LANL, Roadrunner is designed as an accelerated version of a relatively conventional Infiniband-connected cluster of AMD Opteron processors. The 1.3 Petaflop/s performance is primarily provided by the addition of one Cell processor for each Opteron core, to accelerate numerically intensive bottlenecks that can hopefully be isolated for most codes. The Roadrunner system (Table 1) is built up as a cluster-of-clusters, with 17 separate Connected Units (CUs) linked together by eight 288-port InfiniBand 4x DDR switches at the top level. In turn, each CU consists of 180 hybrid compute nodes and 12 I/O nodes that are connected through a single 288-port InfiniBand 4x DDR switch. The total power required to run the cluster is 2.35 MW, providing a theoretical maximum of 437 Megaflop/s/Watt. Roadrunner is installed in 278 racks weighing 500,000 lbs with 55 miles of IB cables, and has a footprint of 5200 ft^2.

Each hybrid compute ("triblade") node consists of two QS22 IBM Cell Blades, each with two PowerXCell8i processors, and one LS21 dual-socket Opteron blade, with one 1.8 GHz dual-core Opteron chip per socket. The Opteron and Cell blades are connected via four PCI Express X8 links, providing a total effective bandwidth of 6.4 GB/s within each node. An InfiniBand 4x DDR link provides a 2.0 GB/s off-node bandwidth, for a full bi-directional bandwidth of 384 GB/s between the 180 compute nodes and 12 I/O nodes of each CU.

Each Opteron core is capable of issuing one fused multiply-add instruction per cycle, providing a peak Opteron double-precision (DP) floating-point rate of (2 operations/cycle) x (4 cores/node) x 1.8 GHz = 14.4 Gflop/s for each node. On the Cell side, each of the four PowerXCell 8i cell blades has a peak DP floating-point rate or 102.4 Gflop/s, leading to the peak performance rates per node of 424 Gflop/s. A summary of performance numbers for a single node, single CU, and the entire Roadrunner system reported in Table 1. (The I/O nodes are not included in the memory or peak Flop/s counts.) Although other arrangements are possible, the approach that most code developers (including us) have taken is to treat each Opteron core as a separate MPI rank, each communicating in turn with a single Cell processor via the Data, Communication, and Synchronization (DaCS) library.

Characteristic	Hybrid Compute Node	Connected Unit	Roadrunner System
IBM eDP PowerXCell8i Processors	4	720	12240
1.8 GHz Dual-core Opterons	2	360 compute + 24 I/O	6120 compute + 408 I/O
Memory (Opteron)	16 GB	2.88 TB	49 TB
Memory (Cell)	16 GB	2.88 TB	49 TB
Memory (Total)	32 GB	5.76 TB	98 TB
Peak DP Flop/s (Opteron)	14.4 GF	2.592 TF	44.1 TF
Peak DP Flop/s (Cell)	409.6 GF	73.73 TF	1.260 PF
Peak DP Flop/s (Total)	424.0 GF	76.32 TF	1.304 PF

Table 1: Summary of different components that make up roadrunner showing the amount of memory on the Opterons and the Cells and the contribution to the performance from each element of the system.

3. MOLECULAR DYNAMICS WITH SHORT-RANGE POTENTIALS

3.1. Background

Molecular dynamics (MD) simulations [8] integrate the classical equations of motion for a collection of atoms interacting via some prescribed analytic or tabulated interatomic potential energy function. The method of molecular dynamics was developed and applied for the first time in 1957 by Alder and Wainwright for a simple system consisting only of on the order of hundred hard sphere particles, which surprisingly was already sufficient to exhibit a phase transformation from the solid to the liquid state [9]. These first pioneers utilized vacuum tube architectures that had on the order of 1000 Flop/s. The method was later used to investigate the long-time tail of the velocity autocorrelation of interacting particles, which could be explained by a hydrodynamic analog [10], as well as to describe simple stationary non-equilibrium flows that could be compared to hydrodynamic modeling [11,12].

Since the invention of MD simulations a half-century ago, improvements in computational speed, efficient potential functions that are able to describe metallic solids (subsection 3.1), and computational algorithms ([8] and subsection 3.2) have contributed to progress in fundamental material science. This is particularly true for in regimes that are difficult to probe experimentally, including extreme conditions of temperature, strain, and strain rates. MD simulations have been used to provide unprecedented insight into fundamental unit processes controlling material strength and mechanical response, including the interaction of dislocations [13] and their nucleation under shock compression [14]. Atomic-scale mechanisms of the solid-solid phase transformation generated in shock-loaded iron have been identified and directly compared to small scale experiments [15-17], as well as the nucleation of pressure-induced solidification in complex metals [18]. There are also numerous examples of successes in the area of biology, chemistry, and bio-physical systems that have been modeled using semi-empirical potentials, not too dissimilar from the ones described below. Even more broadly, we have adapted our molecular dynamics code SPaSM to describe the spread of diseases using an agent-based simulation model, and used it to assess possible mitigation strategies in support of the U.S. pandemic influenza planning process [19]; here each individual person in the United States is represented by an agent (atom), and the interatomic potential function is replaced by stochastic rules for disease transmission between individuals that are in close proximity.

All the aforementioned examples illustrate that the present implementation of a large-scale MD code based on short-range potentials on hybrid architectures paves the way for potential future applications in a whole variety of arenas. The initial scientific application which we will discuss below (in section 5) involves two physical processes that we have already demonstrated can be successfully studied using MD simulations: shock physics [14-17] and complex hydrodynamic flows such as fluid instabilities [20-22]. The Rayleigh-Taylor instability arises when initially a heavy fluid is placed on top of a light fluid under the presence of gravity, leading to subsequent turbulent mixing [23]. A similar instability, the Richtmyer-Meshkov instability, is triggered by a shock waves passing through an interface between two different fluids, a fluid and a vacuum, or a solid that melts due to the shock wave interfacing a gaseous medium [24]. These two instabilities are very interesting to study from a fundamental point of view, but they are also very important to understand in the context of inertial confinement fusion (ICF), a potential future energy source, in which a millimeter-sized capsule is imploded to initiate nuclear fusion between deuterium and tritium. It has been shown previously that under extreme conditions the atomistic description might be necessary to describe such instabilities; in particular there is increasing evidence that fluctuations that are normally ignored in continuum descriptions may in fact be important [21,25].

With the potential now given by Roadrunner and our high-performance implementation, we believe that we can successfully tackle the Richtmyer-Meshkov challenge – on the atomistic scale for micrometer-sized samples – in connection with ejecta formation at the interface of the two substances [26,27]. This problem is extremely difficult, if not impossible, to describe by regular hydrodynamic descriptions because it involves a discontinuous distribution of material and associated break-ups. Again, the understanding of this problem is one of the building blocks to the understanding and the success of ICF. The understanding that we expect will be gained by our simulations will further improve the engineering-scale modeling efforts in this area.

3.2. Interatomic potentials

For metals and other systems that do not involve species with ionic charges or dipole (or higher) electrical moments, interactions are effectively screened such that each atom only interacts noticeably with other atoms in its immediate neighborhood. The interatomic potential, which is often decomposed into two-body pair potentials, three-body angular potentials, and so on, can be further simplified by introducing a cut-off distance r_c, beyond which atoms do not interact at all. Large-scale simulations in which the system size greatly exceeds r_c can thus be reduced from $O(N^2)$ computational cost to $O(N)$ by partitioning three-dimensional space [3,4,28], particles [18,25], or force calculations [29] among processors.

Probably the most commonly utilized pair potential, and a standard benchmark for MD codes, is the Lennard-Jones 6-12 potential. The potential energy between each pair of atoms (closer together than r_c) is given by

$$\phi(r_{ij}) = 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right], \quad r_{ij} \le r_c .$$

Although highly simplified, it has provided a great deal of insight into fundamental statistical mechanics [8], materials science [14], and fluid dynamics [20] issues. A much more realistic description of metals, with only about twice the computational effort of a pairwise potential, is provided by the embedded atom method (EAM), which expresses the energy for each atom i as

$$E_i = \frac{1}{2}\sum_j \phi(r_{ij}) + F\left[\sum_j \rho_j(r_{ij})\right].$$

Here $\phi(r_{ij})$ nominally describes the repulsion between positively charged nuclei, as in the r^{-12} term of the Lennard-Jones potential, and the attraction is described by the second term, a functional of the pairwise additive electron density contributions from each neighboring atom j, $\rho_j(r_{ij})$. (As before, the summations only run over neighboring atoms j within some cutoff distance r_c.) Although this is the interpretation originally motivated by density functional theory, it has become much more commonplace in recent years to include some degree of attraction in the pairwise term, e.g. by a Morse or Lennard-Jones potential. The density function $\rho_j(r_{ij})$ and/or functional $F(\rho)$ are then fit to reproduce equation-of-state and other material properties, such as elastic constants, zone-boundary phonon frequencies, vacancy energies, either from experiments or *ab initio* calculations. The functions $\phi(r_{ij})$, $\rho_j(r_{ij})$, and $F(\rho)$ are typically stored in tabular form, since their analytic representations (if one even exists) are often complex functions involving square root and exponential operations, and are smoothly splined to zero at the cutoff distance. Both simple metals and their alloys can be well-described by EAM potentials, which have been used to study a wide variety of liquid [22] and solid-state [13, 15-17] phenomena.

3.3. Numerical Precision Requirements

MD simulations traditionally utilize double precision for all numerical values, i.e. 64 bits per floating point number. To our knowledge, there are few detailed investigations of circumstances under which single precision (i.e. 32 bits) might suffice, or likewise whether there are situations where even double precision is insufficient. Typically, there has been little to no performance gain if one switches to single precision (i.e. 32 bits) [30], although SSE and other vector instruction sets offer a potential performance gain of 2x. It is pretty clear that the storage of *all* floating point information in double precision is probably overkill, but on the other hand using exclusively single precision has been shown to increase fluctuations of quantities such as the total energy of the system, and therefore may result in a non-converged result that can be misleading in terms of its physical interpretation [8]. On configurable circuits it has been shown that the optimal storage size required to obtain sufficient convergence lies somewhere in between single and double precision for most applications, a choice that we do not have on the present architecture. Furthermore, it has been shown that for intermediate data, i.e. data that is not used for analyses, the precision can be less than double [31]. Another parameter that plays a significant role in reducing the precision of floating point data is the applied timestep, which is used to discretize the equations of motion [8,30]. It has been shown that one can reduce the timestep and reduce the precision to achieve the same result. However, there is to date no general consensus on the reliability of single precision floating point data storage, and associated parameters such as the applied timestep. For this reason, we have chosen to utilize exclusively double precision in our present implementation, although we expect that we would be able to get close to a 2x performance speedup (i.e. 737 TFlop/s peak performance for our Lennard-Jones benchmark) by reverting to single precision. Further in-depth studies are needed to address whether some or all variables (e.g. forces but not distances) might be converted to single precision. These studies would have to be done for particular applications separately due the interplay of parameters such as the timestep, temperature, shock velocity and the like. If the timestep would have to be cut in half when using single precision, the performance gain per timestep would be completely lost. As the applicability of the code was one of the main motivations we decided to use double precision first – which allows us to use well-known parameters – and will investigate the possibility of single precision for certain problems later. However, it is expected that for problems involving steep gradients such as shock waves and associated break-ups, double precision will remain necessary.

3.4. MIMD Design Considerations, ca. 1993

The introduction of massively parallel supercomputers in the early 1990s led to a reconsideration of many high-performance computer algorithms that were geared towards the earlier vector era. A prime example is the case of molecular dynamics simulation algorithms; on vector machines a "neighbor list" would often be maintained for each atom, enabling a rapid evaluation of the sums over neighboring atoms for potentials such as Lennard-Jones or EAM. Arrays for each Cartesian position (x, y, z) and velocity (v_x, v_y, v_z) could be updated in-place during each integration timestep. However, such an approach is not readily adapted to distributed-memory MIMD architectures. Several groups recognized that a more natural parallelization strategy would rely on domain decomposition of the 2D or 3D simulation

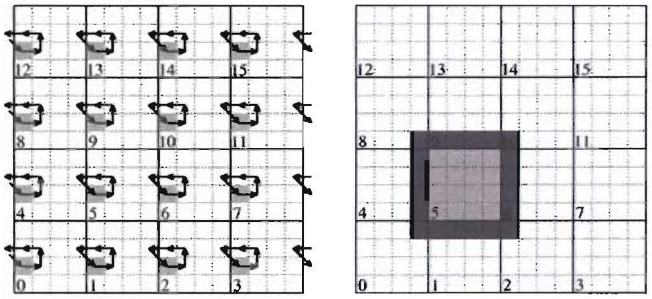

Figure 1: SPaSM algorithm and communication pattern, old and new (examples for 16 CPUs in 2D). (Left) Lock-step computation of force interactions within and between subdomains using an interaction path, with synchronous MPI_Sendrecv() calls when crossing a processor boundary. (Right) Synchronous communication of a ghost subdomain layer (blue) to each CPU at the beginning of each timestep enables each CPU to compute all forces without further communication.

space, taking advantage of the short-range nature of many potentials to limit communication to neighboring processors on a mesh or torus (for periodic boundary conditions) topology.

One of the most successful practical implementations of such a strategy resulted in the SPaSM (Scalable Parallel Short-range Molecular dynamics) code, developed by Peter Lomdahl, David Beazley, and their coworkers at LANL [3,4]. The key insight of the SPaSM code was to continue the domain decomposition between processors, further dividing the space within processors into subdomains. (The original terminology referred to these subdomains as cells; we will avoid this terminology due to possible confusion with the Cell processor.) Provided that each subdomain is at least r_c in length (in each dimension), neighbor lists could be completely eliminated; instead, neighboring atoms could be found on-the-fly during each timestep by examining pairs of atoms within each subdomain (referred to as "self interactions"), and between immediately adjacent subdomains (referred to as "list interactions"). Newton's Third Law can further reduce this search to only half of the neighboring subdomains (4 instead of 8 in 2D, and 13 instead of 26 in 3D) by computing each pairwise force F_{ij} once and assigning $\pm F_{ij}$ to atoms i and j, respectively. By organizing the computation of interactions between the 4 neighboring subdomains in 2D (13 in 3D) by an interaction path, interprocessor communication is readily coordinated by synchronously sending (and receiving) a subdomain whenever its interaction path crosses a processor boundary (Fig 1, left). C data structures are used to simplify manipulation of subdomains (consisting of a pointer to a list of particles, the number of particles, and perhaps some auxiliary data) and particles (a structure with a type, position and velocity vectors, and other quantities depending on the potential). After positions (and velocities) are updated at the end of each timestep, particles crossing subdomain boundaries are readily re-sorted by a single memcpy() of each particle's `struct`, into a buffer to be sent to a neighboring processor if necessary. (Numerical stability of the molecular dynamics integration routine requires that each particle move only a small fraction of r_c during each timestep.)

4. (RE-)IMPLEMENTATION OF THE SPASM CODE ON ROADRUNNER

4.1. An Evolutionary Approach

An analysis of the profile of the existing SPaSM code shows that we spend approximately 95% of the time in the force subroutine. Given this, our first attempt at acceleration on hybrid architectures such as Roadrunner was an 'evolutionary' approach, where we accelerated the force computation on the Cells, leaving particle updates and other steps (comprising over 80% of the code) unchanged, to be carried out by the Opterons. However, even for this port it was immediately apparent that the large number of small messages arising from the interaction path approach would be prohibitively expensive. This is because two steps are required to transfer data from the Opteron to a SPE, or vice versa: a send/receive pair of DaCS library calls to exchange particles between the 4 GB associated with each Opteron core and the 4 GB main memory for each Cell processor, followed by a direct memory access (DMA) instruction to fetch data from the Cell's main memory to the 256 kB local store on each SPE. Latency overhead and data synchronization issues quickly eliminated the interaction path as a viable approach.

Instead, we adopted a "ghost subdomain" approach in which each Opteron core receives a boundary layer of subdomains from neighboring processors in a six-step send-and-receive sequence at the beginning of each timestep. The force computation, and subsequent particle position and velocity updates, can then proceed independently on each Opteron/Cell pair, with no further MPI communication required until particles need to be redistributed and the next timestep is reached. Thus at any given point of time, either the Opterons or the Cells are doing computations while the other sits idle.

Since this was our first foray into the field of Cell programming, we decided that it was better to be correct than fast. For this reason, we decided to forego some optimizations, such as using

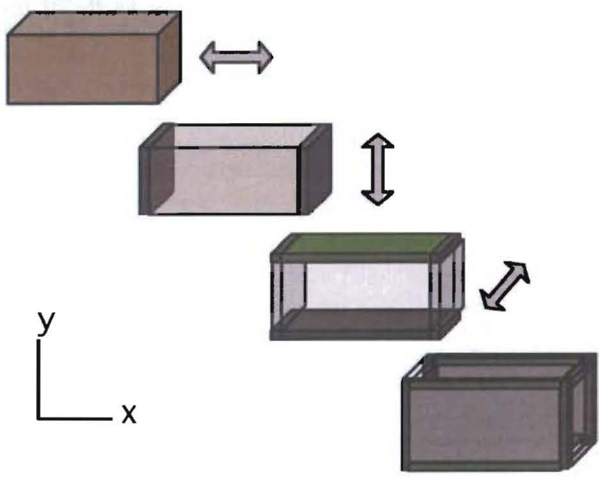

Figure 2: Six-way sequence of MPI_Sendrecv() calls to transmit ghost subdomain. Planes of subdomains in the ±x direction are sent first, followed by planes in the ±y direction (including the extra rows provided by the ±x step). A final pair of calls in the ±z direction completes the communication of all neighboring subdomains, including diagonal ones.

Newton's Third Law (each action has an equal and opposite reaction) for the force computation. This resulted in each interaction being computed twice but reduced the complexity of the Cell code. Additionally, the optimal data layout for the SPUs on the Cell is a Structure of Arrays (SOA), where all the data for a given variable is clustered together in memory, while the SPaSM code uses an Array of Structures (AOS), where all the data for a particle is clustered together. This required us to convert the data from the SPaSM format to one that would be efficient on the Cell. Furthermore, we needed to byte-swap all data, since the Opteron uses little-endian byte ordering, and the Cell big-endian. To achieve this, we initially performed these conversions on the PPU. However, this approach led to a 50% overhead in the simulations. Instead, sharing this responsibility between the Opterons and the SPUs turned out to be the best solution. In our final implementation we convert the AOS into an SOA, and convert the Endianness of the positions on the Opteron. On the return, the SPUs convert the Endianness of the force, while the Opteron again converts the Cell SOA into the SPaSM AOS data layout. With this modification, the overhead for Endian conversion fell below 0.1% of the total time, making it essentially free.

For this approach, we find that the force computation on the Cell takes about 60% of the time while the Opterons compute the remaining 40% of the time. Of this 40% approximately 20% is spent in the data conversion routines. Overall we obtained approximately 2x speedup over SPaSM running on the base Opterons. For the Lennard-Jones potential, we measured a performance of about 8 GF per node, giving a projected performance of 98 TFlop/s on the complete Roadrunner system.

Our goal in all of this, though was to reduce the wall clock time as much as possible, and so we decided to explore whether we could get much better performance by redesigning the entire communication infrastructure and data structures of SPaSM to allow for better asynchrony between the processors. From Table 1 we note that over 96% of the theoretical peak performance of Roadrunner comes from the Cell processors, meaning that they should be kept as busy as possible. Our 'evolutionary' approach of sending particle positions from the Opteron to Cell to compute forces, which are then returned to the Opteron for position and velocity updates (and particle redistribution) was clearly deficient in this regard. We will now describe our 'revolutionary' approach, where almost the entire code base is changed to accommodate the Cell processor, and the unique multi-layer hierarchy of Roadrunner's architecture.

4.2. A Revolutionary Approach

In computing the interactions between particles using SPaSM's subdomain layout, interactions are computed between all particles within a given subdomain, and then with particles in neighboring subdomains. These are termed as 'self' and 'list' interactions respectively. In looking at this it is immediately evident that the 'self' interactions do not depend on any ghost subdomains, since they depend exclusively on the data local to the current processor. Consequently it is possible to overlap the exchange of the ghost subdomains with neighboring Opterons with the 'self' computation on the SPU.

In this approach we have transferred all the computation to the SPUs. The PPU coordinates data movement between the SPUs and the Opterons, while the Opterons handle all the off-node (MPI) communications. Since this was a complicated data flow, we decided to implement this first for the Lennard Jones potential, which does not have to transfer data in the middle of the iteration. This is shown schematically in Figure 3.

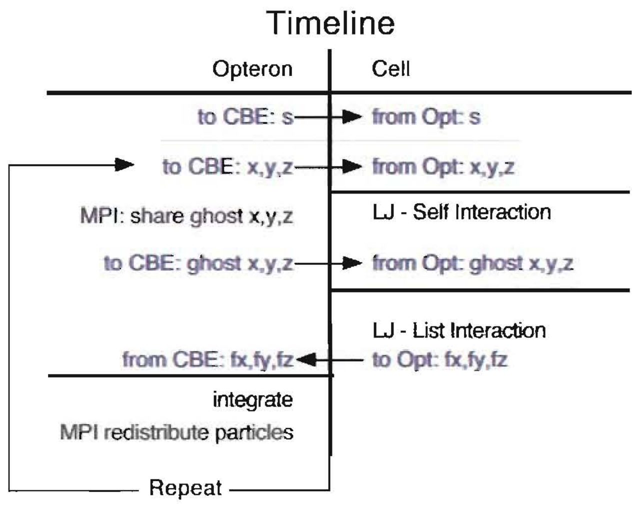

Figure 3: Simplified timeline for the acceleration of the Lennard-Jones timestep within SPaSM on Roadrunner. The self *(within-subdomain) and* list *(between subdomain) interactions are separated, allowing MPI communications on the Opterons to overlap computation on the SPUs. All computation is offloaded to the SPUs while the PPU handles data transfers between the Opterons and the Cell, and the Opterons conduct all MPI traffic. Thick lines in the image are hard synchronizations between the Opteron and the Cell.*

For this implementation, we also redid the interactions so that we use Newton's third law and are now evaluating the same number of interactions as the original Opteron version of SPaSM. For this case we get a speedup of approximately 6x over the base Opterons of Roadrunner.

4.3. Cell-specific optimizations

To achieve the high percentage of peak on the Cells, we had to do several optimizations with regard to the transfer of data to and from the SPUs using DMAs and with regard to the computation itself on the SPUs. The first of these was in the data layout. These days conventional CPUs have very efficient cache systems. So much so that one does not have to worry much about data layout, or the pattern in which data is accessed from Main Memory. On the SPUs however, the programmer is required to design the transfers of data from Main Memory to Local Store. Consequently it is extremely important to plan the data layout in main memory to facilitate efficient access from the SPUs.

The approach we took to solve this problem was to preallocate data for the maximum number of particles we could hold in each subdomain at the beginning of the simulation. Thus, the data for each array (position, forces, electron density, and energy per atom) was subdomain order padded, by which we mean that for each array the amount of space allocated was N_{max}* *ndx* ndy* ndz* doubles where N_{max} is the maximum number of atoms allowed in any subdomain, and *ndx*, *ndy*, and *ndz* are the number of subdomains in *x*, *y*, and *z* respectively. The first N_{max} entries in each array belong to subdomain (0,0,0), the next N_{max} to subdomain (0,0,1), and so on. This layout has two primary advantages: first, the size of all subdomains is the same, and so the addresses can be precalculated at the beginning of the simulation and would remain valid for the entire simulation; and second, when particles move between subdomains it is very fast to pop particles off one subdomains and push them into another. As an added optimization, the addresses of neighboring subdomains were precalculated in the beginning and stored in a dma-list format. This eliminated the need to do pointer math on the SPUs when fetching particle data from disparate parts of memory.

Once the data layout was finalized, the force subroutine was optimized in the following major steps: SIMD vectorization, double buffered DMAs, a hand-optimized reciprocal subroutine, manual loop unrolling, and function parameter culling. Using the original sequential code on the Opterons as a baseline, the speedups we got (in wallclock time) on each successive step for the revolutionary approach are shown in Table 2. Of these five steps, the two that provided the most return for the effort involved were the SIMD vectorization and the manual loop unrolling. Compiler choice makes a difference as well: in general, we found that *gcc* generated executables ran approximately 25% slower for our code as compared to *xlc*.

Strategy	factor	total
Serial on SPUs	0.5	0.5
SIMD Vectorization	2.0	1.0
double buffered DMAs	1.1	1.1
hand-coded reciprocal	1.25	1.38
manual loop unrolling	4.0	5.52
function parameter culling	1.12	6.18

Table 2: Relative contribution of different optimization strategies for the revolutionary approach. The base line for these optimizations is the original code running on a single Opteron in serial mode, which runs twice as slow when ported to the SPUs prior to any optimizations. The speedup given by each strategy is given, as well as the cumulative acceleration in the rightmost column. The main contribution to the speedup was from SIMD vectorization and the manual unrolling of the innermost loop. It should be noted that without the SIMD vectorization, the unrolling sped up the code by less than 25%.

4.4. Performance Benchmarks

To compute elapsed times and diagnose performance without excess overhead, we implemented a lightweight *gettimeofday*() based timing method on the Opteron-level processes, with less than 0.01% overhead. This method is event-based, with each call to a timer recording the current time, an integer flag identifying the timer, and an additional user-specified marker value. The output is flushed to file whenever a buffer (typically set to 1 million events) is full. The file can be post-processed after the run to determine the time spent in the different routines or in subsections of different routines. The implementation allows one to turn timers on and off at will, either one at a time, or all of them. This allows us to sample the performance during a given run either statistically or continuously, and also allows us to turn on the timers on different processors at different times. For the timings obtained here, we enabled timers on MPI rank 0, and left them disabled on all other processors.

For the Lennard-Jones potential, for each atom *i* in a given subdomain, we compute the distance squared, r_{ij}^2, to all other atoms *j* either in the same subdomain (self interaction), or in neighboring subdomains (list interactions). Since branching on the SPU is expensive we computed all interactions, but only kept those that have a distance less than the potential cutoff distance, r_c. Thus, for each pair of atoms we must compute a distance squared (*3 sub*, *1 mul*, and *2 madd* instructions) to obtain $r_{ij}^2 = (x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2$). This is then inverted (*1 reciprocal)* and multiplied (*1 mul)* by either 1.0 or 0.0 to set $1/r_{ij}^2$ to 0.0 for $r_{ij}^2 > r_c^2$. The result it then cubed (*2 muls)* to get $1/r_{ij}^6$, which is subsequently squared (*1 mul)* to get $1/r_{ij}^{12}$. We then obtain the potential energy $\phi(r_{ij})$ with *2 madd* and *2 add* instructions, with an extra addition (*1 add*) to shift the potential by the energy at the cutoff for $r_{ij}^2 < r_c^2$ so that the potential energy vanishes continuously at r_c. Similarly, the force $-d\phi(r_{ij})/dr_{ij}$ is computed by a total of *5 muls* and *6 madds*, for a total of 37 floating-point instructions (flops). However, we found that recoding the reciprocal as a Newton-Raphson loop starting with a single precision floating point reciprocal estimate (*frest*) reduced the wall clock time by over 10%. This hand-coded routine has 9 floating point operations (*1 frest* (not counted), *3 muls, 3 madds)*, bringing the total number of floating point operations to 45 for each pairwise Lennard-Jones interaction (both energy and force). By counting the number N_{pair} of such interactions computed during a series of timesteps (with a cumulative wall clock time t_{wall}, we compute the performance as 45 N_{pair} / t_{wall}.

To compare our performance against earlier results, we utilize a standard Lennard-Jones benchmark problem [3-7] in which the sample is set up in a face-centered-cubic (fcc) crystal lattice, at a density (0.8442 atoms / σ^3) corresponding to the triple point (where gas, liquid, and solid phases coexist), but at an elevated temperature (0.9 instead of 0.687 in reduced Lennard-Jones units [8]). A short timing simulation is run, typically 20-200 timesteps, counting the overall number of pairwise interactions computed and the wall-clock time. To measure weak scaling, we set up a series of such runs with one million fcc unit cells (4 million atoms) per MPI rank, in a 100x100x100 cube of unit cells. They thus range from 2.88 billion atoms on 1 CU (720 MPI ranks) to 48.96 billion atoms on 17 CUs (12240 MPI ranks). We have found the weak scaling of this case to be linear on Roadrunner. This is shown in Figure 4 as a function of number of connected units. The symbols in Figure 4 are the measured values while the solid black line is the best linear fit to the data. This linear scaling is similar to our experience on a number of large machines, most notably on BlueGene/L up to 212,992 processors [6,7]. For the full 17 CU benchmark run, we measured 83.1 x 10^{12} interaction computations with 45 floating-point operations, giving us a total of 3.74 x 10^{15} floating-point operations for each timestep. The wall-clock time for each such step was 10.14s, giving us an aggregate performance of **368.6 TFlop/s**.

We estimate that the main interaction kernel of the code is running at 45% of peak for this run. This high percentage of peak is achieved because we have implemented 49% of the kernel as fused multiply-add instructions, which push through four flops in each cycle on each of the eight SPUs. However, the overall force subroutine on the Cell runs at 34 GFlop/s due to overheads associated with data transfer and data movement. On the Opterons, the MPI communications add to the overhead, and the rate per Opteron core drops to 30.1 GFlop/s.

Strong scaling results, in which the problem size is held fixed and the number of processors varied, is shown in Figure 5. We note that communication overhead diminishes performance for problem sizes smaller than ~1 billion atoms per CU, or ~1 million atoms per MPI rank. The "sweet spot" thus appears to be between 1 and 4 billion atoms per CU, where we measure consistent floating-point rates over 22 TFlop/s per CU, and iteration times that are quite reasonable for production runs, a few seconds per timestep. (Although the total amount of memory available on Roadrunner is comparable to the IBM/LLNL BlueGene/L platform, where we have demonstrated scalable performance up to 320 billion atoms [7], we have gladly sacrificed memory for improved performance in our Roadrunner redesign, since in practice MD simulations are always bound by computation time, not memory.)

Figure 4: Weak scaling of SPaSM on Roadrunner for the benchmark Lennard-Jones (LJ) problem, measured over 20 timesteps using the simplified revolutionary LJ approach. The symbols are measured data while the solid line is the best linear fit through all the data.

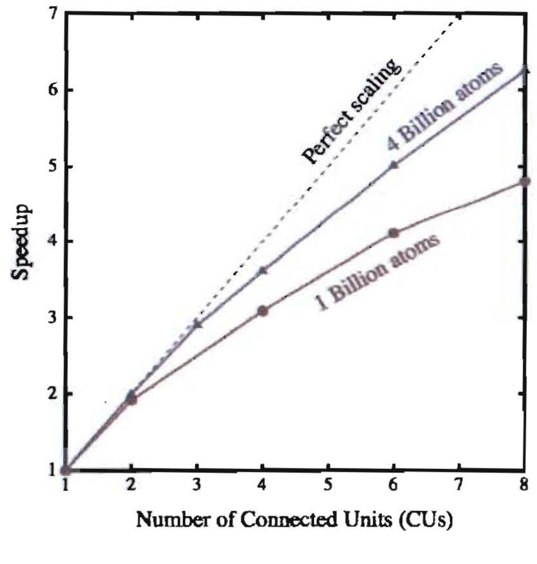

Figure 5: Strong scaling of SPaSM on Roadrunner for the benchmark Lennard-Jones (LJ) problem, measured as in Fig. 4. Two problem sizes are shown, with the speedup relative to the run time on 1 CU in both cases.

A breakdown of the time spent within each timestep in different parts of the code is shown in Figure 6. A quick analysis shows us that 85% of the time is spent in the force subroutine, 9% in the particle redistribution routines, 2% in the position update, and 1% in the velocity update. Of these subroutines the force routine exists primarily on the SPUs, whereas the rest of the routines reside on the Opteron. We believe it is possible to improve the performance of this code by implementing the marking of particles for redistribution, and position and velocity integration steps, on the SPUs. Since parallel communication occurs in the redistribution step, it is not possible to recover the time spent in this function fully. However, we believe that at least half the time spent in this function is consumed by the marking of particles; this work could be transferred to the SPUs, giving us at least a 10% boost in performance, and pushing us over 400 TFlop/s.

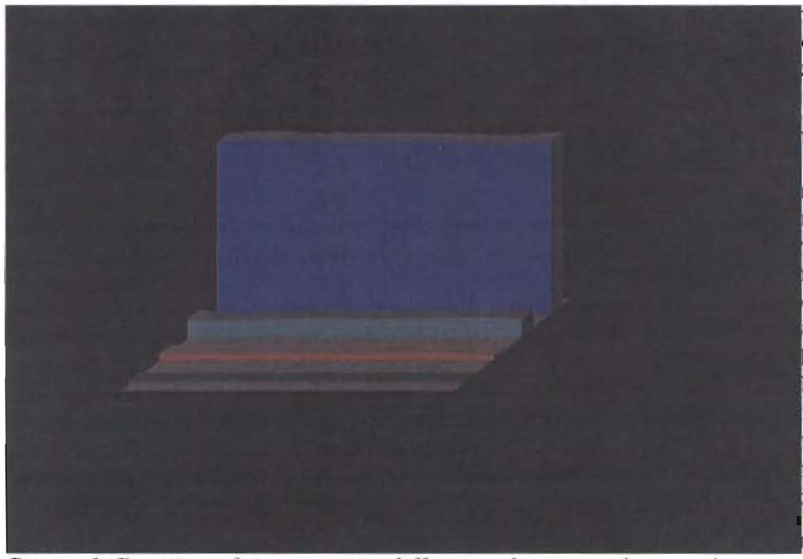
Figure 6: Fraction of time spent in different subroutines for the weak scaling results of Fig. 4 during a single timestep, for different numbers of connected units (CUs). The force subroutine exists entirely on the SPUs, while the rest of the subroutines run on the Opterons.

4.5. Additional Benefits

Since the Cells "own" the particle data in the "revolutionary" redesign just described, while the Opterons primarily conduct traffic, it is possible to periodically copy the current state (particle positions and velocities) to the Opterons and spawn a thread to perform more time-consuming analysis, visualization, and checkpointing of that snapshot in the background while timestep integration continues normally. Further, the use of ghost subdomains eliminates the requirement for a uniform rectilinear spatial decomposition. This can be used to perform a rudimentary load balancing by adjusting the (generally non-uniform) rectilinear decomposition whenever we encounter large variations in density across the simulation domain. Many of the problems which we are interested in, including the Rayleigh-Taylor fluid instability [20,21] and planar shock compression of materials [14-17,27], involve large density gradients in one spatial dimension, so we are currently implementing a simple one-dimensional dynamic load balancing procedure that we expect to utilize by the time Roadrunner delivery to LANL is completed later this fall.

5. MD SIMULATIONS OF SHOCK EJECTION, JET FORMATION, AND BREAKUP

The ejection of material from shocked surfaces is a problem that has attracted increased attention both experimentally [17] and theoretically [18] at LANL. Models are required that can predict the amount of mass ejected from a shocked interface with a given surface finish and loading history (peak shock pressure, either from a supported square-wave or HE-driven Taylor wave), and also the particle size and velocity distributions and their evolutions (and correlations between the two). The total mass can be inferred by measuring the resulting momentum transfer onto an Asay foil or piezoelectric probe at some standoff distance, while particle sizes larger than a micron can be measured using
hy or X-ray radiography. However, there is no
on on the distribution of particle sizes below a micron,
correlation between size and velocity distributions.

Lennard-Jones potentials are extremely useful for
nany basic physical processes, in order to more directly
our ejecta simulation project with a corresponding
ntal effort at LANL we are using EAM potential to
realistic metals, such as copper. We have implemented
olutionary" approach for the EAM potential on
er, shown schematically in Figure 7. The algorithm is a
rward extension of the Lennard-Jones one shown above
vith an added communication step (to exchange F'(ρ)) in
le of the computation, that can also be overlapped in a
ashion. Although loop unrolling and several other
ions such as those in Table 2 have not been completely
ited and optimized, our Cell-accelerated EAM
itation already achieves a speedup of ~4x over the
nly version.

nstrate this code for a problem of shock ejecta production in copper, we carried out a series of runs using 1 CU (720 MPI ranks), for different shock velocities. The sample is 500 x 500 x 2 nm (36 million atoms), with a thin third dimension just thick enough to give realistic (3-dimensional) equation-of-state and material response. The free surface opposite to the shock impact plane contains three machining grooves, each with the same volume but different shapes. Previous theories of shock ejecta have been based on the assumption that defect volume is the dominant variable, but our simulations clearly show significant differences in the initial jetting from differently shaped defects (Figure 8). These simulations are providing useful insight into the initial formation of shock ejecta, leading to a model based on the Richtmyer-Meshkov instability that appears to successfully describe the bubble and spike dynamics, which determine the total ejecta mass and peak velocity of the leading ejecta.

With the computational capabilities of Roadrunner and our optimized SPaSM code, we will also be able to probe the next step, the subsequent breakup of the jets (either liquid or solid, depending on shock pressure) into fragments. This will enable a direct (computational) measurement of the size and velocity distributions of the sub-micron particulates that have thus far eluded experimental measurement, complementing recent experimental advances pushing down into the sub-micron length scale. The fundamental phenomena of liquid jet breakup and atomization have impacts in areas well beyond weapons science, ranging from medical and agricultural applications to ink-jet printing and jet engines. As two recent reviews have highlighted [32,33], several competing theories exist even though very fundamental questions are still unanswered, such as whether the particle size distribution vanishes at some lower limit, or continues smoothly to the atomistic scale. A few molecular dynamics simulations have been performed to investigate the basic hydrodynamics of the necking, or capillary, instability [25], but thus far computational requirements have prevented the collection of sufficient statistics to answer the size distribution question.

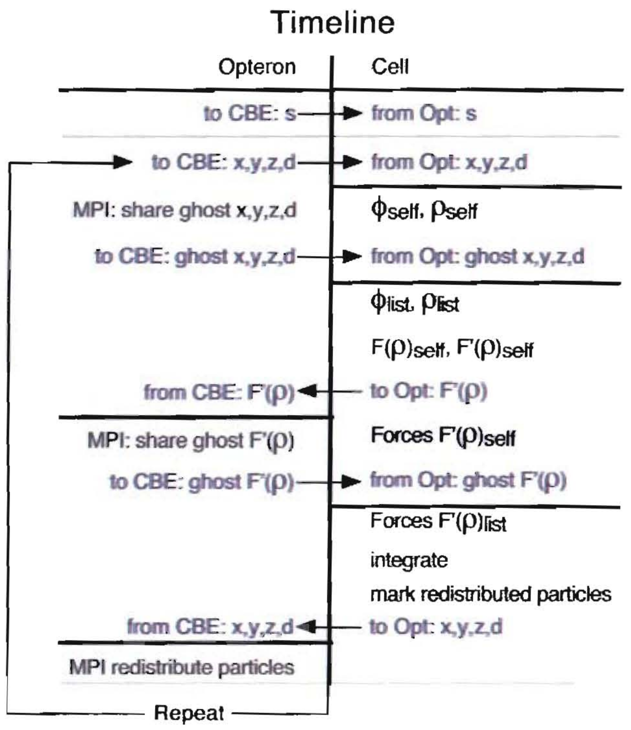

Figure 7: Modified timeline for the Revolutionary Approach for accelerating SPaSM on Roadrunner for the EAM potential. The self *(within a subdomain) and* list *(between subdomain) interactions are separated allowing MPI communications on the Opterons to overlap computation on the SPUs. All computation is offloaded to the SPUs while the PPU handles data transfers between the Opterons and the blade, and the Opterons conduct all MPI traffic. Thick lines in the image are hard synchronizations between the Opteron and the Cell.*

Roadrunner will enable such studies for two reasons: (1) the sheer increase in performance (our 369 TFlop/s represents a ~4x increase in SPaSM performance over the full 212,992-CPU BG/L machine), but more importantly (2) the radical redesign of the message-passing structure within SPaSM that was required to optimize performance on Roadrunner now enables us to load-balance problems with significant density variations, as for a jet expanding into vacuum.

The original (CM-5) implementation of SPaSM would send many small messages between CPUs, as each process marched in lockstep through its grid of subdomains. Consequently, the overall computational cost was determined by the simulation volume, rather then the number of atoms. (Most simulations were chosen to maintain a relatively homogeneous density distribution, so that there was little distinction between the two.) However, for Roadrunner the latency overhead associated with sending large numbers of small messages was prohibitive, so we have removed this lockstep send-and-receive pattern and instead communicate entire planes of ghost subdomains from one processor to another. As such, there is no longer any fundamental restriction that each processor simulate an equal volume (number of subdomains), and we can vary the volume assigned to each processor to balance the computational costs (number of atoms). As mentioned in subsection 4.5, this one-dimensional dynamic load balancing approach is currently being implemented in preparation for the final delivery of Roadrunner later this year.

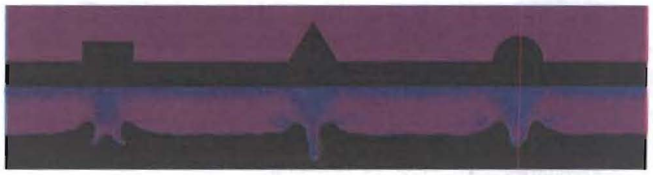

Figure 8: Demonstration run showing the ejection of material from a shocked copper surface, run on 1 CU. The sample is three-dimensional, but with a thin cross-section into the plane; only a small section of the total sample height is shown, at two different times during the simulation. (Top) Free surface prior to shock arrival. (Bottom) Initial jet formation after a shock wave has reflected from the free surface. Three different surface defects are initially present, with the same volume but different shapes, leading to different jetting patterns that will subsequently break up into ejecta particles with different size and velocity distributions.

6. ACKNOWLEDGMENTS

We gratefully acknowledge the advice and support generously provided by members of the Roadrunner Advanced Algorithms Project Team at LANL, and Cornell Wright, Christopher Engel, and Bill Brandmeyer of IBM. This work was supported by the Advanced Simulation and Computing (ASC) Program. LANL is operated by Los Alamos National Security, LLC (LANS) under Contract DE-AC52-06NA25396 with the U.S. Department of Energy (DOE).

7. REFERENCES

[1] Intel® Corporation, Teraflops Research Chip, http://techresearch.intel.com/articles/Tera-Scale/1449.htm

[2] IBM, Cell Broadband Engine resource center, http://www-128.ibm.com/developerworks/power/cell/

[3] P. S. Lomdahl , P. Tamayo , N. Grønbech-Jensen, D. M. Beazley, "50 GFlops molecular dynamics on the Connection Machine 5," Proceedings of the 1993 ACM/IEEE conference on Supercomputing, December 1993, Portland, OR, 520-527. DOI= http://doi.acm.org/10.1145/169627.169794

[4] D. M. Beazley and P. S. Lomdahl, "Message-passing multi-cell molecular dynamics on the Connection Machine 5," Par. Comput., 20: 173-195, 1994.

[5] M. S. Warren, T. C. Germann , P. S. Lomdahl , D. M. Beazley, and J. K. Salmon, "Avalon: an Alpha/Linux cluster achieves 10 Gflops for $150k," Proceedings of the 1998 ACM/IEEE conference on Supercomputing, November 1998, Orlando, FL.

[6] T. C. Germann, K. Kadau, and P. S. Lomdahl, "25 Tflop/s Multibillion-atom Molecular Dynamics Simulations and Visualization/Analysis on BlueGene/L," Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, November 2005, Seattle, WA; http://sc05.supercomputing.org/schedule/pdf/pap122.pdf

[7] K. Kadau, T. C. Germann, and P. S. Lomdahl, "Molecular dynamics comes of age: 320 billion atom simulation on BlueGene/L," Int. J. Mod. Phys. C, 17:1755, 2006.

[8] M. P. Allen and D. J. Tildesley. Computer Simulation of Liquids. Clarendon Press, Oxford, 1987.

[9] B. J. Alder and T.E. Wainwright, "Phase Transition for a Hard Sphere System", J. Chem. Phys. 27: 1208-1209, 1957.

[10] B.J. Alder and T.E. Wainwright, "Decay of Velocity Autocorrelation Function", Phys. Rev. A 1: 18-21, 1970.

[11] W. Hoover and W. Ashurst, "Nonequilibrium Molecular Dynamics", Advances in Theoretical Chemistry, 1-51, 1975.

[12] A. Puhl, M.M. Mansour, M. Mareshal, "Quantitative comparison of molecular dynamics with hydrodynamics in Rayleigh-Benard convection", Phys. Rev. A 40: 1999-2012, 1989.

[13] S. J. Zhou, D. L. Preston, P. S. Lomdahl, and D. M. Beazley, "Large-scale molecular dynamics simulations of dislocation intersection in copper," Science, 279: 1525-1527, 1998.

[14] B. L. Holian and P. S. Lomdahl, "Plasticity Induced by Shock Waves in Nonequilibrium Molecular-Dynamics Simulations," Science, 280: 2085-2088, 1998.

[15] K. Kadau, T. C. Germann, P. S. Lomdahl, and B. L. Holian, "Microscopic View of Structural Phase Transitions Induced by Shock Waves," Science, 296: 1681-1684. 2002.

[16] D.H. Kalantar, J.F. Belak, G. W. Collins, J. D. Colvin, H.M. Davies, J.H. Eggert, T.C. Germann, J. Hawreliak, B.L. Holian, K. Kadau, P.S. Lomdahl, H.E. Lorenzana, M.A. Meyers, K. Rosolankova, M.S. Schneider, J. Sheppard, J.S. Stölken, J.S. Wark, "Direct Observation of the $\alpha-\varepsilon$ Transition in Shock-Compressed Iron via Nanosecond X-ray Diffraction", Phys. Rev. Lett. 95: 075502-1-4, 2005.

[17] K. Kadau, T.C. Germann, P.S. Lomdahl, R.C. Albers, J.S. Wark, A. Higginbotham, B.L. Holian, "Shock Waves in Polycrystalline Iron", Phys. Rev. Lett. 98: 135701-1-4, 2007.

[18] F. H. Streitz, J. N. Glosli, M. V. Patel, B. Chan, R. K. Yates, and B. R. de Supinski, "100+ TFlop Solidification Simulations on Blue Gene/L," Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, November 2005, Seattle, WA; http://sc05.supercomputing.org/schedule/pdf/pap307.pdf

[19] T.C. Germann, K. Kadau, I.M. Longini Jr., C.M. Macken, "Mitigation Strategies for Pandemic Influenza in the United States", Proc. Natl. Acad. Sci. (USA) 103: 5935-5940, 2006.

[20] K. Kadau, T. C. Germann, N. G. Hadjiconstantinou, P. S. Lomdahl, G. Dimonte, B. L. Holian, and B. J. Alder, "Nanohydrodynamics simulations: An atomistic view of the Rayleigh-Taylor instability," Proc. Natl. Acad. Sci. (USA) 101: 5851-5855, 2004.

[21] K. Kadau, C. Rosenblatt, J.L. Barber, T.C. Germann, Z. Huang, P. Carlès, B.L. Holian, B.J. Alder, "The Importance of Fluctuatios in Fluid Mixing", Proc. Natl. Acad. Sci. (USA) **104**: 7741-7746, 2007.

[22] J. N. Glosli, K. J. Caspersen, J. A. Gunnels, D. F. Richards, R. E. Rudd, and F. H. Streitz, "Extending Stability Beyond CPU Millennium: A Micron-Scale Atomistic Simulation of Kelvin–Helmholtz Instability," Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, November 2007, Reno, NV; http://sc07.supercomputing.org/schedule/pdf/gb109.pdf.

[23] Lord Rayleigh, Proc. London Math. Soc. 14: 170-177 (1883); G.I. Taylor, Proc. R. Soc. London Ser. A 201: 192-196 (1950).

[24] R. Richtmyer, Sov. Fluid Dyn. 4: 151-157, 1969.

[25] M. Moseler and U. Landman, "Formation, Stability, and Breakup of Nanojets," Science, 289: 1165-1169, 2000.

[26] M. B. Zellner et al., "Effects of shock-breakout pressure on ejection of micron-scale material from shocked tin surfaces," J. Appl. Phys., 102: 013522, 2007.

[27] T. C. Germann, J. E. Hammerberg, and B. L. Holian, "Large-Scale Molecular Dynamics Simulations of Ejecta Formation in Copper," in Shock Compression of Condensed Matter - 2003: Proceedings of the Conference of the American Physical Society Topical Group on Shock Compression of Condensed Matter. AIP Conference Proceedings, Volume 706, pp. 285-288, 2004.

[28] T. Narumi, Y. Ohno, N. Okimoto, T. Koishi, A. Suenaga, N. Futatsugi, R. Yanai, R.Himeno, S. Fujikawa, M. Taiji, and M. Ikei, "A 55 TFLOPS Simulation of Amyloid-forming Peptides from Yeast Prion Sup35 with the Special-purpose Computer System MDGRAPE-3," Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, November 2006, Tampa, FL; http://sc06.supercomputing.org/schedule/pdf/gb106.pdf.

[29] K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregerson, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, J. K. Salmon, Y. Shan, and D. E. Shaw, "Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters," Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, November 2006, Tampa, FL; http://sc06.supercomputing.org/schedule/pdf/pap259.pdf.

[30] Y. Gu, T. VanCourt, M. C. Herbordt, "Accelerating Molecular Dynamics Simulations with Configurable Circuits", *IEEE 0-7803-9362-7/05*, 475-480, 2005.

[31] T. Amisaki, T. Fujiwara, A. Kusumi, H. Miyagawa, K. Kitamura, "Error evaluation in the design of a special purpose processor that calculates nonbonded forces in molecular dynamics simulations", J. Comp. Chem. 16: 1120-1130, 1995.

[32] E. Villermaux, "Fragmentation," Annu. Rev. Fluid. Mech., 39:419-46, 2007.

[33] J. Eggers and E. Villermaux, "Physics of liquid jets," Rep. Prog. Phys. 71: 036601, 2008.