Anton 2: Raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer

David E. Shaw,[a] J.P. Grossman, Joseph A. Bank, Brannon Batson, J. Adam Butts, Jack C. Chao,[b]
Martin M. Deneroff,[b] Ron O. Dror,[b] Amos Even, Christopher H. Fenton, Anthony Forte, Joseph Gagliardo,
Gennette Gill, Brian Greskamp, C. Richard Ho,[b] Douglas J. Ierardi, Lev Iserovich, Jeffrey S. Kuskin,
Richard H. Larson,[b] Timothy Layman,[b] Li-Siang Lee,[b] Adam K. Lerer, Chester Li,[b] Daniel Killebrew,[b]
Kenneth M. Mackenzie, Shark Yeuk-Hai Mok,[b] Mark A. Moraes, Rolf Mueller,[b] Lawrence J. Nociolo,
Jon L. Peticolas, Terry Quan, Daniel Ramot,[b] John K. Salmon, Daniele P. Scarpazza, U. Ben Schafer,[b]
Naseer Siddique, Christopher W. Snyder,[b] Jochen Spengler, Ping Tak Peter Tang,[b] Michael Theobald, Horia Toma,[b]
Brian Towles, Benjamin Vitale,[b] Stanley C. Wang, and Cliff Young[b]

D. E. Shaw Research, New York, NY 10036, USA

***Abstract*—Anton 2 is a second-generation special-purpose supercomputer for molecular dynamics simulations that achieves significant gains in performance, programmability, and capacity compared to its predecessor, Anton 1. The architecture of Anton 2 is tailored for fine-grained event-driven operation, which improves performance by increasing the overlap of computation with communication, and also allows a wider range of algorithms to run efficiently, enabling many new software-based optimizations. A 512-node Anton 2 machine, currently in operation, is up to ten times faster than Anton 1 with the same number of nodes, greatly expanding the reach of all-atom biomolecular simulations. Anton 2 is the first platform to achieve simulation rates of multiple microseconds of physical time per day for systems with millions of atoms. Demonstrating strong scaling, the machine simulates a standard 23,558-atom benchmark system at a rate of 85 µs/day—180 times faster than any commodity hardware platform or general-purpose supercomputer.**

I. INTRODUCTION

Molecular dynamics (MD) simulations offer exceptional visibility into the behavior of biological macromolecules at an atomic level of detail [15]. The computational demands of these simulations have historically restricted their length, but advances in parallel algorithms [3, 6, 9, 18, 22, 26, 28, 36, 44, 48] and special-purpose hardware [34, 45, 51] have in recent years extended the reach of such simulations to much longer timescales. Studies enabled by this new capability—including several based on millisecond-scale all-atom MD simulations [32, 37, 38, 46, 47]—have led to a number of scientific discoveries (e.g., [4, 8, 16, 23, 30, 35, 49, 55]) involving biological phenomena that were previously inaccessible to both computational and experimental investigation.

Chemical systems of interest for biomolecular simulation (e.g., proteins in water or membranes) range from thousands to millions of atoms in size, with timescales for important biological events ranging from microseconds to seconds of physical time. For these systems, the best current software implementations of MD on commodity hardware or general-purpose supercomputers [6, 9, 21, 22, 36, 39] are limited to simulation rates of hundreds of nanoseconds of physical time per day of wall-clock time, even on the most powerful conventional supercomputers. MD simulations can be accelerated by over an order of magnitude with special-purpose hardware: the first-generation Anton machines [45] (referred to here as Anton 1) are capable of simulating multiple microseconds per day for chemical systems in the 10,000- to 500,000-atom range [4, 41]. Further performance improvements will expand the utility of MD simulations by significantly reducing the time required for simulations in the hundred-microsecond range, by enabling millisecond-scale simulations for larger, more complex chemical systems, and by bringing even longer timescales within practical reach.

We have designed and built Anton 2, a special-purpose supercomputer for MD simulation that outperforms Anton 1 by up to an order of magnitude and outperforms currently available general-purpose hardware by two orders of magnitude, while simultaneously presenting a simpler, more flexible programming model than Anton 1. Like its predecessor, Anton 2 performs the entire MD computation within custom ASICs that are tightly interconnected by a specialized high-performance network. Each ASIC devotes a quarter of its die area to specialized hardware pipelines for calculating interactions between pairs of atoms, and also contains 66 general-purpose programmable processor cores that deliver data to these pipelines and perform the remaining computations required by the MD simulation.

Improvements in VLSI chip fabrication technology provide Anton 2 with more computational units than Anton 1, but due to the overheads of distributing work to and coordinating a larger number of units, this alone is insufficient to deliver proportionally better performance. A key component of the Anton 2 design is thus a set of new mechanisms devoted to efficient fine-grained operation. The resulting architecture more aggressively exploits the parallelism of MD simulations, which fundamentally consist of a large number of fine-grained computations involving individual atoms or small groups of atoms. By providing direct hardware support for fine-grained communication and synchronization [20, 52], Anton 2 allows these computations to be distributed across an increased number of functional units while maintaining high utilization of the underlying hardware resources.

[a] David E. Shaw is also with the Department of Biochemistry and Molecular Biophysics, Columbia University, New York, NY 10032. E-mail correspondence: David.Shaw@DEShawResearch.com.
[b] For current affiliations, see Acknowledgements section.

978-1-4799-5500-8/14 $31.00 © 2014 IEEE
DOI 10.1109/SC.2014.9

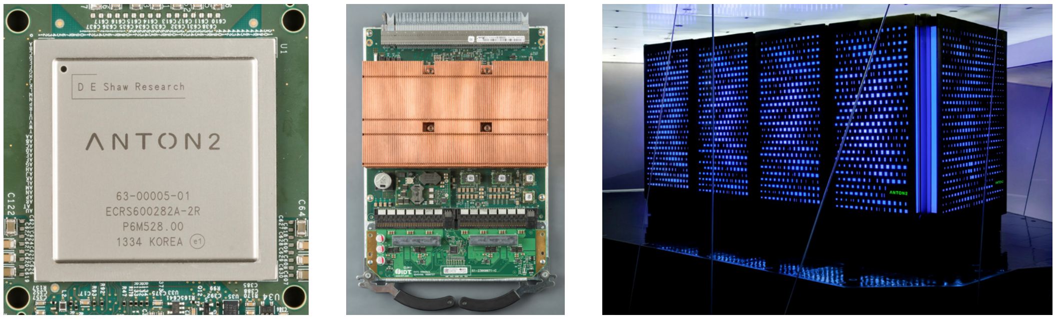

(a) (b) (c)

Fig. 1 (a) An Anton 2 ASIC. (b) An Anton 2 node board, with one ASIC underneath a phase-change heat sink. (c) A 512-node Anton 2 machine, with four racks.

Fine-grained operation is exposed to software via distributed shared memory and an event-driven programming model, with hardware support for scheduling and dispatching small computational tasks. Our MD software for Anton 2 leverages these general-purpose mechanisms with new algorithms, consisting of many sequentially dependent tasks, that would have been impractical on Anton 1, but provide additional performance improvements on Anton 2.

In conjunction with support for fine-grained operation, Anton 2 contains architectural improvements that are more specific to accelerating MD simulations. Both Anton 1 and MDGRAPE [34, 50, 51] use specialized hardware pipelines to significantly speed up the computation of interactions between pairs of atoms. Anton 2 takes the same approach and improves the utilization of these pipelines by eliminating many wasted cycles resulting from non-uniformities in the computation. In addition, Anton 2 uses a new *topological ID* mechanism to avoid the computation of certain undesired interactions.

A 512-node Anton 2 is currently in operation (Fig. 1), and offers three major advances for MD simulation over any other modern supercomputer (including Anton 1):

1. **Peak Performance**. On a 512-node machine, Anton 2 achieves a simulation rate of 85 µs/day for dihydrofolate reductase (DHFR), a benchmark system containing 23,558 atoms that is near the limits of practical parallelism because it has less than one atom per processor core (a 512-node Anton 2 machine contains 33,792 processor cores). This rate represents a 4.5-fold improvement over Anton 1, and is 180 times faster than any implementation of MD on another platform.
2. **Throughput**. On larger chemical systems, Anton 2 demonstrates performance gains of about ten times over Anton 1 (with the same number of nodes), which provides either an increased simulation rate for a fixed chemical system size, or the same simulation rate for a chemical system with over ten times as many atoms.
3. **Capacity at 1+ µs/day.** Anton 2 breaks the microsecond-per-day barrier on million-atom systems, allowing larger biomolecules such as ribosomes to be simulated for much longer timescales. A 2.2-million-atom ribosome simulation runs at 3.6 µs/day on 512 nodes, 21 times faster than a simulation of a similar ribosome system on 1,024 nodes of the SuperMUC cluster [28]. Additional performance could be obtained on larger Anton 2 machines: while the largest machine constructed to date contains 512 nodes, the architecture scales up to 4,096 nodes.

II. Background: Molecular Dynamics Simulation

An MD simulation models the motion of a set of atoms over a large number of discrete time steps. During each time step, the forces acting on all atoms are computed: these forces consist of "bond term" forces between small groups of atoms usually separated by 1–3 covalent bonds, and "non-bonded" forces between all remaining pairs of atoms. The forces are used to "integrate" the atom positions and velocities, which are updated according to Newton's laws of motion. The atoms are typically enclosed in a box with periodic boundary conditions, effectively modelling an infinite number of copies of the chemical system in all three dimensions.

In most implementations (including both Anton 1 and Anton 2), the non-bonded forces are expressed as a sum of "range-limited" forces, which decay rapidly with distance and are explicitly computed between pairs of atoms up to a cutoff radius, and "long-range" forces, which decay more slowly. The long-range forces are computed efficiently using a range-limited pairwise interaction of the atoms with a regular lattice of grid points, followed by an on-grid convolution, followed by a second range-limited pairwise interaction of the atoms with the grid points. Fig. 2 shows the dataflow of an MD simulation, as well as the division of labor on Anton 2 between special-purpose datapaths and general-purpose processors.

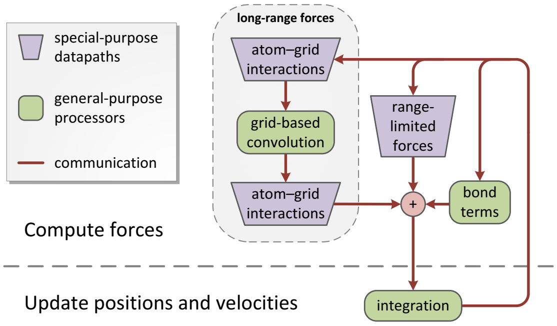


Fig. 2. Dataflow of an MD simulation. On both Anton 1 and Anton 2, range-limited forces and atom–grid interactions are computed by special-purpose datapaths for pairwise interactions, while grid-based convolution, bond terms and integration are computed by general-purpose processors.

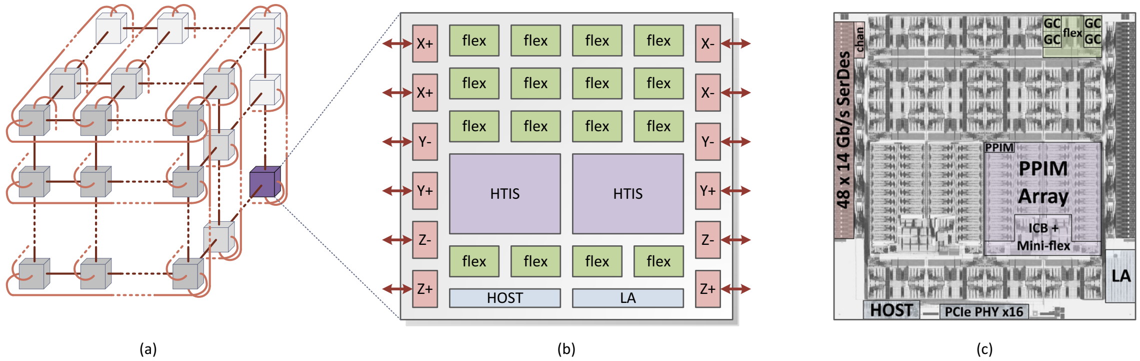


Fig. 3. (a) The Anton 2 ASICs are directly connected by high-speed channels to form a three-dimensional torus topology. (b) Schematic view of an Anton 2 ASIC, which contains 2 connections to each neighbor in the torus topology, 16 flexible subsystem ("flex") tiles, 2 high-throughput interaction subsystem (HTIS) tiles, a host interface (HOST), and an on-die logic analyzer (LA). (c) Physical layout of a 20.4 mm × 20 mm Anton 2 ASIC implemented in 40-nm technology. One of the 12 channel blocks is highlighted as well as one of the 16 flex tiles (with all 4 of the tile's geometry cores shown) and one of the 2 HTIS tiles (with one of the tile's 38 pairwise point interaction modules shown).

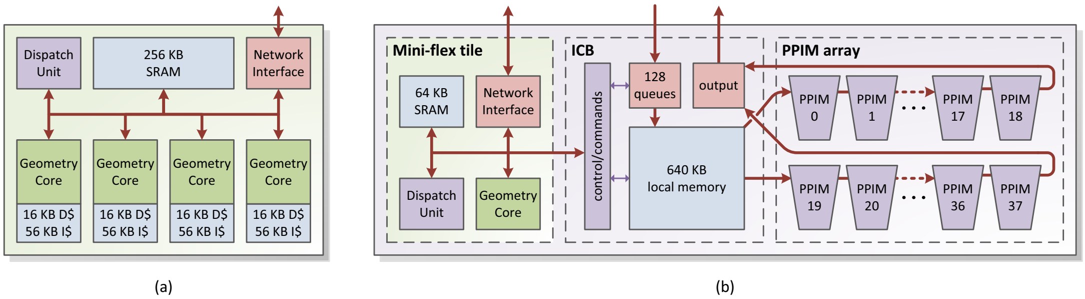


Fig. 4. (a) A flex tile contains 4 geometry cores, 256 KB of shared SRAM, a dispatch unit, a network interface, and an internal network. (b) An HTIS tile contains a mini-flex tile, an interaction control block (ICB), and an array of 38 pairwise point interaction modules (PPIMs). The mini-flex tile contains a geometry core, a dispatch unit, 64 KB of SRAM, and a network interface. The ICB sorts atoms and grid points into 128 queues stored in a 640-KB local memory, loads data from the local memory into the PPIM array, and converts the output of the PPIM array into network packets. The ICB is controlled by a sequence of commands that it receives from the mini-flex GC.

III. Anton 2 System Overview

Anton 2 comprises a number of identical ASICs ("nodes") that are directly connected to form a three-dimensional torus topology (Fig. 3). Inter-node connections consist of two bidirectional 112 Gb/s "channels", providing a total raw bandwidth (data plus encoding overhead) of 224 Gb/s to each of an ASIC's six neighbors within the torus. The ASIC contains an array of two types of computational tiles. There are 16 *flexible subsystem* ("flex") tiles, each with 4 embedded processors called *geometry cores* (GCs), 256 KB of SRAM, a network interface, and a *dispatch unit* [20] that provides hardware support for fine-grained event-driven computation (Fig. 4a). In addition, there are 2 *high-throughput interaction subsystem* (HTIS) tiles, each with an array of 38 *pairwise point interaction modules* (PPIMs) for computing pairwise interactions (Fig. 4b). These tiles are clocked at 1.65 GHz and are connected by an on-chip mesh network consisting of 317 Gb/s bidirectional links [52]. A host interface communicates with an external host processor over a PCI Express link, and a "logic analyzer" captures and stores on-die activity for debugging purposes.

The HTIS tile computes pairwise interactions between two (possibly overlapping) sets of atoms or grid points using the same strategy as the Anton 1 HTIS [29]. It has a maximum throughput of two interactions per cycle per PPIM. The computation is directed by an *interaction control block* (ICB), which receives atoms and grid points from the network, sorts them into 128 queues within a 640-KB local memory, then loads data from the queues into the PPIM array. The ICB is in turn controlled by a sequence of commands that it receives from the GC within a *mini-flex tile*, which is a scaled-down version of the flex tile containing a single GC, 64 KB of SRAM, a dispatch unit, and a network interface. Controlling the ICB with a mini-flex tile simplifies the task of programming Anton 2 by allowing the same toolchain to be used for all embedded software and enabling code sharing between the flex tile and HTIS tile software.

While the GCs and PPIMs have been given the same names as their Anton 1 counterparts, the blocks themselves have been redesigned. The GCs have an entirely new instruction-set architecture, and the PPIM datapaths have fundamentally changed to provide improved performance and flexibility.

IV. Architectural Improvements over Anton 1

Table 1 gives a block-level comparison of the Anton 1 and Anton 2 ASICs. The improvement in inter-node bandwidth is due to a combination of more aggressive signaling technology (14 Gb/s vs. 4.6 Gb/s), more total pins devoted to inter-node signaling (384 vs. 264), and a more efficient physical-layer

TABLE 1. BLOCK-LEVEL COMPARISON OF ANTON 1 AND ANTON 2 ASICS

	Anton 1	Anton 2
Process technology	90 nm	40 nm
Clock speed (GC/PPIM)	485/970 MHz	1.65/1.65 GHz
# of general-purpose processor cores	13	66
# of PPIMs	32	76
Total SRAM + data cache	384 KB	5,280 KB
HTIS memory capacity (atoms)	6,144	32,768
Total data bandwidth to torus neighbors	221 Gb/s	1,075 Gb/s

encoding. The marked increase (14-fold) in on-die SRAM and data cache makes it easier to write embedded software for Anton 2, while significantly increasing the range of chemical system sizes that fit in on-die SRAM.

In addition to these block-level improvements, a number of architectural innovations, described in the following sections, contribute to Anton 2's performance and programmability.

A. *Fine-grained event-driven computation*

One of the biggest improvements over Anton 1 is an architecture tailored for fine-grained operation. To understand the advantages of this approach, consider a simple "range-limited" MD time step with no bond terms or long-range forces[1] consisting of four phases: (1) send atom positions from flex tiles to HTIS tiles, (2) compute interactions within the PPIM array, (3) return forces from HTIS tiles to flex tiles, and (4) integrate atom positions and velocities. With Anton 1's coarse-grained implementation, there is no overlap between these phases: all data must arrive at a tile before computation begins, and all computation must be completed before output data is sent (Fig. 5a). With a fine-grained implementation, individual results are communicated as soon as they have been computed, and individual computations begin as soon as their data is available, irrespective of the order in which the data arrives. This enables significant overlap between the phases (Fig. 5b), and accelerates the range-limited time step by as much as 18%.

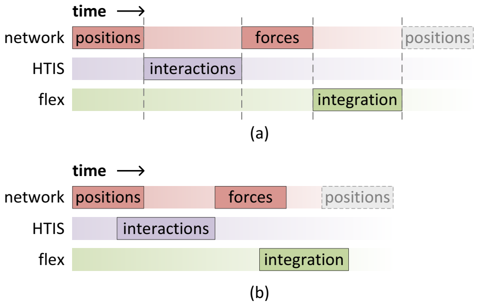


Fig. 5. A range-limited time step with no bond terms consists of (1) sending atom positions to HTIS tiles, (2) computing pairwise interactions, (3) returning forces to flex tiles, then (4) integrating atom positions and velocities, after which positions are sent for the next time step. (a) With a coarse-grained implementation, there is no overlap between communication and computation. (b) A fine-grained implementation allows communication and computation to be substantially overlapped, with the exception that forces cannot be returned from the PPIM array until all interactions have been computed.

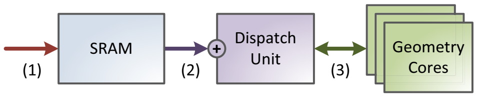


Fig. 6. Event-driven computation on Anton 2. (1) A counted remote write arrives over the network. (2) After the write is processed by SRAM, one or more counters within the dispatch unit are incremented. (3) The geometry cores query the dispatch unit for tasks that are ready to execute (as determined by counters that have reached their thresholds).

The focus on supporting fined-grained operation is reflected in nearly every aspect of the Anton 2 design. The fundamental unit of computation and communication is a 128-bit quad word, which we refer to as a *quad* for brevity. The network efficiently supports very small packets—a typical packet contains 64 header bits and a single quad of data—and in particular every atom position and atom force is transmitted in a separate packet. The ICB queues operate in a streaming fashion, with individual atom positions forwarded to the PPIM array as they arrive over the network.

Fine-grained synchronization and event-driven computation are supported within the flex tiles by a set of hardware counters that reside in the dispatch units. These counters are used to track quad-sized "counted remote writes" [14], wherein a quad write (or accumulation) to a remote SRAM causes one or more counters to be incremented upon receipt. The local GCs can poll the dispatch unit for tasks that are ready to execute, as indicated by counters that have reached their thresholds (Fig. 6). An arbitrary many-to-many mapping from SRAM quads to hardware counters is provided so that software tasks can wait for exactly the data that they need, even if they just need a single quad. The dispatch unit allows these tasks to run in the order in which their input data is received—a scheduling problem that would be prohibitively expensive in software.

In addition to directly improving time-step performance by increasing the overlap of communication and computation, support for fine-grained operation provides a number of secondary benefits. The space of algorithms that can be implemented efficiently is expanded to include those with many small, sequentially dependent tasks; two important examples will be described in Section 6. Data reordering that would ordinarily be performed by software following a communication step can instead be performed directly within the network by sending individual quads to different destination addresses in flex-tile SRAM. Finally, dynamic task scheduling within the dispatch unit enables both load-balancing across the GCs in a flex tile and fine-grained interleaving of tasks with different priorities.

B. *PPIM Improvements*

Another departure from the Anton 1 architecture is a redesign of the PPIM, which computes range-limited interactions between pairs of atoms. Each interaction is typically the sum of separate electrostatic and van der Waals forces. In the Anton 1 PPIM, these forces are computed on the same cycle by two different pipelines, one of which is specialized for the computation of van der Waals forces. The Anton 2 PPIM replaces these heterogeneous pipelines with two identical pairwise point interaction pipelines (PPIPs) that can operate independently on different pairs of atoms; the

[1] This type of time step is encountered in simulations of rigid water molecules (which have no bond terms) with long-range forces evaluated on every other time step.

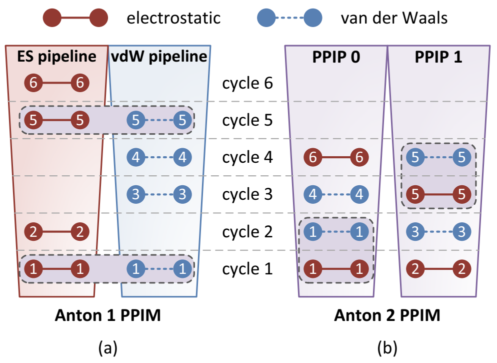


Fig. 7. (a) In the Anton 1 PPIM, electrostatic and van der Waals forces between a pair of atoms are computed on the same cycle by two different pipelines. If one of these forces is not required for a pair of interacting atoms, then the corresponding pipeline is unused during that cycle. (b) The Anton 2 PPIM contains two identical PPIPs. The electrostatic and van der Waals forces between a pair of atoms are computed on consecutive cycles by the same PPIP, improving pipeline utilization.

electrostatic and van der Waals forces are computed on consecutive cycles by the *same* PPIP (Fig. 7).

While both designs offer a peak computational throughput of two force evaluations per cycle, the "generalized" pipeline in Anton 2 has the advantage of maintaining peak throughput in the presence of atom pairs that only require computation of one of the two forces (due to either the properties of the specific atoms, or the use of different cutoff radii for electrostatic and van der Waals forces). In addition, the Anton 1 PPIM can only compute one atom–grid point interaction per cycle, whereas the generalized pipeline allows the Anton 2 PPIM to compute two atom–grid point interactions per cycle (one on each PPIP). In total, the generalized PPIPs improve PPIM efficiency by roughly 35%, which justifies the 25% increase in PPIM area relative to an implementation with a specialized van der Waals pipeline.

The PPIPs also add flexibility by allowing up to four separate forces to be computed on consecutive cycles and added together for a given pair of atoms. This greatly expands the range of interactions that are directly supported. For example, Buckingham potentials [7] modify the van der Waals force in a manner that requires the overall range-limited interaction to be computed on a PPIM as a sum of three forces [41]. On Anton 1, the atoms must be sent through the PPIM array twice in order to compute this interaction, since the PPIMs can only compute the sum of at most two forces between a given pair of atoms. On Anton 2, the atoms only need to be sent through the PPIM array once, with each interaction computed in (up to) three consecutive cycles on a single PPIP.

The Anton 2 PPIM introduces additional flexibility in the ways that per-atom parameters can be combined to form the parameters used for pairwise interactions. Both Anton 1 and MDGRAPE-3 support arithmetic-mean and geometric-mean combining, which is sufficient for the basic electrostatic and van der Waals forces. Anton 2 adds a number of additional combining functions and allows up to six per-atom parameters, providing support for terms such as non-Coulombic interactions [13] and Buckingham potentials. Pairwise parameters can also be retrieved from a two-dimensional table instead of being computed from the individual atom parameters, further increasing flexibility by allowing the computed parameters to be overridden on a per-atom-type basis. In particular, this could allow the PPIM to support coarse-grained models such as the MARTINI force field [33].

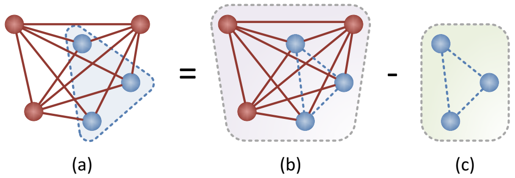


Fig. 8. (a) Three atoms (within the shaded triangle) participate in a bond term. Electrostatic and van der Waals forces (solid lines) are excluded between pairs of these atoms. (b) If the PPIM array computes these undesired interactions (dashed lines), then they must be (c) explicitly recomputed elsewhere and subtracted out. Anton 2 uses topological IDs to avoid computing the majority of these undesired interactions.

C. *Topological IDs*

In addition to delivering improved performance and flexibility, the Anton 2 PPIM contains a novel mechanism to address the fact that non-bonded interactions should *not* be computed between certain pairs of atoms—usually those that participate in a common bond term (Fig. 8a). If these "excluded interactions" are computed within the PPIM array, which is well suited for unconditionally computing interactions between all pairs of atoms separated by less than a cutoff radius, then they must be recomputed elsewhere and subtracted back out as separate correction terms (Fig. 8b,c). Computing all of these correction terms in software would result in a large slowdown, so Anton 1 computes them within a specialized hardware block called the "correction pipeline" [27], which has both hardware and software costs.

In Anton 2, we avoid computing these excluded interactions in the first place, obviating the need for a separate correction pipeline. We do this by labeling atoms with unique identifiers (IDs), carefully constructed based on the molecular topology such that the determination of whether or not two atoms need to interact can be made, with high accuracy, simply by inspecting their IDs. These *topological IDs* are able to eliminate over 95% of the excluded interactions, leaving a small number of correction terms that can be computed by software without adversely affecting performance. Disabling the topological ID mechanism and computing software corrections for all excluded interactions reduces simulation performance by up to 1.8 times, underscoring the value of hardware support.

1) Constructing topological IDs

Consider the "bond graph" defined by covalent bonds between atoms (Fig. 9a). For most molecules, nearly every pair of atoms separated by at most three edges represents an excluded interaction. It therefore suffices to construct a topological ID such that the distance between two atoms within the graph (their *topological distance*) can be computed based on their topological IDs. If the graph consisted solely of a long linear chain (a "backbone"), then we could simply label the atoms with consecutive positive integers, and the distance between atoms with labels n_1 and n_2 would be $|n_1 - n_2|$ (Fig. 9b). We extend this simple labelling to handle proteins and lipids by

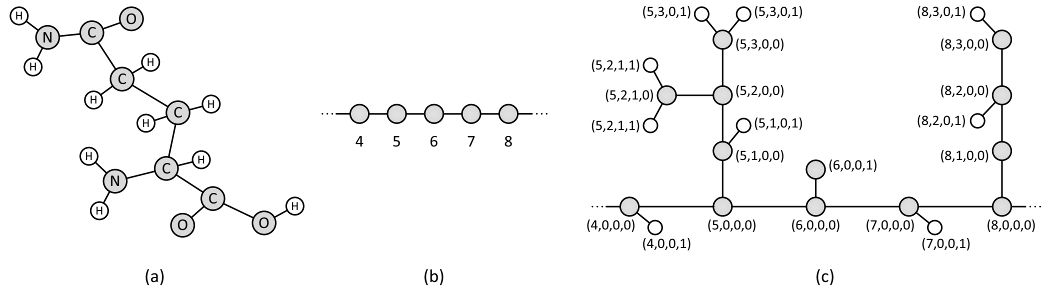


Fig. 9. (a) The bond graph of a molecule (shown for glutamine) is defined by the covalent bonds between atoms. (b) A linear graph could be labeled with consecutive integers; the topological distance between two atoms is just the absolute difference of their labels. (c) Example subgraph labeled with actual topological IDs (n, m, k, t) where n is the backbone index, m is the side-chain index, k is the secondary side-chain index, and $t \in \{0, 1\}$ is a "terminal" flag.

defining the topological ID as a tuple (n, m, k, t), where n, m, k are integers and t is a "terminal" flag, as follows:

- Backbone atoms are labeled (n, 0, 0, 0).
- Side-chain atoms are labeled (n, m, 0, 0), where $m \geq 1$ is the distance from the backbone atom (n, 0, 0, 0).
- Secondary side-chain atoms are labeled (n, m, k, 0), where $k \geq 1$ is the distance from the side-chain atom (n, m, 0, 0).
- Terminal atoms (having a single neighbor in the graph) are labeled (n, m, k, 1), where (n, m, k, 0) is the label of their neighbor.

Fig. 9c shows a portion of a graph labeled according to these rules. This labelling could naturally be further extended to handle tertiary (or deeper) side chains, but we found that secondary side chains are sufficient to label nearly all protein and lipid atoms. The topological distance between two atoms with IDs (n_1, m_1, k_1, t_1) and (n_2, m_2, k_2, t_2) is

$$\begin{array}{ll} |n_1 - n_2| + m_1 + m_2 + k_1 + k_2 + t_1 + t_2 & \text{if } n_1 \neq n_2, \\ |m_1 - m_2| + k_1 + k_2 + t_1 + t_2 & \text{if } n_1 = n_2 \text{ and } m_1 \neq m_2, \\ |k_1 - k_2| + t_1 + t_2 & \text{if } n_1 = n_2 \text{ and } m_1 = m_2. \end{array}$$

The above construction assumes that the graph is a tree with at most one side chain (or secondary side chain) emanating from any given atom. Both proteins and lipids contain exceptions to these rules, which we handle by strategically removing a small number of edges from the graph until the assumptions hold (Fig. 10). Topological IDs are then assigned based on the modified graph.

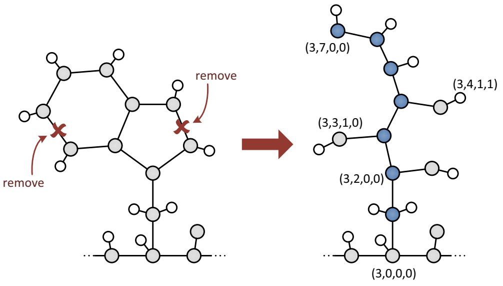


Fig. 10. The bond graph for a tryptophan residue (left) contains cycles, so it must be modified before topological IDs are assigned. Two edges are removed as indicated, resulting in the graph on the right. The side chain within the modified graph is highlighted, and several topological IDs are shown.

2) *Encoding the IDs*

Since topological IDs must be bundled with atom positions as they are communicated over the network and through the PPIM array, their encoding should be as compact as possible. We begin with the empirical observation that the side chains are short: for the vast majority of protein and lipid atoms, three bits suffice to encode m, and a single bit suffices to encode k. This suggests a simple L-bit encoding using five bits for (m, k, t) and the remaining $L - 5$ bits for n.

Anton 2 uses a variation of this simple encoding with 21-bit topological IDs, which provides enough encoding space for chemical systems containing several million atoms. Chemical systems larger than this require an increasing number of correction terms to be computed by software, resulting in graceful performance degradation relative to an implementation with wider topological IDs. The all-zero encoding is reserved for atoms that cannot be labeled with a topological ID, either because the encoding space has been exhausted, or because too many bits would be required for m or k, or because the atom was removed when the graph was modified as described in the previous section.

V. Programming Model

Having discussed the hardware design of Anton 2, we now turn our attention to its programming model, which differs significantly from that of Anton 1. All Anton 2 embedded software is written in C++ and compiled using a version of the GNU Compiler Collection (GCC) that has been ported to the GCs. The compiled software runs on the GCs, which are 32-bit fixed-point processor cores with four-way SIMD ALUs. The ALU has two multipliers per lane and supports up to eight fixed-point multiplications and eight additions in a single instruction (there is a dual complex multiply-add instruction, for example).

Special load/store instructions provide quad-sized access to the flex-tile SRAM. Any GC can read or write any SRAM quad within the entire machine, presenting a distributed shared memory model to software. In addition to quad writes, the SRAM supports three read-modify-write operations: SET (bitwise OR), CLEAR (bitwise AND with complement), and ADD (4-way SIMD 32-bit addition). The ADD operation is useful for performing accumulations within memory, especially forces within an MD simulation. The SRAM also supports atomic memory operations, which atomically modify a quad in memory (via a write or a read-modify-write) and return the previous value of the quad.

The dispatch units support an event-driven programming model, which we have adopted for the Anton 2 software. All software is expressed as a collection of tasks (such as integrating an atom's positions and velocities) that run in response to the arrival of their input data. Before an application runs, these tasks and their dependencies must be specified; a software library then automatically maps the tasks to counters with the appropriate thresholds. At run time, each GC runs an event loop, alternately querying the local dispatch unit for work, then performing the next task that is ready for execution.

The Anton 2 embedded programming interface is notably simpler than that of Anton 1. There are three distinct types of embedded processor in Anton 1, which are programmed using a mix of C and assembly, and must be carefully coordinated with synchronization protocols involving hardware queues. Anton 2 has a single type of embedded processor, which is programmed in C++. The read-modify-write operations supported by the Anton 2 SRAM are powerful programming primitives, and in particular they simplify the accumulation of atom forces, which in Anton 1 require writes to a DRAM controller. Finally, fine-grained packets (including counted remote writes) can be launched directly from the Anton 2 GC with a single write instruction; on Anton 1 software must assemble larger packets in SRAM, issue a DMA operation to send the packets, and then synchronize with the DMA unit to determine when the SRAM can be reused.

VI. Algorithms Enabled by Anton 2

The improved programmability of Anton 2, particularly its support for fine-grained event-driven execution, has enabled several new algorithms, two of which are described in the following sections. These algorithms share the characteristic that they are multi-staged, with many small, sequentially dependent tasks. The network provides support for the fine-grained (often single-quad) communication required by these tasks, and the dispatch unit allows the tasks to run as soon as their data has arrived and a GC is idle.

A. *Bond proxies*

The first algorithm reduces the communication bandwidth for bond terms. This communication arises from the fact that, on each time step, the atoms within every bond term (typically 2–4 atoms per bond term) must be brought together to compute the forces arising from the term. Because the set of bond terms is fixed for the duration of a simulation, each term can be statically assigned to a "bond worker" flex tile, so that the atoms within the term are sent to the same known location on every time step. After the bond worker receives the atom positions, it computes the bond term forces, which are then returned for accumulation with other forces (Fig. 11a). This strategy was described for Anton 1 [14], and it has also been adopted for Anton 2.

The communication costs associated with bond terms are quite large: a chemical system can easily contain more than twice as many bond terms as atoms, and a given atom can participate in over 50 bond terms, requiring that it be sent to many bond workers. As the chemical system evolves, these costs increase, since the bond workers are fixed but the atoms must migrate between nodes as their positions change. Thus, an assignment of bond terms to nodes that minimizes communication for the initial atom placement becomes increasingly suboptimal as the system evolves, with gradual increases in both the bandwidth and latency of bond-term communication. On Anton 1, we improved performance by as much as 14% by periodically reassigning bond terms to nodes based on updated atom locations.

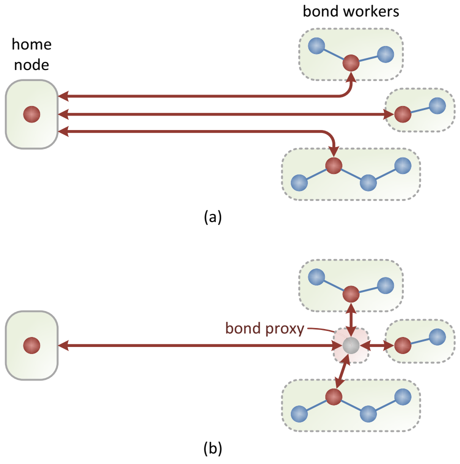


Fig. 11. (a) On Anton 1, an atom is sent from its home node to multiple bond workers; the bond workers return forces to the home node. This requires significant inter-node communication when the home node is far away from the worker nodes in the machine. (b) Communication overhead is reduced on Anton 2 by sending the atom to a single bond proxy, which forwards the atom to the multiple bond workers. Forces are returned from the workers to the proxy, which sends a single aggregate force back to the home node.

On Anton 2, we reduce the overhead of bond-term communication with a new algorithm in which each atom is sent to a single intermediate *bond proxy* rather than sending it directly to its bond workers; the bond proxy then forwards the atom to the workers. This communication pattern is reversed for force return: the bond workers return forces for the atom to the bond proxy, where they are accumulated in SRAM, then the bond proxy returns a single aggregate force for the atom (Fig. 11b). Placing an atom's bond proxy close to its bond workers significantly reduces the bond-term communication bandwidth as the atom itself migrates through the machine, and the overall performance degradation due to atom migration as an MD simulation evolves is less than 4% on average. This removes most of the impetus to complicate the software by periodically reassigning bond terms to new workers. The bond-proxy tasks are extremely small, with the smallest task (returning the aggregate force for a single atom) consisting of reading a single quad from local SRAM (the force) and forwarding it to its destination using a counted remote write.

B. *Real-space mesh convolution*

The second algorithm reduces the communication latency associated with long-range electrostatic forces. In most MD implementations, these forces are computed using Ewald summation [12, 17], which decomposes the electrostatic force into the sum of two terms: a rapidly decaying "range-limited" term that can be explicitly computed between pairs of atoms up to a cutoff radius, and a slowly decaying "long-range" term that can be efficiently computed using a grid-based convolution. Mathematically, it suffices to describe how the net electrostatic potential energy is divided among these terms; the force on a given atom is then the negative gradient of this energy with respect to the atom's position. Anton 1 was designed to use the Gaussian split Ewald method [43], for which the total long-range electrostatic energy E^{σ} can be expressed as

$$E^{\sigma} = \left(\rho * G^{\sigma_1}\right) * \left(G^{\sigma_2} * \gamma\right) * \left(G^{\sigma_1} * \rho\right) \qquad (1)$$

where $*$ denotes convolution, ρ is the net charge distribution, G^{σ} is a normalized Gaussian with standard deviation σ, $\gamma(\mathbf{x}) = 1/|\mathbf{x}|$, and σ_1, σ_2 are constants. The convolutions with ρ are computed using range-limited pairwise interactions of the atoms with the points of a regular grid; the remaining convolutions are computed on the grid itself. Since convolution becomes multiplication in Fourier space, this is done by performing a discrete Fourier transformation on the grid, followed by a complex multiplication at each grid point, followed by an inverse Fourier transformation.

Anton 2 was originally designed to implement this same method; the GCs even include specialized, high-density arithmetic instructions for accelerating fast Fourier transformations (FFTs). Recently, however, a different decomposition called the *u*-series [40] was implemented on Anton 1, and we were able to leverage its properties to gain additional performance on Anton 2 by eliminating the FFT. The *u*-series expresses the long-range electrostatic potential energy ϕ^{σ} between a pair of atoms with unit charge as a sum of Gaussians:

$$\phi^{\sigma}(r) = 2\log(b)\sum_{j=0}^{N-1} G^{\sigma b^j}(r) \qquad (2)$$

where r is the distance between the atoms, $b > 1$ is a fixed constant, $\sigma^2 = 2\sigma_1{}^2 + \sigma_2{}^2$, and N is a small positive integer (typically 4–6, depending on the chemical system size). This sum rapidly converges to $1/r$ as r grows large. With this decomposition, the total long-range electrostatic energy becomes

$$E^{\sigma} = 2\log(b)\left(\rho * G^{\sigma_1}\right) * \left(\sum_{j=0}^{N-1} G^{\sqrt{\sigma^2 b^{2j} - 2\sigma_1^2}}\right) * \left(G^{\sigma_1} * \rho\right) \qquad (3)$$

The convolutions with ρ are again computed using range-limited pairwise interactions of atoms and grid points. Instead of computing the remaining convolutions in Fourier space, we take advantage of the fact that a three-dimensional convolution with a Gaussian can be efficiently computed in *real* space by performing the convolution one dimension at a time. This results in an algorithm with significantly reduced communication latency: on an 8×8×8 machine, for example, the FFT-based approach requires six sequential rounds of communication with at least 24 inter-node hops on the critical path,[2] whereas the real-space algorithm requires three rounds of communication with a total of 12 hops (this is the theoretical minimum, since it is the diameter of an 8×8×8 torus network).

The drawback of the real-space approach is that each of the N Gaussians within the *u*-series sum must be convolved separately, then the results added together at the end, so the improvement in communication latency comes at the cost of increased communication bandwidth and computation. This is a good tradeoff for Anton 2 because there are many GCs to absorb the additional computational kernels, the scheduling of these kernels is handled by the dispatch unit, and the kernels themselves are very fast owing to a dual 4-vector dot-product (*ddot4*) instruction. A typical kernel multiplies a 4×32 matrix of input values by a 32×1 matrix of coefficients to produce a 4×1 matrix of output values, which on a GC requires only 23 arithmetic instructions (16 ddot4s and 7 adds). Computing the mesh convolution in real space rather than Fourier space improves the overall performance of MD simulations with long-range forces computed on every other time step by up to 15%.

VII. Performance

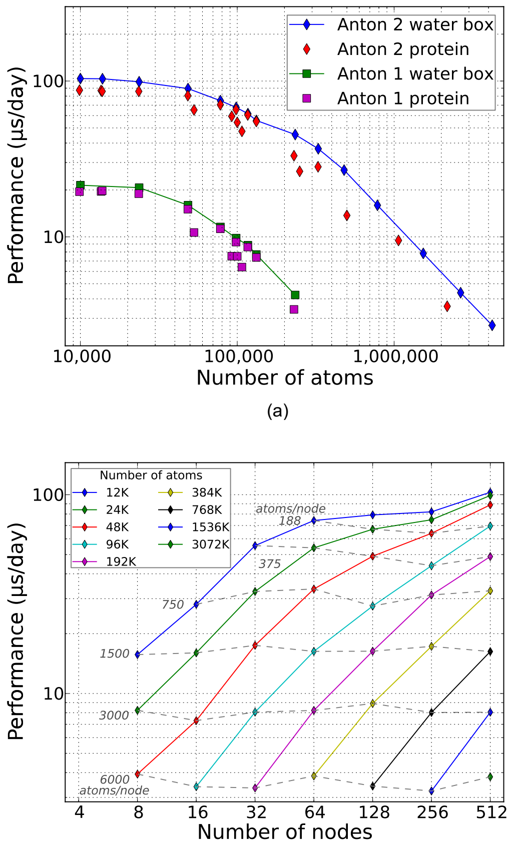
Our performance experiments were conducted using a 512-node Anton 2 machine configured as an 8×8×8 torus with all nodes running at 1.65 GHz. All chemical systems were run with a 2.5-fs time step. Long-range forces were computed on every other time step using the *u*-series method with $b = 2$ and N between 4 and 6, depending on system size. These form part of our "production parameters", that is, the parameters used for actual biochemical research on both Anton 1 and Anton 2.

Fig. 12a shows performance for both proteins in water and pure water ("water box") systems ranging in size from 10,000 atoms to 4.2 million atoms. The water-box simulations contain no bond terms, so they tend to give an upper bound on performance for a given system size. Performance is also shown for Anton 1, using the same production parameters as on Anton 2, for the systems that are within the capacity of a 512-node Anton 1 machine. The performance improvement relative to Anton 1 ranges from 4.4 times for the smallest protein system to 10.7 times for a water box with 234,000 atoms. This water box represents the approximate capacity of a 512-node Anton 1 machine with current software, as it is the largest system we have been able to fit within on-die SRAM.[3] The within-SRAM capacity of a 512-node Anton 2 machine is around 18 times larger than that of Anton 1, with a 4.2-million-atom water-box simulation running at 2.7 μs/day.

The fastest simulations shown in Fig. 12a are 10,000- and 14,000-atom water boxes that run at over 103 μs/day. Although these are just simple test systems, achieving this performance for *any* system on any platform is a challenge, requiring an average wall-clock time of less than 2.1 μs per time step. By contrast, the fastest simulations on general-purpose supercomputers are measured in hundreds of nanoseconds of physical time per day, or hundreds of microseconds of wall-clock time per time step. To our knowledge, the only report of exceeding 1 μs/day using commodity hardware is a GPU-accelerated

[2] The FFT implementation on Anton 1 used additional inter-node hops (32 or 34) because it was optimized for bandwidth rather than latency [54].

[3] One can also write software for both Anton 1 and Anton 2 that pages data to/from off-chip DRAM, which would increase their effective capacity but incur a significant performance penalty. Earlier versions of the Anton 1 software supported this mode of operation with a 2–3× slowdown, but we found that it is much more effective to build larger machines to handle larger chemical system sizes. A 2,048-node Anton 1 machine was built in 2009, for example.



(a)

(b)

Fig. 12. (a) Performance versus system size for protein and water-box systems on both Anton 1 and Anton 2. All simulations were run using the same production parameters, which include a 2.5-fs time step and long-range forces computed every other time step. (b) Performance versus machine size for water-box systems on Anton 2. Smaller machines are obtained by software-partitioning the full 8×8×8 machine in one or more dimensions.

simulation of an 8,000-atom system, running at 1.1 µs/day with an average wall-clock time per time step of 393 µs [31].[4]

Anton 2 demonstrates significant gains in throughput over Anton 1 along two different axes: performance and chemical system size. For a fixed chemical system, Anton 2 achieves simulation rates up to 10.7 times faster than Anton 1. For a fixed target simulation rate (determined, for example, by the timescale of a biological process under study), Anton 2 can simulate systems over ten times larger than an Anton 1 machine with the same number of nodes.

A 512-node Anton 2 machine can be software-partitioned along one or more dimensions into multiple smaller machines, using the inter-node network as a mesh rather than a torus along the partitioned dimensions (the use of a mesh incurs a small performance penalty, typically around 5–10%). Fig. 12b shows how performance scales with machine size for a variety of water-box systems. The solid curves highlight strong scaling on Anton 2, while the dashed lines show that performance is largely a function of the number of atoms per node.

TABLE 2. PERFORMANCE FOR VARIOUS CHEMICAL SYSTEMS ON ANTON 2 WITH 512 NODES, ANTON 1 WITH 512 NODES, AND OTHER HARDWARE.

Performance numbers are in units of microseconds of physical time per day. General-purpose hardware performance numbers are for (a) GROMACS on a Xeon E5-2690 processor with an NVIDIA GTX TITAN GPU, running with a triclinic cell to reduce the number of atoms to 8,000 [31], (b) Desmond on 1,024 cores of a Xeon E5430 cluster [10], (c) ACEMD on four NVIDIA GTX TITAN GPUs [1], (d) NAMD on 4,096 cores of Cray Jaguar XK6 [48], (e) Amber on an NVIDIA GTX TITAN GPU [53], (f) GROMACS on four NVIDIA GTX680 GPUs [31], (g) NAMD on 65,536 cores of IBM Blue Gene/Q [26], (h) NAMD on 4,096 cores of Cray Jaguar XK6 [48], (i) NAMD on 32,768 cores of Cray Jaguar XT4 [5], (j) NAMD on 16,384 cores of Cray Jaguar XK6 [48], (k) NAMD on 65,536 cores of Blue Gene/L [5], (l) GROMACS on 16,384 cores of the IBM SuperMUC cluster, running a smaller version of the system, with 2.1 million atoms [28].

	Villin	**DHFR**	**ApoA1**	**ATPase**	**STMV**	**Ribosome**
# atoms	13,773	23,558	92,224	327,506	1 M	2.2 M
Anton 2 (µs/day)	85.8	85.8	59.4	28.2	9.5	3.6
Anton 1 (µs/day)	19.7	18.9	7.5	—	—	—
General-purpose hardware (µs/day)	1.1[a]	0.471[b] 0.235[c] 0.144[d] 0.108[e] 0.100[f]	0.289[b] 0.127[g] 0.076[h]	0.039[i]	0.035[j] 0.018[k]	0.171[l]

A. Accuracy

The accuracy of MD simulations on Anton 2 is a function of both numerical precision and the specific set of parameters used for simulations. The precision and simulation parameters used for Anton 1 have been well-validated over the course of five years, during which over 220 million node hours have been used to simulate a total of over 280 ms of physical time across a wide variety of chemical systems. Anton 2 uses the same fixed-point format for positions and forces as Anton 1, with slightly improved precision for certain computations in both the GCs and the PPIMs. The root-mean-square relative force error compared to a double-precision floating-point implementation[5] is 4.2×10^{-6}.

A full justification of the production parameters used for biochemistry research on Anton 1 and Anton 2 is well beyond the scope of this paper, but we present one example of the type of validation that is required, particularly in light of the relatively new *u*-series method for long-range electrostatics. Using a 160,000-atom lipid bilayer system containing 512 DPPC lipid molecules, we measure the electrostatic potential drop between water and the center of the lipid bilayer, which forms a very sensitive test for the accuracy of the long-range electrostatics. With our *u*-series parameters, we observe an error in the electrostatic potential drop of only 0.5%. By comparison, the use of a 2.5-fs time step (instead of a conservative 1.0-fs time step), which we have used extensively for long-timescale simulations, introduces an error of roughly 1.5%.

B. Comparison to other platforms

In Table 2 we compare the performance of Anton 2 to both Anton 1 and general-purpose-hardware-based implementations

[4] The Anton 2 water-box simulation ran at ~103 µs/day using a 2.5-fs time step, giving an average time-step time of 86.4 × 2.5 / 103 ≈ 2.1 µs. The GPU simulation ran at 1.1 µs/day using a 5-fs time step, giving an average time-step time of 86.4 × 5 / 1.1 ≈ 393 µs.

[5] Forces were compared to those produced by Desmond 3.6.0.3 for a simulation of DHFR using Anton 2 production parameters.

of MD for several of the systems in Fig. 12a. Performance data for other implementations was obtained from the literature. Direct comparisons to other implementations are complicated by the fact that each implementation has a slightly different set of preferred production parameters. The time step, in particular, varies from 1 fs to 5 fs. Generally speaking, shorter time steps improve simulation accuracy and allow long-range electrostatics to be computed less frequently. Longer time steps can improve the overall simulation rate, but require long-range electrostatics to be computed more often, and rely on techniques such as holonomic constraints [11] to eliminate the fastest vibrations. Rather than attempting to normalize the results, we simply present the performance numbers as they appear in the literature.[6] Despite recent advances in FPGA-based implementations of MD [2, 24, 25, 42], we are unaware of any such implementations whose performance is comparable to GPU-based or interconnected-CPU implementations.

For most of the systems, Anton 2 is more than two orders of magnitude faster than any general-purpose-hardware-based implementation of MD. The exceptions are the smallest system, a model of the villin headpiece that achieves outstanding performance on a single GPU running GROMACS 4.6, and the largest system, a 2.2-million-atom ribosome model that leverages weak scaling on a very large cluster. Even on this large chemical system, Anton 2 maintains a 21-fold performance advantage. Moreover, while the largest Anton 2 machine constructed to date has 512 nodes, the architecture itself scales to 4,096 nodes. Significantly higher simulation rates could be achieved for the larger chemical systems by running on an Anton 2 machine with more nodes, while the GROMACS simulation was performed near the limit of scaling and would have received little benefit from additional cluster nodes.

It is worth noting that, in addition to improving performance, the use of special-purpose hardware results in extremely energy-efficient computation. For example, the GROMACS ribosome simulation was run on 16,384 cores of the IBM SuperMUC cluster, consuming roughly 316 kW of power [19]. By contrast, a 512-node Anton 2 machine consumes around 150 kW of power. Thus, Anton 2 is able to run the ribosome simulation 21 times faster using less than half as much power—more than a 40-fold improvement in energy efficiency.

C. *One millisecond in two weeks*

Millisecond-long MD simulations can be used to study biological phenomena that occur over corresponding timescales. As an example, a 1.119-ms simulation was performed on Anton 1 to study the folding kinetics of the Fip35 WW domain protein (Fig. 13) [37]. This 10,000-atom simulation took a total of 164 days of machine time to run on Anton 1 using 128 nodes. We ran a new simulation of this system for the same amount of physical time on a 64-node Anton 2 machine (with the network configured as a 4×4×4 torus). The entire simulation took 16.9 days of machine time, and it reached a millisecond in just over two weeks of machine time (15.2 days). This is a 9.7-fold improvement in time-to-solution over the Anton 1 simulation despite using only half as many nodes.

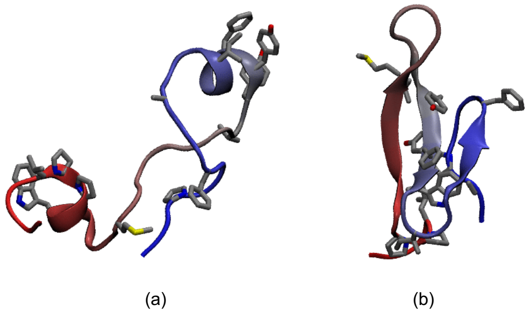


Fig. 13. Fip35 WW domain in an (a) unfolded and (b) folded state.

VIII. Conclusion

Anton 2 advances the state of the art for both performance and programmability of a special-purpose molecular dynamics supercomputer. Simulations on Anton 2 run 4–10 times faster than on Anton 1, and over two orders of magnitude faster than on any general-purpose hardware, for a wide range of system sizes. Achieving this performance required a variety of techniques, including an architecture optimized for fine-grained operation, more efficient hardware pipelines, a novel topological ID mechanism, and a set of algorithms enabled by hardware support for event-driven computation.

The gains in performance on Anton 2 are coupled with improved flexibility. In addition to accelerating simulations with the most commonly used MD force fields, Anton 2 contains direct support for a variety of force field extensions, including coarse-grained models and non-standard pairwise interactions. This support, combined with improved programmability, makes Anton 2 a powerful platform for alternate force fields and force field research.

While Anton 2 was designed to accelerate MD simulations, many of the mechanisms it employs to do so—namely its support for efficient fine-grained communication, computation, and synchronization—would be of benefit to a broader class of scientific computation. Applications as diverse as neural networks, sparse linear algebra, and fluid dynamics, to name three examples, can all be naturally expressed as large numbers of small, event-driven computations. The ability to send counted remote writes with a single instruction, perform accumulations within memory and synchronize on individual quads of data makes it easier to write software that directly captures the dataflow of the underlying computation. Moreover, the dispatch unit allows computations to be expressed as a collection of fine-grained tasks without the burden of having to explicitly schedule these tasks in software.

Our qualitative experience in implementing MD software for Anton 2 is that it is much easier to program than Anton 1, primarily due to its support for fine-grained operation and the use of a homogeneous set of embedded processor cores. Although highly specialized, Anton 2 delivers significant programmability and flexibility, which will facilitate its use in advancing the frontiers of biomolecular simulation.

[6] NAMD performance numbers are typically reported as average wall-clock time-step time, with long-range electrostatics computed every four time steps. These parameters correspond to a 1-fs time step [36], which we use to convert the reported average time-step times to simulation rates.

Acknowledgments

We thank Cristi Predescu for developing the *u*-series decomposition; Robert Dirks for assistance with the DPPC simulation; Stefano Piana for assistance with the WW-domain simulation; Michael Bergdorf, Alex Donchev, Brent Gregersen, John Klepeis, Tom Kneeland, Je-Luen Li, Kim Palmo and Andrew Taube for sharing insights into biomolecular force field requirements; David Borhani, Michael Eastwood, Justin Gullingsrud, Morten Jensen, Kresten Lindorff-Larsen, Paul Maragakis, Albert Pan, Yibing Shan and Huafeng Xu for helpful discussions about biochemistry algorithms and for providing biochemical systems for our performance measurements; Peter Bogart Johnson, Cariann Chan, An Do, George Gemelos and Anissa Harper for dealing with countless operational details; Tom Hart, James Keithan, Purvesh Khona, Bipul Talukdar and Bill Vick for assistance with design verification; Wilson Chan for helping with physical design; Stefan Freudenberger for many compiler optimizations and bug fixes; Mike Coleman, Eric Radman and Reinhard Tartler for developing system software infrastructure; Michael Fenn, Chris Harwell, Doug Hughes, Nathan Olla and Goran Pocina for the design and implementation of the host and front-end systems and networks; Eric Bovell for mechanical and thermal concepts; Chuck Rendleman and Ross Lippert for comparison of results with Desmond; Anton Arkhipov, Tarun Chitra, Hillary Green-Lerman, Sahar Shahamatdar, James Valcourt and Victor Zhao for preparing biochemical systems and testing the correctness of simulations; and Mollie Kirk for editorial assistance.

[b] New affiliations of indicated authors: Jack C. Chao: Columbia University; Martin M. Deneroff: GreenWave Systems; Ron O. Dror: Stanford University; C. Richard Ho, Chester Li, Rolf Mueller, U. Ben Schafer, Benjamin Vitale, Cliff Young: Google; Daniel Killebrew: DNAnexus; Richard H. Larson, Shark Yeuk-Hai Mok: Apple; Timothy Layman: Cadence Design Systems; Li-Siang Lee: Barefoot Networks; Daniel Ramot: Via; Christopher W. Snyder: Edgeflip; Ping Tak Peter Tang: Intel; Horia Toma: Synopsys.

References

[1] Acellera, "ACEMD: High-throughput molecular dynamics with NVIDIA Kepler GPUs", http://www.slideshare.net/CanOzdoruk/gtc-expressacemdwebinar, retrieved April 2, 2014.

[2] Sadaf R. Alam, Pratul K. Agarwal, Melissa C. Smith, Jeffrey S. Vetter and David Caliga, "Using FPGA Devices to Accelerate Biomolecular Simulations", *IEEE Computer*, Volume 40, Issue 3, March, 2007, pp. 66–73.

[3] Yoshimichi Andoh, Noriyuki Yoshii, Kazushi Fujimoto, Keisuke Mizutani, Hidekazu Kojima, Atsushi Yamada, Susumu Okazaki, Kazutomo Kawaguchi, Hidemi Nagao, Kensuke Iwahashi, Fumiyasu Mizutani, Kazuo Minami, Shin-ichi Ichikawa, Hidemi Komatsu, Shigeru Ishizuki,Yasuhiro Takeda and Masao Fukushima, "MODYLAS: A Highly Parallelized General-Purpose Molecular Dynamics Simulation Program for Large-Scale Systems with Long-Range Forces Calculated by Fast Multipole Method (FMM) and Highly Scalable Fine-Grained New Parallel Processing Algorithms", *Journal of Chemical Theory and Computation*, Volume 9, Issue 7, July 9, 2013, pp. 3201–3209.

[4] Anton Arkhipov, Yibing Shan, Rahul Das, Nicholas F. Endres, Michael P. Eastwood, David E. Wemmer, John Kuriyan and David E. Shaw, "Architecture and Membrane Interactions of the EGF Receptor", *Cell*, Volume 152, Issue 3, January 31, 2013, pp. 557–569.

[5] Abhinav Bhatelé, Sameer Kumar, Chao Mei, James C. Phillips, Gengbin Zheng, Laxmikant V. Kalé, "Overcoming Scaling Challenges in Biomolecular Simulations across Multiple Platforms", *22nd IEEE International Parallel and Distributed Processing Symposium*, Miami, FL, April 14–18, 2008, pp. 1–12.

[6] Kevin J. Bowers, Edmond Chow, Huafeng Xu, Ron O. Dror, Michael P. Eastwood, Brent A. Gregersen, John L. Klepeis, Istvan Kolossvary, Mark A. Moraes, Federico D. Sacerdoti, John K. Salmon, Yibing Shan and David E. Shaw, "Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters", *2006 ACM/IEEE Conference on Supercomputing*, Tampa, FL, November 11–17, 2006.

[7] Richard A. Buckingham, "The Classical Equation of State of Gaseous Helium, Neon and Argon", *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, Volume 168, Number 933, October 25, 1938, pp. 264–283.

[8] Nicolas Calimet, Manuel Simoes, Jean-Pierre Changeux, Martin Karplus, Antoine Taly and Marco Cecchini, "A gating mechanism of pentameric ligand-gated ion channels", *Proceedings of the National Academy of Sciences of the United States of America*, Volume 110, Number 42, October 15, 2013, pp. 3987–3996.

[9] David A. Case, Thomas E. Cheatham III, Tom Darden, Holger Gohlke, Ray Luo, Kenneth M. Merz Jr., Alexey Onufriev, Carlos Simmerling, Bing Wang and Robert J. Woods, "The Amber Biomolecular Simulation Programs", *Journal of Computational Chemistry*, Volume 26, Issue 16, December, 2005, pp. 1668–1688.

[10] Edmond Chow, Charles A. Rendleman, Kevin J. Bowers, Ron O. Dror, Douglas H. Hughes, Justin Gullingsrud, Federico D. Sacerdoti and David E. Shaw, "Desmond Performance on a Cluster of Multicore Processors", Technical Report DESRES/TR–2008-01, https://www.deshawresearch.com/publications/Desmond%20Performance%20on%20a%20Cluster%20of%20Multicore%20Processors.pdf, retrieved February 8, 2014.

[11] Giovanni Ciccotti and Jean-Paul Ryckaert, "Molecular Dynamics Simulation of Rigid Molecules", *Computer Physics Reports*, Volume 4, Issue 6, September–October, 1986, pp. 356–392.

[12] Tom Darden, Darrin York and Lee Pederson, "Particle mesh Ewald: An *N*·log(*N*) method for Ewald sums in large systems", *Journal of Chemical Physics*, Volume 98, Number 12, June 15, 1993, pp. 10089–10092.

[13] A. G. Donchev, V. D. Ozrin, M. V. Subbotin, O. V. Tarasov and V. I. Tarasov, "A quantum mechanical polarizable force field for biomolecular interactions", *Proceedings of the National Academy of Sciences*, Volume 102, Number 22, May 31, 2005, pp. 7829–7834.

[14] Ron O. Dror, J.P. Grossman, Kenneth M. Mackenzie, Brian Towles, Edmond Chow, John K. Salmon, Cliff Young, Joseph A. Bank, Brannon Batson, Martin M. Deneroff, Jeffrey S. Kuskin, Richard H. Larson, Mark A. Moraes and David E. Shaw, "Exploiting 162-Nanosecond End-to-End Communication Latency on Anton", *2010 International Conference for High Performance Computing, Networking, Storage and Analysis*, New Orleans, LA, November 13–19, 2010.

[15] Ron O. Dror, Robert M. Dirks, J.P. Grossman, Huafeng Xu and David E. Shaw, "Biomolecular Simulation: A Computational Microscope for Molecular Biology", Annual Review of Biophysics, Volume 41, June, 2012, pp. 429–452.

[16] Ron O. Dror, Hillary F. Green, Celine Valant, David W. Borhani, James R. Valcourt, Albert C. Pan, Daniel H. Arlow, Meritxell Canals, J. Robert Lane, Raphaël Rahmani, Jonathan B. Baell, Patrick M. Sexton, Arthur Christopoulos and David E. Shaw, "Structural basis for modulation of a G-protein-coupled receptor by allosteric drugs", *Nature*, Volume 503, Number 7475, November 14, 2013, pp. 295–299.

[17] Paul P. Ewald, "Die Berechnung optischer und elektrostatischer Gitterpotentiale", *Annalen der Physik*, Volume 369, Issue 3, 1921, pp. 253–287.

[18] Blake G. Fitch, Aleksandr Rayshubskiy, Maria Eleftheriou, T.J. Christopher Ward, Mark Giampapa, Yuri Zhestkov, Michael C. Pitman, Frank Suits, Alan Grossfield, Jed Pitera, William Swope, Ruhong Zhou, Scott Feller and Robert S. Germain, "Blue Matter: Strong Scaling of Molecular Dynamics on Blue Gene/L", *Lecture Notes in Computer Science*, Volume 3992, 2006, pp. 846–854.

[19] The Green 500, "The Green500 List – November 2013", http://www.green500.org/lists/2013/11/top/green500_top_201311.xls, retrieved April 15, 2014.

[20] J.P. Grossman, Jeffrey S. Kuskin, Joseph A. Bank, Michael Theobald, Ron O. Dror, Douglas J. Ierardi, Richard H. Larson, U. Ben Schafer, Brian Towles, Cliff Young and David E. Shaw, "Hardware Support for Fine-Grained Event-Driven Computation in Anton 2", *18th International Conference on Architectural Support for Programming Languages and Operating Systems*, Houston, TX, March 16–20, 2013, pp. 549–560.

[21] Matt J. Harvey, Giovanni Giupponi and Gianni De Fabritiis, "ACEMD: Accelerated molecular dynamics simulations in the microsecond time-scale", *Journal of Chemical Theory and Computation*, Volume 5, Issue 6, June 9, 2009, pp. 1632–1639.

[22] Berk Hess, Carsten Kutzner, David van der Spoel and Erik Lindahl, "GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation", *Journal of Chemical Theory and Computation*, Volume 4, Issue 3, February 2008, pp. 435–447.

[23] Morten Ø. Jensen, Vishwanath Jogini, David W. Borhani, Abba E. Leffler, Ron O. Dror and David E. Shaw, "Mechanism of Voltage Gating in Potassium Channels", *Science*, Volume 336, Number 6078, April 13, 2012, pp. 229–233.

[24] Server Kasap and Khaled Benkrid, "Parallel Processor Design and Implementation for Molecular Dynamics Simulations on a FPGA-Based Supercomputer", *Journal of Computers*, Volume 7, Number 6, June, 2012, pp. 1312–1328.

[25] Md. Ashfaquzzaman Khan, Matt Chiu and Martin C. Herbordt, "FPGA-Accelerated Molecular Dynamics", in Wim Vanderbauwhede and Khaled Benkrid (eds.), *High-Performance Computing Using FPGAs*, Springer Science+Business Media, LLC, 2013, pp. 105–135.

[26] Sameer Kumar, Yanhua Sun and Laxmikant V. Kalé, "Acceleration of an Asynchronous Message Driven Programming Paradigm on IBM Blue Gene/Q", *27th IEEE International Parallel and Distributed Processing Symposium*, Boston, MA, May 20–24, 2013, pp. 689–699.

[27] Jeffrey S. Kuskin, Cliff Young, J.P. Grossman, Brannon Batson, Martin M. Deneroff, Ron O. Dror and David E. Shaw, "Incorporating Flexibility in Anton, a Specialized Machine for Molecular Dynamics Simulation", *14th Annual International Symposium on High-Performance Computer Architecture*, Salt Lake City, UT, February 16–20, 2008, pp. 343–354.

[28] Carsten Kutzner, Rossen Apostolov, Berk Hess and Helmut Grubmüller, "Scaling of the GROMACS 4.6 molecular dynamics code on SuperMUC", *unpublished*, 2014.

[29] Richard H. Larson, John K. Salmon, Ron O. Dror, Martin M. Deneroff, Cliff Young, J.P. Grossman, Yibing Shan, John L. Klepeis and David E. Shaw, "High-Throughput Pairwise Point Interactions in Anton, a Specialized Machine for Molecular Dynamics Simulation", *14th Annual International Symposium on High-Performance Computer Architecture*, Salt Lake City, UT, February 16–20, 2008, pp. 331–342.

[30] Jing Li, Saher A. Shaikh, Giray Enkavi, Po-Chao Wen, Zhijian Huang and Emad Tajkhorshid, "Transient formation of water-conducting states in membrane transporters", *Proceedings of the National Academy of Sciences of the United States of America*, Volume 110, Number 19, May 7, 2013, pp. 7696–7701.

[31] Erik Lindahl, "Evolutions & Revolutions in Peta- and Exascale Biomolecular Simulation", *Conference on Scientific Computing*, Paphos, Cyprus, December 3–6, 2013.

[32] Kresten Lindorff-Larsen, Stefano Piana, Ron O. Dror and David E. Shaw, "How Fast-Folding Proteins Fold", *Science*, Volume 334, Number 6055, 2011, pp. 517–520.

[33] Siewert J. Marrink, H. Jelger Risselada, Serge Yefimov, D. Peter Tieleman and Alex H. de Vries, "The MARTINI Force Field: Coarse Grained Model for Biomolecular Simulations", *Journal of Physical Chemistry B*, Volume 111, Issue 27, July 12, 2007, pp. 7812–7824.

[34] Tetsu Narumi, Ryutaro Susukita, Takahiro Koishi, Kenji Yasuoka, Hideaki Furusawa, Atsushi Kawai and Toshikazu Ebisuzaki, "1.34 Tflops Molecular Dynamics Simulation for NaCl with a Special-Purpose Computer: MDM", *2000 ACM/IEEE Conference on Supercomputing*, Dallas, TX, November 5–10, 2000.

[35] Jared Ostmeyer, Sudha Chakrapani, Albert C. Pan, Eduardo Perozo and Benoît Roux, "Recovery from slow inactivation in K+ channels is controlled by water molecules", *Nature*, Volume 501, Issue 7465, September 5, 2013, pp. 121–124.

[36] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant V. Kalé and Klaus Schulten, "Scalable Molecular Dynamics with NAMD", *Journal of Computational Chemistry*, Volume 26, Issue 16, December, 2005, pp. 1781–1802.

[37] Stefano Piana, Kresten Lindorff-Larsen and David E. Shaw, "Protein folding kinetics and thermodynamics from atomistic simulation", *Proceedings of the National Academy of Sciences of the United States of America*, Volume 109, Number 44, October 30, 2012, pp. 17845–17850.

[38] Stefano Piana, Kresten Lindorff-Larsen and David E. Shaw, "Atomic-level description of ubiquitin folding", *Proceedings of the National Academy of Sciences of the United States of America*, Volume 110, Number 15, April 9, 2013, pp. 5915–5920.

[39] Steve Plimpton, "Fast Parallel Algorithms for Short-Range Molecular Dynamics", *Journal of Computational Physics*, Volume 117, Issue 1, March 1, 1995, pp. 1–19.

[40] Cristian Predescu, Ross A. Lippert, Adam K. Lerer, Brian Towles, J.P. Grossman, Robert M. Dirks and David E. Shaw, "The *u*-series: A separable, Gaussian-based decomposition for electrostatics computation", *unpublished*, 2014.

[41] Daniele P. Scarpazza, Douglas J. Ierardi, Adam K. Lerer, Kenneth M. Mackenzie, Albert C. Pan, Joseph A. Bank, Edmond Chow, Ron O. Dror, J.P. Grossman, Daniel Killebrew, Mark A. Moraes, Cristian Predescu, John K. Salmon and David E. Shaw, "Extending the generality of molecular dynamics simulations on a special-purpose machine", *27th IEEE International Parallel and Distributed Processing Symposium*, Boston, MA, May 20–24, 2013, pp. 933–945.

[42] Ronald Scrofano, Maya B. Gokhale, Frans Trouw and Viktor K. Prasanna, "Accelerating Molecular Dynamics Simulations with Reconfigurable Computers", *IEEE Transactions on Parallel and Distributed Systems*, Volume 19, Number 6, June, 2008, pp. 764–778.

[43] Yibing Shan, John L. Klepeis, Michael P. Eastwood, Ron O. Dror and David E. Shaw, "Gaussian split Ewald: A fast Ewald mesh method for molecular simulation", *Journal of Chemical Physics*, Volume 122, Number 5, February 1, 2005, pp. 054101:1–13.

[44] David E. Shaw, "A Fast, Scalable Method for the Parallel Evaluation of Distance-Limited Pairwise Particle Interactions", *Journal of Computational Chemistry*, Volume 26, Issue 13, October, 2005, pp. 1318–1328.

[45] David E. Shaw, Martin M. Deneroff, Ron O. Dror, Jeffrey S. Kuskin, Richard H. Larson, John K. Salmon, Cliff Young, Brannon Batson, Kevin J. Bowers, Jack C. Chao, Michael P. Eastwood, Joseph Gagliardo, J.P. Grossman, C. Richard Ho, Douglas J. Ierardi, István Kolossváry, John L. Klepeis, Timothy Layman, Christine McLeavey, Mark A. Moraes, Rolf Mueller, Edward C. Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles and Stanley C. Wang, "Anton, a Special-Purpose Machine for Molecular Dynamics Simulation", *34th Annual International Symposium on Computer Architecture*, San Diego, CA, June 9–13, 2007, pp. 1–12.

[46] David E. Shaw, Ron O. Dror, John K. Salmon, J.P. Grossman, Kenneth M. Mackenzie, Joseph A. Bank, Cliff Young, Martin M. Deneroff, Brannon Batson, Kevin J. Bowers, Edmond Chow, Michael P. Eastwood, Douglas J. Ierardi, John L. Klepeis, Jeffrey S. Kuskin, Richard H. Larson, Kresten Lindorff-Larsen, Paul Maragakis, Mark A. Moraes, Stefano Piana, Yibing Shan and Brian Towles, "Millisecond-Scale Molecular Dynamics Simulations on Anton", *2009 Conference on High Performance Computing, Networking, Storage and Analysis*, Portland, OR, November 14–20, 2009.

[47] David E. Shaw, Paul Maragakis, Kresten Lindorff-Larsen, Stefano Piana, Ron O. Dror, Michael P. Eastwood, Joseph A. Bank, John M. Jumper, John K. Salmon, Yibing Shan and Willy Wriggers, "Atomic-Level Characterization of the Structural Dynamics of Proteins", *Science*, Volume 330, Number 6002, October 15, 2010, pp. 341–346.

[48] Yanhua Sun, Gengbin Zheng, Chao Mei, Eric J. Bohm, James C. Phillips and Laxmikant V. Kalé, "Optimizing Fine-grained Communication in a Biomolecular Simulation Application on Cray XK6", *2012 International Conference for High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, UT, November 10–16, 2012.

[49] Johan Sund, Martin Andér and Johan Åqvist, “Principles of stop-codon reading on the ribosome”, *Nature*, Volume 465, Issue 7300, June 17, 2010, pp. 947–950.

[50] Makoto Taiji, Junichiro Makino, Akihiro Shimizu, Ryo Takada, Toshikazu Ebisuzaki and Daiichiro Sugimoto, “MD-GRAPE: a parallel special-purpose computer system for classical molecular dynamics simulations”, *6th Joint EPS-APS International Conference on Physics Computing*, Lugano, Switzerland, August 22–26, 1994, pp. 609–612.

[51] Makoto Taiji, Noriyuki Futatsugi, Tetsu Narumi, Atsushi Suenaga, Yousuke Ohno, Naoki Takada and Akihiko Konagaya, “Protein Explorer: A Petaflops Special-Purpose Computer System for Molecular Dynamics Simulations”, *2003 ACM/IEEE Conference on Supercomputing*, Phoenix, AZ, November 15–21, 2003.

[52] Brian Towles, J.P. Grossman, Brian L. Greskamp and David E. Shaw, “Unifying on-chip and inter-node switching within the Anton 2 network”, *2014 International Symposium on Computer Architecture*, Minneapolis, MN, June 14–18, 2014, pp. 1–12.

[53] Mike Wu and Ross Walker, “Amber 12 NVIDIA GPU acceleration support”, http://ambermd.org/gpus/benchmarks.htm, retrieved April 2, 2014.

[54] Cliff Young, Joseph A. Bank, Ron O. Dror, J.P. Grossman, John K. Salmon and David E. Shaw, “A 32×32×32, spatially distributed 3D FFT in four microseconds on Anton”, *2009 Conference on High Performance Computing, Networking, Storage and Analysis*, Portland, OR, November 14–20, 2009.

[55] Gongpu Zhao, Juan R. Perilla, Ernest L. Yufenyuy, Xin Meng, Bo Chen, Jiying Ning, Jinwoo Ahn, Angela M. Gronenborn, Klaus Schulten, Christopher Aiken and Peijun Zhang, “Mature HIV-1 capsid structure by cryo-electron microscopy and all-atom molecular dynamics”, *Nature*, Volume 497, Issue 7451, May 30, 2013, pp. 643–646.