2013 IEEE 27th International Symposium on Parallel & Distributed Processing

Extending the generality of molecular dynamics simulations on a special-purpose machine

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, David E. Shaw†

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

***Abstract*—Special-purpose computing hardware can provide significantly better performance and power efficiency for certain applications than general-purpose processors. Even within a single application area, however, a special-purpose machine can be far more valuable if it is capable of efficiently supporting a number of different computational methods that, taken together, expand the machine's functionality and range of applicability. We have previously described a massively parallel special-purpose supercomputer, called Anton, and have shown that it executes traditional molecular dynamics simulations orders of magnitude faster than the previous state of the art. Here, we describe how we extended Anton's software to support a more diverse set of methods, allowing scientists to simulate a broader class of biological phenomena at extremely high speeds. Key elements of our approach, which exploits Anton's tightly integrated hardwired pipelines and programmable cores, are applicable to the hardware and software design of various other specialized or heterogeneous parallel computing platforms.**

I. INTRODUCTION

Specialization of computing hardware to a particular application can be a powerful approach for accelerating certain computations on both high-performance and embedded systems [1–15]. Many application areas, however, benefit from the utilization of a variety of computational methods, and supporting diverse methods efficiently in a specialized hardware system has historically proven challenging.

We have previously described a massively parallel special-purpose machine, called Anton, and have shown that it performs traditional molecular dynamics (MD) simulations of biomolecular systems nearly two orders of magnitude faster than was previously possible on any existing hardware platform[1] [17, 18]. An Anton machine performs the entire MD computation on a large number of identical application-specific integrated circuits (ASICs), each of which includes hardwired pipelines for fast particle–particle and particle–gridpoint interactions [19], programmable cores with instructions tailored to MD simulations [20], fast on-chip static memory, and a specialized interconnect that allows both the hardwired and programmable components to send short messages efficiently [21].

Our previous publications have focused on the implementation and performance of traditional, "plain vanilla" MD simulations on Anton. In this paper, we discuss the software implementation of a much more diverse set of methods on Anton, substantially expanding Anton's functionality and range of applicability. Despite the fact that Anton is a specialized machine, we were able to accommodate the great majority of these methods while maintaining simulation performance orders of magnitude greater than that achievable on other hardware platforms. In particular, most such methods achieve performance exceeding 90% of Anton's peak benchmark performance.

The diverse methods implemented on Anton, which involve a wide variety of computational routines and data structures, fall into several categories:

- Techniques for incorporating various physical interactions and effects within the physical model, referred to as a force field, that is used in an MD simulation to calculate the forces acting on all atoms in the simulated biological system. Examples include methods for supporting applied electric fields, electrostatic polarizability, cross terms, "virtual" atomic sites, free energy perturbation calculations, and the special-case treatment of forces between certain pairs of atoms that are not covalently bonded to each other.
- More complex integration methods, which are used to calculate where atoms move in response to these forces. Such methods include several schemes for controlling the simulated pressure and temperature of the molecular system being simulated.
- Enhanced sampling techniques employed to facilitate the efficient exploration of the set of distinct three-dimensional configurations that can be assumed by a given biomolecule, and to efficiently capture important but infrequent structural changes

[1] In addition to providing much higher performance than general-purpose supercomputers and commodity clusters, Anton consumes dramatically less power: a 512-node Anton machine consumes 52 kW, which includes the power required for the built-in cooling units. While low power was not a primary goal in the design, it is a valuable side effect of hardware specialization that is likely to become increasingly important as supercomputing designs hit the "power wall" [16].

* Edmond Chow is currently with the School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA, 30332.
† David E. Shaw is also with the Center for Computational Biology and Bioinformatics, Columbia University, New York, NY 10032. E-mail correspondence: David.Shaw@DEShawResearch.com.

1530-2075/13 $26.00 © 2013 IEEE
DOI 10.1109/IPDPS.2013.93

in biomolecular systems. These methods include temperature-accelerated molecular dynamics, simulated tempering, and the application of several types of non-physical biasing forces.

Anton simulations employing various combinations of these methods have helped characterize processes including the binding of drug molecules to their protein targets [22–24], the folding of proteins into their characteristic three-dimensional structures [25, 26], and key structural changes underlying the function of important classes of protein molecules, including ion channels [27, 28], kinases [29], and G protein–coupled receptors [30, 31].

As for other massively parallel specialized or heterogeneous machines, the main challenge of mapping additional functionality to Anton reflects Amdahl's law: calculations that are not accelerated using Anton's hardware may become a performance bottleneck even if they represent only a small fraction of total runtime on more conventional machines. Several features of Anton's hardware and software architecture proved critical in addressing this challenge. Our software implementation of various methods heavily exploited the fact that Anton provides general-purpose compute capacity that is tightly integrated with the more specialized components of each ASIC: Anton's programmable cores can manipulate the inputs and outputs of its hardwired pipelines at very fine granularity (i.e., at the word level), and communication operations can be performed with very low latency between computational subunits within an ASIC or on different ASICs. We also architected our software to allow users to specify different execution frequencies for each of a wide variety of operations involved in these methods, providing opportunities to reduce performance penalties with little or no loss of accuracy. While Anton is specialized for molecular dynamics, a similar combination of hardware and software design strategies may be useful in accommodating a diverse set of methods on machines designed for other applications.

II. Background

This section provides a brief introduction to traditional MD simulation and to Anton's hardware architecture, with an emphasis on details relevant to the implementation of the extended MD methods that are described in the rest of this paper. More details about Anton's architecture are available in our previous work [17–21].

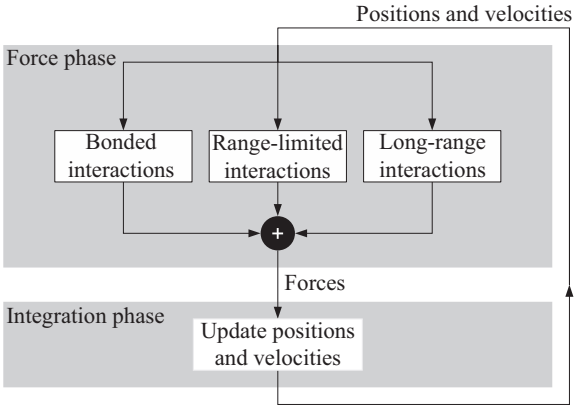

Figure 1. Data flow diagram of a basic MD iteration.

A molecular dynamics simulation consists of a sequence of iterations, each covering a time step of fixed duration, for example of length 2 fs. In its most basic form, each "MD iteration" consists of the calculation of the forces acting on all atoms, followed by an integration phase that updates the positions and momenta of the atoms according to the classical laws of motion, as depicted in Figure 1. The forces are calculated as a function of particle positions using a physical model known as a force field, which traditionally includes bonded, van der Waals, and electrostatic forces. Bonded forces involve interactions between small groups of atoms connected by one or more covalent bonds, whereas van der Waals and electrostatic forces (non-bonded forces), involve all pairs of atoms in the system. On Anton, non-bonded forces are expressed as the sum of *range-limited interactions* and *long-range interactions*. Range-limited interactions, which consist of van der Waals interactions and a contribution from electrostatic interactions, decay quickly with distance and are computed explicitly between all pairs of atoms separated by less than a specified cutoff radius. Long-range interactions, which comprise the remaining electrostatic contributions, decay slowly but smoothly; they are efficiently computed on Anton using a grid-based method [32].

Anton is a special-purpose machine designed to accelerate the MD computations described above. An Anton machine consists of nodes connected in a toroidal topology; for example, a 512-node configuration uses an $8 \times 8 \times 8$ topology, corresponding to an $8 \times 8 \times 8$ spatial partitioning of the biological system with periodic boundary conditions. The partitioning maps each atom to a "home node," which is responsible for its force accumulation and integration.

Each Anton node includes an application-specific integrated circuit (ASIC) containing two major computational subsystems: the high-throughput interaction subsystem (HTIS) [18] and the flexible subsystem [19]. Figure 2 summarizes Anton's hardware organization in three views: a machine-level view, a node-level view, and an overview of the flexible subsystem. The HTIS contains parallel hardwired pipelines called pairwise point interaction pipelines (PPIPs) and is controlled by a specialized programmable core called the interaction control block processor (ICB). The flexible subsystem contains four general-purpose programmable cores, which are responsible for the overall data flow of the MD computation, and eight programmable geometry cores (GCs) with a specialized instruction set designed to accelerate MD computations. The flexible subsystem also contains four asynchronous data transfer engines and a correction pipeline (CP)—a single instance of the pipeline contained in the HTIS—which selectively reverses a few of the interactions computed by the HTIS. Finally, the ASIC also contains DRAM controllers, an intra-chip ring network, communication channels to neighboring ASICs, and a host interface.

The node's hardware units, both hardwired and programmable, are tightly integrated so that they can operate

efficiently on fine-grained data. This tight integration is made possible by communication primitives for fine-grained, low-latency data transfer [21], and smart DRAM controllers that are accessible to both programmable and hardwired units and are equipped with independent arithmetic units (e.g., for force, charge, and potential accumulation) and sizeable caches.

Anton has a highly customized memory hierarchy of caches and software-managed scratchpad memories without hardware cache coherence across cores. Cache memories are sized to contain the code and data associated with an MD simulation, and are smaller than those typically found in commodity systems; for example, each GC has a 4K-instruction I-cache and a 16 KB data cache. Cores implement simple branch predictors (general-purpose cores) or no branch predictors at all (GCs). The simplicity of branch prediction, absence of cache coherence, and small size of cache contribute to significant area and power savings, but create additional software design challenges.

III. Hardware and Software Design Strategies to Support Extensibility

Several aspects of Anton's hardware and software architecture allowed us to implement the broad range of methods discussed in this paper with only a modest slowdown, despite the high degree of hardware specialization in Anton's design.

First, many of these implemented methods take advantage of Anton's fine-grained, low-latency communication mechanisms. In a plain vanilla simulation, the general-purpose cores of the flexible subsystem primarily calculate bonded forces and perform integration. These calculations require massive gathers and scatters of position and force data. Anton's communication networks were designed to facilitate these operations by supporting primitives that act at a very fine granularity (a single force or position), both within a node and across the entire machine. Bonded force computation and integration have often served as the paradigms for the implementation of the enhancements discussed later in this paper, but even for methods that do not strictly follow this communication pattern, like the conformation restraints in Section V-A, Anton's support for fine-grained, low-latency communication allows us to distribute and balance computations effectively across flexible resources throughout the entire machine.

Second, Anton's software architecture allows users to specify different application frequencies for each of a wide variety of force calculations and integration operators. Many of the extended methods described in this paper can be applied infrequently with no significant reduction in overall simulation accuracy, and such infrequent application substantially decreases their impact on performance. Because each simulation uses a different combination of methods, and because the optimal frequencies for applying a given method may vary from one simulation to the next, each simulation requires a different schedule for the execution of various operations. To minimize code complexity without sacrificing efficiency, we use a software "sequencer" abstraction to implement these schedules. The sequencer is analogous to the hardware sequencer in micro-programmed CPUs [33], but generalized for parallel software on a heterogeneous machine with many programmable cores and hardwired pipelines. In our implementation, each general-purpose core runs an independent replica of the same software sequencer, and issues commands to a subset of the GC cores and hardwired pipelines. Anton's fine-grained, low-latency communication between the general-purpose cores and hardwired pipelines supports an efficient implementation of this scheduler. (The sequencer will be discussed further in Section IV.)

Finally, to support some classes of methods, Anton's hardware pipelines were designed with a limited degree of programmability. For example, the PPIP functions include a tabulated piecewise polynomial subexpression that admits a variety of functional forms. Moreover, the PPIP's parameters and tables can be efficiently reloaded on the fly at a time-

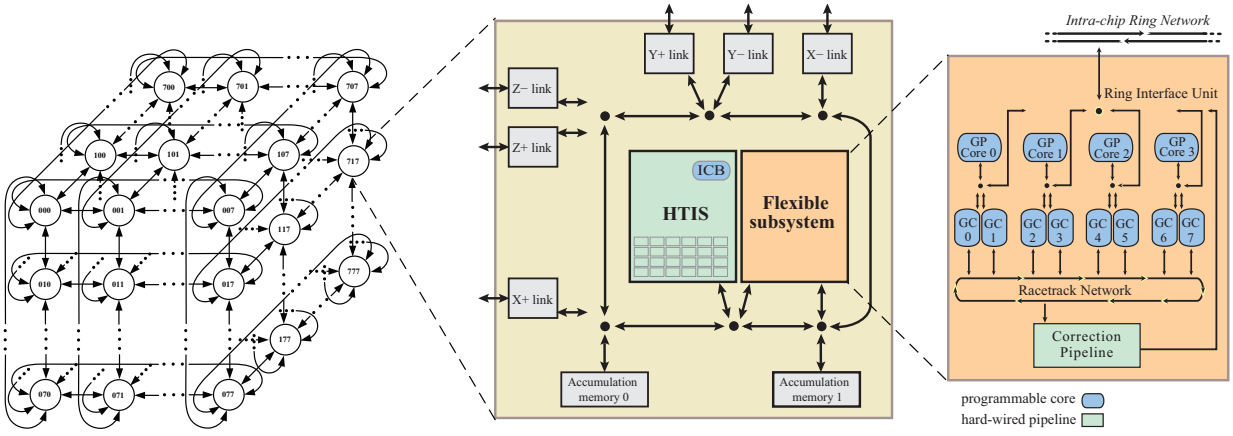

Figure 2. Hardware architecture of Anton at the machine level, node-level, and flexible-subsystem level.

step granularity. Thus the HTIS may be quickly switched from computing forces to computing potential energy or pressure, and these alternative quantities can be computed in the same amount of time as a comparable force calculation. The frequent computation of such quantities, like energy and pressure, is crucial to the implementation of several important integrators and enhanced sampling methods discussed below.

In the following sections we describe the implementation of several of these more general MD methods on Anton. Figure 3 illustrates how these methods fit into the traditional data flow introduced in Figure 1 and how they interact with each other. In this diagram, each new method is represented as a block. The computations associated with the blocks may take place in parallel on multiple programmable and/or hardwired units. The arrows in the figure denote data dependencies between computational blocks, which impose temporal constraints between communicating blocks. Data transfers implicitly enforce these dependencies through the counted remote write mechanism described in previous publications [21].

IV. Temperature- and Pressure-controlled Simulations

The basic integration phase described in the Background section simulates a biological system in which the number of particles (N), the volume (V), and the total energy of the system (E) are constant. In MD parlance, such a basic MD simulation samples the "NVE ensemble." Yet most biological processes occur in an environment where temperature and pressure are fixed. As a consequence, users frequently wish to maintain constant temperature ("NVT ensemble"), or both temperature and pressure ("NPT ensemble").

The algorithms that keep the temperature and the pressure constant in a simulation are called "thermostats" and "barostats," respectively. Thermostats operate by coupling the atoms in the simulation with a virtual heat bath, while barostats operate by adjusting the volume and the aspect ratio of the simulation box in response to pressure fluctuations.

The thermostat selection offered by Anton includes Nosé-Hoover chains and the Berendsen, Andersen, and Langevin thermostats [34]. Some of these thermostats operate in a closed loop and require a measurement of the current system's total kinetic energy (which is directly related to the system's temperature) as an input at every invocation. Others operate in an open loop and use a source of randomness to model the influence of the heat bath; for example, the Langevin thermostat adds a small random noise to the velocities of atoms and imposes a frictional force on each atom directly proportional to its velocity. The initial design of the parallel random number generator presented in [35] was motivated by these applications.

Barostats operate in a closed loop, taking as input the instantaneous pressure, which is computed and globally reduced across the machine at every barostat invocation. Computing the pressure requires the calculation of the virial, labeled "virial phase" in Figure 3. Anton implements the Berendsen barostat [34] and the Martyna, Tuckerman and Klein (MTK) barostat [36].

Thermostats and barostats, as well as the integration phase from the basic MD iteration, can be construed formally as "state operators," that is, functions $F: (\vec{x}, \vec{p}, \vec{\gamma}) \mapsto (\vec{x}', \vec{p}', \vec{\gamma}')$ that map positions $\vec{x}$, momenta $\vec{p}$, and extended variables $\vec{\gamma}$ (e.g., internal thermostat or barostat state) to new values. The formalism of state operators has the useful property that the composition of state operators is itself a state operator; we use this property to describe an entire simulation as a nested composition of primitive state operators. State operators can be composed with arbitrary intervals in theoretically rigorous ways to maintain desirable integration properties such as symplecticity and time reversibility. Furthermore, if operators for infrequent temperature or pressure control are constructed correctly, they provably converge to the same distribution produced by frequent temperature or pressure control in the limit of long timescales. An extensive treatment of this topic will appear in a separate work [37]. The main advantage of this approach is that users can apply operators infrequently, thus amortizing the cost of their computation over several basic MD iterations, without degrading

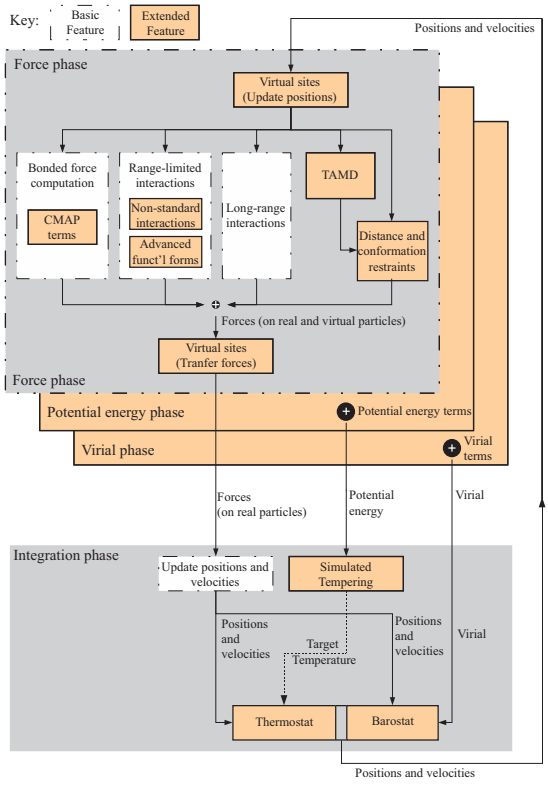

Figure 3. Data flow diagram of an MD iteration that includes the extended methods presented in Sections IV, V, and VI. The new data flow conserves a division of methods between the force and integration phases, as in the basic MD data flow illustrated in Figure 1. Two new phases are present to compute the potential energy and the virial. The new phases are composed of the same feature blocks as in the force phase, but blocks are executed in special modes, leading them to compute potential energy or virial terms instead of force terms.

accuracy.

To implement such schedules, the software sequencer running on each general-purpose core iterates over an "MD program." An MD program is a predetermined sequence of "MD instructions," where each MD instruction corresponds to one iteration of the simulation, and specifies the set of blocks (in Figure 3) enabled during that iteration. The pressure-controlled simulation includes an MD instruction that enables the virial computation phase and the barostat block, but not the integration block. The corresponding step in the MD program computes the system's pressure and adjusts the simulation box's dimensions, but does not integrate atom positions.

To hide the latency of inter-unit command communication, we perform software pipelining on the sequencer loop running on the general-purpose cores, which results in computation blocks being pipelined across the entire machine. All the sequencers proceed independently, issuing multiple commands for each MD instruction; as a result, several commands may be outstanding at any time. Correctness does not require additional synchronization beyond that which is implicit in the data flow; for example, the force phase begins on the arrival of new atom positions, and integration begins when all force components (which are counted by the DRAM controllers) have been accumulated. Thus, the sequencer abstraction provides an efficient way to schedule computation at a fine grain without the need for synchronizing control state explicitly.

Finally, to maintain performance comparable to traditional MD simulation on Anton, we ensure that the code and data used by simulation capabilities that are applied frequently fit in on-chip cache or SRAM. Anton's general-purpose and geometry cores have instruction and data caches, but they are single-level and backed by DRAM, with typical DRAM-latency penalties: a fill from DRAM costs about 100 ns, which can cause a 1% slowdown in a basic MD iteration. To avoid these penalties, we explicitly control the layout of our object code and data, so that all frequently used code and data are placed in a single cache image that can be accessed at run time without cache conflicts. To accomplish this, we rebuild our MD software for the specific needs of each user simulation, allowing the compiler to eliminate blocks of code that are not needed by that simulation, and the linker to achieve an optimized layout. In addition to reducing footprint (and thus cache misses), this "specialized build" produces code that benefits from reduced branching (cores have limited or no branch prediction) and call-stack depth (a shallow call stack makes more efficient use of the register window–based stack cache of the general-purpose cores). As a result, a given simulation only pays a performance penalty (in terms of cache usage and branching) for the methods it employs. On the GCs, even a specialized build exceeds the size of the instruction cache, so the code is further organized into overlays by capability and frequency of use, so that cache misses are incurred only for the most infrequently applied methods.

V. Enhanced Sampling

While Anton provides the ability to simulate biological processes that occur over millisecond timescales, many processes occur over timescales that are orders of magnitude larger and are currently inaccessible to traditional MD simulations. Enhanced sampling refers to a group of methods that attempt to bridge this timescale gap [38]. When used carefully, these methods can often provide a qualitative, and sometimes quantitative, view of long-timescale events using a fraction of the simulation time required to observe the events directly with traditional MD. Most enhanced sampling methods involve introducing biasing forces and other modifications to traditional MD to increase the likelihood of observing events of interest. In the next two sections, we will discuss the modifications to traditional MD that have been implemented on Anton and some of the enhanced sampling methods that are made possible by these new features.

A. Restraints

Distance and conformation restraints are examples of techniques commonly used in enhanced sampling. A distance restraint is an interaction between the centers of mass of two groups of particles. Such a restraint allows users to bias a simulation by keeping two molecules—a drug molecule and a protein receptor, for example—or two portions of molecules, at a particular distance from one another. A conformation restraint generates forces that tend to hold a group of particles in a desired conformation. This allows a user to bias a simulation by holding part or all of a molecule—the binding pocket of a protein, for example—in a particular conformation. Distance and conformation restraints play a role in many of the MD simulations that scientists run on a day-to-day basis, not only in the context of enhanced sampling simulations.

The "target value" for a restraint—that is, the desired distance between two molecules, or the desired distance of a molecule from a model conformation—is often a constant value, particularly in enhanced sampling methods like umbrella sampling [38]. Alternatively, target values and force constants can vary over the course of a simulation allowing for steered molecular dynamics (SMD) [40, 41] and temperature-accelerated molecular dynamics (TAMD) [42]. In SMD, target values change according to a predefined schedule, whereas in TAMD they change dynamically in response to fluctuations of the system and a high-temperature external Brownian thermostat.

We now present the algorithm and implementation of conformation restraints as an example of how enhanced sampling methods can map to Anton. Forces in a conformation restraint cause a selected molecule to assume a conformation that is close to a desired "model" that the user specifies. Not only does the user specify the model, but also the desired distance between the actual and the model conformations, which we call the target value of the restraint. Distances between the actual and model conformations are measured as a root-mean-square deviation (RMSD) of the restrained atoms relative to the model. More formally, if $\vec{x}_i$

and $\vec{y}_i$ are the positions of the i^{th} atom in the actual and model conformation, respectively, then:

$$RMSD = \min_{\theta} \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|\vec{x}_i - \theta(\vec{y}_i)\|^2}$$

where θ is a rigid-body transformation (rotations and translations) applied to the model, and N is the number of atoms restrained. By construction, the RMSD is invariant to rotations or translations of the model or of the actual conformations.

We now describe the algorithm used to compute conformation restraint forces. We represent the actual conformation $\boldsymbol{X}$ and the model conformation $\boldsymbol{Y}$ as two $N \times 3$ matrices, each containing all the $\vec{x}_i$ as rows and all the $\vec{y}_i$ as rows, respectively. The algorithm proceeds as follows:

1. For each conformation $\boldsymbol{X}$ and $\boldsymbol{Y}$, center it at the origin by subtracting its center of mass from each atom's coordinates, thus obtaining $\boldsymbol{X'}$ and $\boldsymbol{Y'}$, respectively;
2. Compute the covariance matrix $\boldsymbol{R} = \boldsymbol{X'^T Y'}$;
3. Compute matrix $\boldsymbol{U}$, the rotation matrix that aligns $\boldsymbol{X'}$ and $\boldsymbol{Y'}$ optimally, using Kabsch's algorithm [42, 43];
4. Compute the RMSD distance

$$d = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|\vec{x}_i{}' - \boldsymbol{U}\vec{y}_i{}'\|^2}$$

5. Compute the restraint force $\overrightarrow{F_i} = -\frac{k\,(d - d_{eq})\,(\vec{x}_i{}' - \boldsymbol{U}\vec{y}_i{}')}{N\,d}$ on each atom $i \in \{1,2,\ldots,N\}$, where k denotes the spring constant and d_{eq} denotes the target distance.

Now we present our parallel implementation. It statically assigns each restrained atom i to a distinct "worker core" W_i. This assignment has two benefits: first, it achieves a better load balancing than the "natural" assignment of each restrained atom to its home core; in fact, atoms involved in a conformation restraint tend to be spatially close to each other, and thus clustered on a small number of home cores. The natural assignment will distribute all the load of a conformation restraint to the few home cores interested, and leave all other cores idle. Second, model positions for each atom can reside permanently on each worker rather than having to migrate with each atom. At each application of the restraint, each core where a restrained atom i resides (i.e., its home core) transfers the atom's coordinates to worker core W_i. Each worker W_i receives the coordinates of atom i and cooperates with the other workers, performing both computations specific to atom i and computations associated to the entire restraint. At the end, each worker returns forces for its atom to the atom's home for accumulation. The specific parallelization proceeds as follows:

1. Each restrained atom's home core sends atom position $\vec{x}_i$ to core W_i, which owns the model coordinates $\vec{y}_i{}'$
2. W_i computes atom i's contribution to covariance matrix $\boldsymbol{R}$, which is $\boldsymbol{r_i} = \vec{x}_i{}^T \vec{y}_i{}'$
3. Workers perform one global reduction that yields $(\boldsymbol{R}\,; \langle\vec{x}\rangle) = reduce\left(\boldsymbol{r_i}\,; \frac{1}{N}\vec{x}_i\right)$
4. Workers compute their centered atoms' coordinates $\vec{x}_i{}' = \vec{x}_i - \langle\vec{x}\rangle$
5. Workers perform identical computations of $\boldsymbol{U}$, by applying Kabsch's algorithm to $\boldsymbol{R}$
6. Each worker W_i computes atom i's contribution to the mean-squared deviation

$$m_i = \frac{1}{N}\|\vec{x}_i{}' - \boldsymbol{U}\vec{y}_i{}'\|^2$$

7. All workers perform a global reduction $m = reduce(m_i)$ and compute the RMSD as $d = \sqrt{m}$;
8. Each worker calculates the restraint force $\boldsymbol{F_i}$ and sends it to atom i's home core for accumulation.

The parallelized algorithm takes advantage of an optimization to combine the global reduction of $\boldsymbol{R}$ and $\langle\vec{x}\rangle$ into a single step (Step 3 in the parallel algorithm). First, we compute $\boldsymbol{Y'}$ offline; then, thanks to the mathematical property $\boldsymbol{R} = \boldsymbol{X'^T Y'} = \boldsymbol{X^T Y'}$, we compute $\boldsymbol{r_i}$ using the uncentered conformation $\boldsymbol{X}$ without the need for $\langle\vec{x}\rangle$. We can thus delay the computation of $\langle\vec{x}\rangle$, and compute it, together with $\boldsymbol{R}$, in one global reduction rather than two, which reduces latency. Step 5 in the parallel algorithm is replicated across cores: all cores compute the same value at the same time; this choice leads to lower latencies than computing $\boldsymbol{U}$ on a single core and broadcasting the result.

The parallel portion of the computation of conformation restraints benefits from the tight integration among cores made possible by the low-latency interconnect and by the smart DRAM controllers on Anton, especially to accelerate the coordinate distribution of Step 1, the force accumulation of Step 8 and the global reductions in Steps 3 and 7. Some additional challenges are involved in optimizing the serial portion of the computation, primarily Step 5, whose relative latency after parallelization is significant. To mitigate its performance impact, we adopt a selection of the strategies presented before in Section III. We allow users to apply conformation restraints infrequently by packaging them in the form of an extended state operator. In addition to that, Kabsch's algorithm of Step 5 is amenable to an optimization that preconditions its arguments and postconditions its results using the values of arguments and results of the previous iteration, thus rendering its cost even lower. To make the general-purpose cores most able to participate at a fine grain in the MD computation by accumulating forces on individual

atoms, we used mixed arithmetic, performing only Steps 5 and 6 in floating point, and all other steps in the same fixed-point format used to represent and process forces and coordinates across Anton.

Thanks to these optimizations, Anton supports conformation restraints at a small performance cost, as reported in Section VII. In addition, Anton supports variants of conformation restraints that use different atom weights, multiple models, time-varying parameters and composite flat-bottom harmonic potentials.

B. Simulated Tempering

The "conformation space" of a system is the set of all the spatial conformations that the system's molecules can assume. Simulated tempering attempts to speed up the sampling of conformation space by periodically modifying the temperature of the system [45]. At lower temperatures, molecules may not have sufficient thermal energy to overcome barriers in their energy landscape frequently, while at higher temperatures they overcome these barriers more frequently, resulting in better sampling. By periodically raising and lowering the temperature along a discrete "ladder of temperatures," simulated tempering facilitates conformational exploration.

Transitions occur only between target temperatures that are consecutive along the ladder, up or down one rung, in a Monte Carlo process. At each interval, the system either transitions or not according to the following Metropolis acceptance criterion:

$$A(\varphi, T_a \to T_b) = \min\left\{ e^{-\frac{U(\varphi)}{kT_a}+\frac{U(\varphi)}{kT_b}+w_a-w_b}, 1 \right\},$$

where A is the probability of accepting a transition from temperature T_a to temperature T_b (where T_b is selected with equal probability to be either one rung above or below T_a), given the state of the system φ. $U(\varphi)$ is the energy of state φ, k is Boltzmann's constant, T_i is the temperature of ladder rung i, and w_i is a weight chosen for T_i to promote equal sampling time at each temperature. Note that the system's potential energy must be computed before each putative transition to calculate the acceptance probability. Figure 4 shows temperature changes produced by simulated tempering in a short simulation of dialanine on Anton.

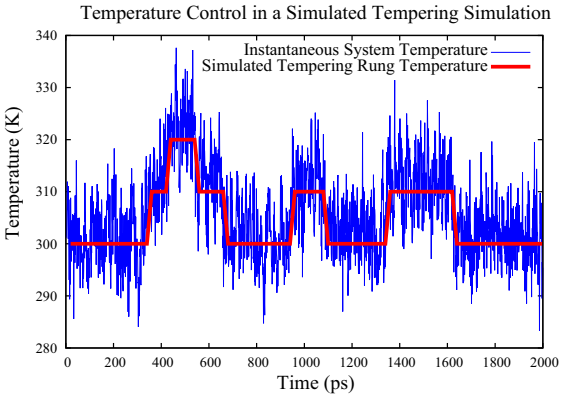

Figure 4. Temperature of dialanine in a simulated tempering simulation with ladder rungs at 300 K, 310 K, and 320 K. The red line is the thermostat temperature imposed by simulated tempering; the blue line is the instantaneous system temperature.

On commodity hardware, a related enhanced sampling algorithm, replica exchange, is often preferred over simulated tempering [46]. Replica exchange performs concurrent simulations of multiple "replicas" of the system, running at different temperatures, and swaps configurations between replicas in a Monte Carlo process. Replica exchange has the advantage that it allows for parallelization over the replicas being simulated, and it guarantees equal simulation time at each temperature without having to choose weights.

Anton's architecture, however, favors simulated tempering over replica exchange. Anton's limited SRAM size per node puts a lower bound on the number of ASICs required to simulate a single replica efficiently, so the parallelizability of replica exchange is limited by the number of ASICs available. On the other hand, Anton's architecture allows for efficient use of many ASICs in parallel for a single simulation. Furthermore, replica exchange imposes additional input/output costs on Anton, either to communicate between machines running parallel replicas or to swap state from DRAM between time-multiplexed replicas running on the same machine. These costs are considerable, since an exchange interval of a few picoseconds corresponds to only tens of milliseconds of wall-clock time on Anton. As for the preliminary simulations required to calculate weights for simulated tempering, they can be run very quickly on Anton, as they typically require hundreds of nanoseconds of simulation time.

An Anton simulation that employs simulated tempering typically attempts a Monte Carlo exchange every 1,000 to 10,000 time steps. This procedure consists of a potential energy computation, a transition attempt, and—if successful—a modification to the thermostat temperature.[2]

The data flow of a potential energy computation on Anton is very similar to that of a force computation, but each of the force computation kernels—bonded, range-limited, long-range, etc.—is replaced by a related but distinct potential energy computation kernel, as represented in Figure 3. The force polynomial tables in the HTIS and CP must therefore be swapped for energy tables, and modified bond calculations must be performed to compute energies instead of forces. All of these changes are controlled by the sequencers running on general-purpose cores, which send commands to the GCs, HTIS, and CP, indicating that an energy computation is to be performed.

Since potential energy computation is performed so infrequently, the energy computation code is not included in

[2] Anton also supports generalized simulated tempering, in which components of the Hamiltonian are modified. In that case, the simulated tempering algorithm must perform two potential energy calculations (one for each Hamiltonian), and must modify the Hamiltonian in addition to thermostat state. This two-energy computation program is managed by the sequencer, with no changes required to the code on the GC or ICB units.

the general-purpose cores' specialized build (mentioned in Section III) but instead lives in a separate code section in DRAM. The normal force computation thus pays no branch or instruction cache penalty for potential energy computation, even when simulated tempering is enabled.

Simulated tempering transition attempts and state updates are executed on each node in parallel, with no need for communication between nodes. On each node, only the core responsible for the thermostat executes the update routine and stores the simulated tempering data and state, saving space on other cores.

VI. Force field support and new force terms

Biomolecular force fields are continuously improved by the computational chemistry research community, and Anton's software has been extended to support many force field features needed by the researchers using Anton. These features include virtual sites, which have been introduced in more accurate models of electrostatics; uniform electric fields used, for example, to model transmembrane potentials; Drude particles that model polarization; and soft-core potentials, employed in certain free-energy perturbation techniques that compute differences in free energy between two states of a system (for example, ligand-binding free energies). Our strategies for handling these features are diverse. Here we discuss in detail two of these features, nonstandard interactions and CMAP terms, to illustrate some of the techniques we used to implement such diverse features on a specialized platform.

One example involves adding support for nonstandard van der Waals interactions. Force fields generally specify the energy terms corresponding to the interaction between a pair of non-bonded particles in diverse forms that can all be represented by $E = a \cdot f(k \cdot r^2)$, where f is a suitable function, r is the distance between the two particles, and a and k are each computed as either the arithmetic or the geometric mean of parameters associated with the two particles. This is the functional form hardwired into the pipelines of Anton's HTIS. Buckingham's form of the van der Waals interaction [47] does not directly fit this model, although it does decompose as a sum of terms that are supported directly by the HTIS. In this case, we accelerate evaluation of the force term by executing two passes through the HTIS—essentially running two force phases back-to-back. In other cases, a force field may override the values of a and k for specific pairs of atom types (e.g., selected ion–protein interactions). When there are few of these exceptional cases, we make use of the CP to correct the calculation. First, the CP repeats the same computation that takes place on the HTIS with the sign reversed. This effectively removes the undesired force contribution produced by the HTIS, which has treated the interaction as if it were standard. Then, we compute the forces due to the nonstandard interaction by loading into the CP the variant a and k parameters.

A second example involves CMAP terms, which appear in the CHARMM family of force fields, starting with variant CHARMM22/CMAP. CMAP is a cross term that adds a correction for each pair of neighboring φ- and ψ-dihedral angles along a protein's backbone. The energy of each term is computed by first calculating the angles (φ, ψ), and then interpolating the energy from one of four two-dimensional tables (each containing 24 × 24 discrete points) by bicubic interpolation. There are generally few instances of these terms in a simulation—roughly one for each amino acid in a protein.

CMAP terms can be viewed as a new kind of bond term. Yet, in contrast to other bond terms, CMAP terms are relatively complex, both in computation and in code and data size. We store precomputed CMAP interpolation functions in DRAM. The GC data caches, though, can only accommodate a small window of that table, centered around the most recently used entry of each term; since the changes to these dihedrals are generally small, table lookups are almost always serviced from the cached windows. After each force or energy phase, a GC may explicitly request updates to these windows; these requests are queued and handled by a general-purpose core, off the critical path.

As noted earlier, we often assign specialized roles to various cores. In general, we do this either to parallelize a task on a small scale, or to reduce miss rates in the core's instruction or data caches. Since the code for CMAP terms is relatively large—more than a quarter of the available instruction cache—our implementation uses core specialization for the latter purpose. Anton's bond-program generator, which assigns bond terms to GCs and balances their load across the machine, is modified to reserve a set of GCs for CMAP computation. CMAPs are assigned only to these cores. We also compile the CMAP-only code and non-CMAP code in separate segments of the GCs' code image; we deliberately align these segments so that they occupy overlapping regions in cache, forming overlays. The result is that each GC either has a CMAP-only role or a non-CMAP role during the calculation of bond terms. Either way, the footprint of the code run by each GC during an MD iteration will not exceed the capacity of core's instruction cache.

VII. Performance

In this section, we measure Anton's performance on the extended set of methods presented in the previous sections, using six representative biological systems, listed at the top of Table 1, which range in size from 13,543 to 229,983 atoms.

First, we present Anton's baseline performance using traditional, plain vanilla MD (i.e., without the use of extended MD techniques), measured on a representative set of biological systems under conditions identical to those used in our previously reported benchmarks [18]: a time step duration of 2.5 fs and a RESPA integrator that computes long-range electrostatics every 5 fs with the k-space Gaussian split Ewald (GSE) method [32]. Performance values are expressed in microseconds of simulated biological time per day of wall-clock time.

Then, we measure the performance with each extended method described in this paper enabled individually, and report the slowdown relative to the baseline. The results are in Table 1.

Among the biological systems considered, DHFR and ApoA1 are widely used benchmark systems; we have previ-

ously reported baseline performance for these systems on both Anton and commodity hardware [18, 48]. BPTI was the subject of Anton's first millisecond-scale simulation [18]. DHFR and BPTI consist of proteins and ions solvated in water; ApoA1 also includes a lipid membrane. The other three systems (GPCR, K^+ channel 1, and K^+ channel 2) have been studied on Anton using methods presented in this paper. The GPCR system was used to study the escape pathway of a drug from its binding site in a G protein–coupled receptor [24], represented in Figure 5. The two K^+ channel systems were used to model the behavior of a voltage-gated ion channel (represented in Figure 6), a protein embedded in a cell's membrane that regulates the flow of ions in and out of the cell in response to changes in the transmembrane voltage [27]. The GPCR and K^+ channel simulations contain proteins embedded in a lipid membrane, as well as ions, water, and—in the case of GPCR—a drug molecule.

In order to obtain baseline performance measurements for the BPTI, GPCR, and K^+ channel systems, we disabled all extended methods utilized in the published simulations. We performed all simulations on 512-node Anton machines, except for K^+ channel 2, which we simulated on a 1024-node machine. In order to obtain stable performance measurements, we simulated each system for 100 ns of biological time.

We chose for each method parameters typical of those

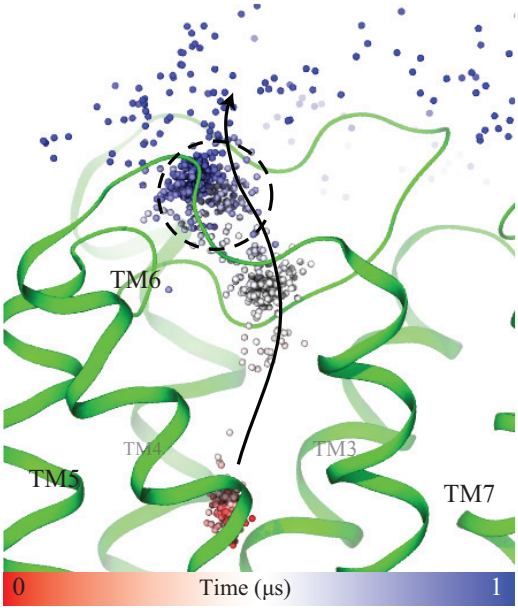

Figure 5: A simulation of a drug unbinding from a G protein–coupled receptor using one of the extended capabilities of Anton. GPCRs represent the largest class of drug targets, and one-third of all drugs act by binding to them. Here, the target value of the harmonic distance restraint between the protein and the ligand is varied as a function of time to gently push the drug, tiotropium, out of the binding pocket of the M3 muscarinic receptor [24]. Simulations suggest that, as the ligand exits the pocket, it pauses in the extracellular vestibule in the region outlined with a dashed circle. Spheres represent positions of tiotropium's C3 tropane atom at successive points in time from early (red) to late (blue). The direction of motion is indicated by the arrow. (Figure adapted from [24].)

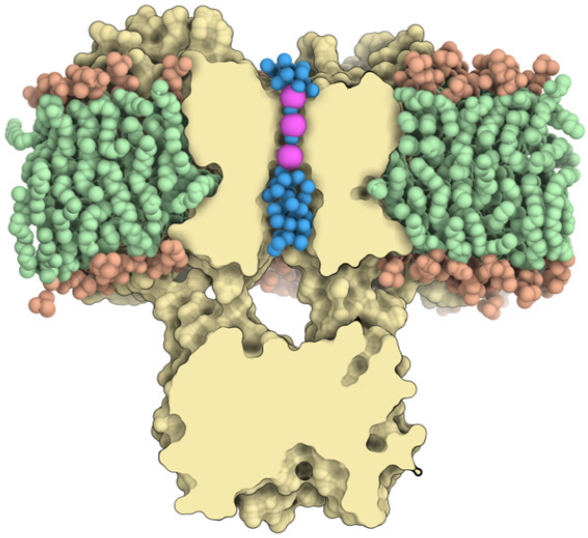

Figure 6: A simulation snapshot of a voltage-gated potassium channel (K^+ channel 2). The figure shows the active, ion-conducting state of this protein as a (cross-sectional) yellow surface with three ions (pink) interspersed with water molecules (blue) permeating the ion-selective filter (SF) of the channel. Water molecules occupying the pore cavity and the lipid membrane (orange and green) are also shown (for clarity, water molecules and ions hydrating the membrane and the extra-membrane part of the protein are not shown). A transition between the open, conducting and the closed, non-conducting states of the channel occurs as the transmembrane voltage switches from a positive to negative value, which is part of how nerve signals propagate. Thanks to Anton's support for extended MD methods, scientists could study this transition [27] with the voltage imposed with the use of a uniform electric field [28], to reveal the mechanism of ion permeation at the level of a single ion and assess how accurately, given the limitations of current force fields, MD simulations can reproduce conduction through the channel with and without nonstandard interactions between the potassium ions and the SF [49].

that scientists use in Anton simulations. To simulate NVT and NPT ensembles, we employed a Nosé-Hoover thermostat applied every 24 time steps and an MTK barostat applied every 480 time steps. For the application of distance and conformation restraints, we chose an interval of 12 time steps. Conformation restraints were applied to a subset of atoms in the protein backbones. We applied TAMD on the target value of distance restraints. For simulated tempering, the transition attempt interval was 10 ps.

CMAP terms were added by parameterizing systems with the CHARMM27 force field [50]. To add virtual sites, we adopted the TIP4P model [51] for all water molecules. The TIP4P model uses, in addition to the three real atoms, one virtual particle for each water molecule.[3] To add nonstandard interactions, we selected, for all systems, parameters con-

[3] The implementation of TIP4P water on Anton takes advantage of the fact that all atoms in a water molecule are located on the same node to perform virtual site calculations locally. "Generalized virtual sites," in which the atoms determining a virtual site's position may be located on more than one node, are currently only used for force field experimentation, and their performance is not measured here.

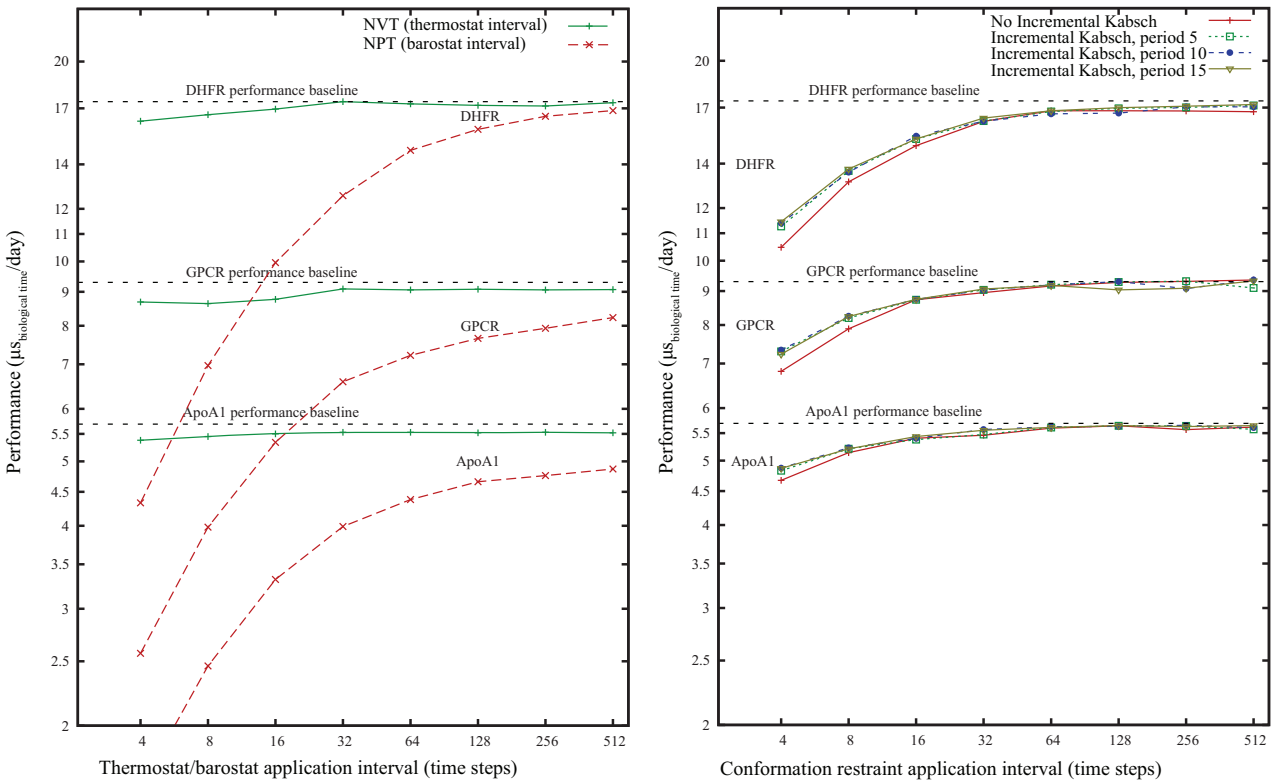

Figure 7: Performance of simulations with extended techniques applied at different intervals, for three biological systems (DHFR, GPCR and APOA1). On the left, performance of simulations of NVT and NPT ensembles are plotted as a function of the thermostat/barostat application interval. On the right, performance of simulations with conformation restraints are plotted as a function of restraint application interval; different data series compare the traditional and the incremental version of Kabsch's algorithm (discussed in Section V).

sistent with the originally published simulations of K^+ channel 1. This corresponded to adding nonstandard interactions between 24 protein oxygen atoms and potassium ions added at a concentration of 500 mM, which produced between 672 (BPTI) and 12,312 (K^+ channel 2) nonstandard interactions.

As a point of comparison, the fastest reported non-Anton simulation performance of the DHFR and ApoA1 benchmarks of which we are aware of is 0.471 and 0.289 μs/day, respectively [48], obtained using Desmond, a software package for MD simulation on commodity clusters [52]. This performance was achieved on a cluster consisting of 512 nodes, each with two 2.66-GHz Intel Xeon E5430 processors, connected by a DDR InfiniBand network. DHFR used only two of the eight cores on each node in order to increase network bandwidth per core.

In addition to the performance results presented above, we also analyze the performance of simulations that utilize multiple extended methods at the same time. In particular, we consider the sets of methods used in the published simulations on the GPCR and K^+ channel systems. The K^+ channel systems employ an NPT ensemble, a uniform electric field, and the CHARMM27 protein force field, which includes CMAP terms. Although the simulations published on this system [27] used a Berendsen thermostat and a barostat applied at every time step, in our simulations we use the same Nosé-Hoover/MTK barostat combination described above for consistency. Additionally, K^+ channel 1 uses nonstandard interactions [53], while K^+ channel 2 uses a conformation restraint. The GPCR system used a TAMD-controlled distance restraint, an NPT ensemble, and the CHARMM27 force field with CMAP terms. Our measurements are reported in Table 2. We find that the sum of the individual percentage slowdowns associated with each method is usually a good predictor of the slowdown measured when all the methods in a set are used; the slowdowns in the Table are within 5% of the sum of the individual percentage slowdowns reported in Table 1.

In Section III, we discussed the infrequent application of an MD method as a means to mitigate its performance impact. Figure 7 demonstrates how the choice of interval impacts the performance of a simulation. The performance of simulations for three biological systems (DHFR, ApoA1, and GPCR) are plotted as a function of the application interval of three extended MD operations (NVT thermostat, NPT barostat, and conformation restraints).

The plot of conformation restraint interval contains four data series. In the first data series, we used a traditional form of Kabsch's algorithm (discussed in Section V). In the other three data series, we used the incremental version of Kabsch's algorithm that accelerates convergence by reusing results from its previous run. We report performance as the incremental Kabsch algorithm is reset via a traditional run

	GPCR	K^+ channel 1	K^+ channel 2
System size (# of atoms)	53,005	107,117	229,983
Machine size (# of nodes)	512	512	1024
Baseline performance (μs/day)	9.10	5.22	5.13
Performance (μs/day)	7.62	4.23	3.65
Slowdown	1.19	1.23	1.41

Table 2: Aggregate slowdown measured for MD simulations when multiple extended capabilities are employed at the same, relative to the baseline MD performance. The combinations of systems and features measured replicate those presented in biochemical research published by our group [24, 27, 28].

every 5, 10, or 15 iterations. Note that, because of the parallelization approach, the performance of conformation restraints is approximately invariant over the number of atoms restrained.

Unsurprisingly, slowdowns are approximately inversely proportional to the application interval of MD methods. At long intervals, the performance asymptotically approaches a value near the baseline simulation performance. The asymptotic performance for NPT is not as high as the baseline performance because of the specialized builds described in Section III: the specialized build of the baseline simulation does not require code for the barostat and virial computation, and can thus reduce branching and instruction cache usage, leading to higher performance.

The plot of NPT performance in Figure 5 illustrates a trade-off inherent in the barostat of [37], in that it permits application intervals of roughly 480 time steps, but requires several consecutive iterations of force and virial calculations for each barostat invocation. An alternative, such as the Berendsen or classic MTK barostat, would require less work per invocation but more frequent application.

VI. Conclusions

Extending Anton's software to support diverse methods in molecular dynamics has allowed biochemistry researchers to use Anton for a broad set of simulations while maintaining performance dramatically superior to that of general-purpose supercomputers. This performance was achieved using software approaches that exploited Anton's low-latency communication mechanisms and tight, fine-grained coupling between programmable cores and hard-wired pipelines, in order to effectively parallelize computation in the face of Amdahl's law. As a result, the extended methods discussed in this paper incur only a modest slowdown, allowing simulations using combinations of complex methods to run at a substantial fraction of peak performance. We believe that extending the repertoire of MD methods on Anton, combined with the performance advantage of special-purpose hardware, will enable the study of biological processes that are currently considered outside the reach of traditional MD methods.

More broadly, our experience has shown that specialized machines can support a diverse set of methods while maintaining a substantial performance advantage over general-purpose machines. Future specialized machines may benefit from some of the hardware features we exploited in implementing these methods on Anton, including support for fine-grained operations and low-latency communication within and between chips. Likewise, the features of Anton's software architecture that facilitated efficient implementation of these methods, including our approach to enabling variable-frequency execution of a wide variety of methods, may prove applicable on other high-performance computing platforms and in other application domains.

Acknowledgments

We thank Daniel Arlow, Morten Jensen, Kresten Lindorff-Larsen, and Stefano Piana for their help in specifying and testing the MD methods described, as well as for providing biochemical systems for our performance measurements; Michael Eastwood, John Jumper, and John Klepeis for helpful discussions; Liming Zhao for the workflow system that we used for measurements; Peter Bogart-Johnson, Anissa Harper, Eric Radman, Chris Snyder, and Linda Stefanutti for their ongoing care and feeding of all our Anton machines; and Mollie Kirk for editorial assistance.

	BPTI	DHFR	GPCR	ApoA1	K^+ channel 1	K^+ channel 2
System size (# of atoms)	13,543	23,558	53,005	92,224	107,117	229,983
Machine size (# of nodes)	512	512	512	512	512	1024
Baseline performance (μs/day)	18.1	17.4	9.10	5.69	5.22	5.13
NVT ensemble	1.02	0.99	0.98	1.01	1.01	1.00
NPT ensemble	1.05	1.03	1.09	1.15	1.15	1.18
Distance Restraints	1.05	1.05	0.99	1.02	1.01	1.01
Distance Restraints + TAMD	1.07	1.07	1.00	1.01	1.01	1.02
Conformation Restraints	1.19	1.18	1.06	1.05	1.05	1.06
Simulated Tempering	1.07	1.04	1.00	1.02	1.02	1.01
Uniform E Field	1.16	1.22	1.01	1.10	1.07	1.09
CMAP	1.30	1.38	1.05	1.03	1.03	1.07
Virtual Sites	1.05	1.15	1.54	1.10	1.10	1.15
Nonstandard Interactions	1.07	1.37	1.17	1.13	1.04	1.05

Table 1. Slowdowns caused by each of the extended methods discussed in this paper, relative to baseline performance with all extended methods disabled. We evaluate the performance across six biological systems, presented in order of increasing number of atoms. Two systems (DHFR, and ApoA1) are common benchmark systems; the other four (BPTI, GPCR, K^+ channel 1, and K^+ channel 2) come from recent biochemical studies [24, 25, 27].

REFERENCES

[1] R. D. Fine, G. Dimmler, and C. Levinthal, "FASTRUN: A special purpose, hardwired computer for molecular simulation," Proteins, vol. 11, pp. 242–253, 1991.

[2] M. Taiji, T. Narumi, Y. Ohno, N. Futatsugi, A. Suenaga, N. Takada, and A. Kanagaya, "Protein Explorer: a petaflops special purpose computer system for molecular dynamics simulations," Proc. ACM/IEEE Conf. on Supercomputing (SC03), New York, NY, 2003. ACM.

[3] S. Toyoda, H. Miyagawa, K. Kitamura, T. Amisaki, E. Hashimoto, H. Ikeda, and N. Miyakawa, "Development of MD Engine: high-speed accelerator with parallel processor design for molecular dynamics simulations," J. Comput. Chem., vol. 20, pp. 185–199, 1999.

[4] S. Habata, K. Umezawa, M. Yokokawa, and S. Kitawaki, "Hardware system of the Earth Simulator." Parallel. Comput., vol. 30, pp. 1287–1313, 2004.

[5] J. Makino, E. Kokubo, and T. Fukushige, "Performance evaluation and tuning of GRAPE-6—towards 40 'real' Tflops," Proc. ACM/IEEE Conf. on Supercomputing (SC03), New York, NY, 2003. ACM.

[6] R. Perley, P. Napier, J. Jackson, B. Butler, B. Carlson, D. Fort, et al., "The expanded Very Large Array," Proc. IEEE, vol. 97, 1448–1462, 2009.

[7] M. Awad, "FPGA supercomputing platforms: a survey," Proc. Intl. Conf. on Field Programmable Logic and Applications (FPL 2009), New York, NY, 2009. IEEE.

[8] P. A. Boyle, D. Chen, N. H. Christ, M. A. Clark, S. D. Cohen, C. Cristian, et al., "Overview of the QCDSP and QCDOC computers," IBM J. Res. Dev., vol. 49, pp. 351–365, 2005.

[9] J. D. Bakos, "High-performance heterogeneous computing with the Convey HC-1," Comput. Sci. Eng., vol. 12, pp. 80–87, 2010.

[10] V. Heuveline and J.-P. Weiß, "Lattice Boltzmann methods on the ClearSpeed Advance™ accelerator board," Euro. Phys. J. Special Topics, vol. 171, pp. 31–36, 2009.

[11] H. Baier, H. Boettiger, M. Drochner, N. Eicker, U. Fischer, Z. Fodor, et al., "QPACE: power-efficient parallel architecture based on IBM PowerXCell 8i," Comput. Sci. Res. Dev., vol. 25, pp. 149–154, 2010.

[12] G. Goldrian, T. Huth, B. Krill, J. Lauritsen, H. Schick, I. Ouda, et al., "QPACE: quantum chromodynamics parallel computing on the Cell Broadband Engine," Comput. Sci. Eng. vol. 10, 46–54, 2008.

[13] F. Belletti, M. Cotallo, A. Cruz, L. A. Fernández, A. Gordillo, M. Guidetti, et al., "JANUS: an FPGA-based system for high performance scientific computing," Comput. Sci. Eng. vol. 11, 48–58, 2009.

[14] F. Belletti, M. Cotallo, A. Cruz, L. A. Fernández, A. Gordillo, A. Maiorano, et al., "IANUS: scientific computing on an FPGA-based architecture," in Parallel Computing: Architectures, Algorithms and Applications, vol. 38, C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr, and F. Peters, Eds., Jülich: John von Neumann Institute for Computing, pp. 553–560, 2007.

[15] F. Belletti, S. F. Schifano, R. Tripiccione, F. Bodin, P. Boucaud, J. Micheli, et al., "Computing for LQCD: apeNEXT," Comput. Sci. Eng. vol. 8, pp. 18–29, 2006.

[16] P. Kogge, "The tops in flops," IEEE Spectrum, vol. 48, pp. 48–54, 2011.

[17] D. E. Shaw, M. M. Deneroff, R. O. Dror, J. S. Kuskin, R. H. Larson, J. K. Salmon, et al., "Anton: a special-purpose machine for molecular dynamics simulation," Proc. 34th Annual Intl. Symp. on Computer Architecture (ISCA '07), New York, NY, 2007. ACM.

[18] D. E. Shaw, R. O. Dror, J. K. Salmon, J.P. Grossman, K. M. Mackenzie, J. A. Bank, et al., "Millisecond-scale molecular dynamics simulations on Anton," Proc. Intl. Conf. for High Performance Computing, Networking, Storage and Analysis (SC09), New York, NY, 2009. ACM.

[19] R. H. Larson, J. K. Salmon, R. O. Dror, M. M. Deneroff, C. Young, J.P. Grossman, et al., "High-throughput pairwise point interactions in Anton, a specialized machine for molecular dynamics simulation," Proc. 14th Annual Intl. Symp. on High-Performance Computer Architecture (HPCA '08), New York, NY, 2008. IEEE.

[20] J. S. Kuskin, C. Young, J.P. Grossman, B. Batson, M. M. Deneroff, R. O. Dror, and D. E. Shaw, "Incorporating flexibility in Anton, a specialized machine for molecular dynamics simulation," Proc. 14th Annual Intl. Symp. on High-Performance Computer Architecture (HPCA '08), New York, NY, 2008. IEEE.

[21] R. O. Dror, J.P. Grossman, K. M. Mackenzie, B. Towles, E. Chow, J. K. Salmon, et al., "Exploiting 162-nanosecond end-to-end communication latency on Anton," Proc. Intl. Conf. for High Performance Computing, Networking, Storage and Analysis (SC10), Washington, D.C., 2010. IEEE.

[22] R. O. Dror, A. C. Pan, D. H. Arlow, D. W. Borhani, P. Maragakis, Y. Shan, et al., "Pathway and mechanism of drug binding to G-protein-coupled receptors," Proc. Natl. Acad. Sci. USA, vol. 108, pp. 13118–13123, 2011.

[23] Y. Shan, E. T. Kim, M. P. Eastwood, R. O. Dror, M. A. Seeliger, and D. E. Shaw, "How does a drug molecule find its target binding site?" J. Am. Chem. Soc., vol. 133, pp. 9181–9183, 2011.

[24] A. C. Kruse, J. Hu, A. C. Pan, D. H. Arlow, D. M. Rosenbaum, E. Rosemond, et al., "Structure and dynamics of the M3 muscarinic acetylcholine receptor," Nature, vol. 482, pp. 552–556, 2012.

[25] D. E. Shaw, P. Maragakis, K. Lindorff-Larsen, S. Piana, R. O. Dror, M. P. Eastwood, et al., "Atomic-level characterization of the structural dynamics of proteins," Science, vol. 330, pp. 341–346, 2010.

[26] K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw, "How fast-folding proteins fold," Science, vol. 334, pp. 517–520, 2011.

[27] M. Ø. Jensen, V. Jogini, D. W. Borhani, A. E. Leffler, R. O. Dror, and D. E. Shaw, "Mechanism of voltage gating in potassium channels," Science, vol. 336, pp. 229–233, 2012.

[28] M. Ø. Jensen, V. Jogini, M. P. Eastwood, D. E. Shaw, "Atomic-level simulation of current–voltage relationships in single-file ion channels," J. Gen. Phys., in press.

[29] Y. Shan, M. P. Eastwood, X. Zhang, E. T. Kim, A. Arkhipov, R. O. Dror, et al., "Oncogenic mutations counteract intrinsic disorder in the EGFR kinase and promote receptor dimerization," Cell, vol. 149, pp. 860–870, 2012.

[30] R. O. Dror, D. H. Arlow, P. Maragakis, T. J. Mildorf, A. C. Pan, H. Xu, et al., "Activation mechanism of the β_2-adrenergic receptor," Proc. Natl. Acad. Sci. USA, vol. 108, pp. 18684–18689, 2011.

[31] D. M. Rosenbaum, C. Zhang, J. A. Lyons, R. Holl, D. Aragao, D. H. Arlow, et al., "Structure and function of an irreversible agonist-β2 adrenoceptor complex," Nature, vol. 469, pp. 236–240, 2011.

[32] Y. Shan, J. L. Klepeis, M. P. Eastwood, R. O. Dror, and D. E. Shaw, "Gaussian split Ewald: a fast Ewald mesh method for molecular simulation," J. Chem. Phys., vol. 122, pp. 054101:1–13, 2005.

[33] M. Wilkes, "The best way to design an automatic computing machine," Report of Manchester University Computer Inaugural Conference, pp. 182–184, 1951.

[34] P. H. Hunenberger, "Thermostat algorithms for molecular dynamics simulations," Adv. Polym. Sci., vol. 173, pp. 105–149, 2005.

[35] J. K. Salmon, M. A. Moraes, R. O. Dror, and D. E. Shaw, "Parallel random numbers: as easy as 1, 2, 3," Proc. Intl. Conf. for High Performance Computing, Networking, Storage and Analysis (SC11), New York, NY, 2011. ACM.

[36] G. J. Martyna, M. L. Klein, and M. Tuckerman, "Nosé-Hoover chains: the canonical ensemble via continuous dynamics," J. Chem. Phys., vol. 97, pp. 2635–2643, 1992.

[37] R. A. Lippert, C. Predescu, D. Ierardi, M. P. Eastwood, Ron O. Dror, and D. E. Shaw, "Multi-phased integrator for constant temperature and pressure molecular dynamics simulations," under review.

[38] D. M. Zuckerman, "Equilibrium sampling in biomolecular simulations," Annu. Rev. Biophys., vol. 40, pp. 41–62, 2011.

[39] G. M. Torrie and J. P. Valleau, “Nonphysical sampling distributions in Monte Carlo free energy estimation: umbrella sampling,” J. Comput. Phys., vol. 23, pp. 187–199, 1977.

[40] H. Grubmüller, B. Heymann, and P. Tavan, “Ligand binding: molecular mechanics calculation of the streptavidin-biotin rupture force,” Science, vol. 271, pp. 997–999, 1996.

[41] S. Izrailev, S. Stepaniants, M. Balsera, Y. Oono, and K. Schulten, “Molecular dynamics study of unbinding of the avidin-biotin complex,” Biophys J., vol. 72, pp. 1568–1581, 1997.

[42] L. Maragliano and E. Vanden-Eijnden, “A temperature accelerated method for sampling free energy and determining reaction pathways in rare events simulations,” Chem. Phys. Lett., vol. 426, pp. 168–175, 2006.

[43] W. Kabsch, “A solution of the best rotation to relate two sets of vectors,” Acta Crystallogr., vol. A32, pp. 922–923, 1976.

[44] W. Kabsch, “A discussion of the solution for the best rotation to relate two sets of vectors,” Acta Crystallogr., vol. A34, pp. 827–828, 1978.

[45] E. Marinari and G. Parisi, “Simulated tempering: a new Monte Carlo scheme,” Europhys. Lett., vol. 19, pp. 451–458, 1992.

[46] Y. Sugita and Y. Okamoto, “Replica-exchange molecular dynamics method for protein folding,” Chem. Phys. Lett., vol. 314, 1–2, 1999.

[47] P. Atkins and J. de Paula, Atkins' Physical Chemistry, 8th ed., Oxford: Oxford UP, pp. 637.

[48] E. Chow, C. A. Rendleman, K. J. Bowers, R. O. Dror, D. H. Hughes, J. Gullingsrud, F. D. Sacerdoti, and D. E. Shaw, “Desmond performance on a cluster of multicore processors.” D. E. Shaw Research Technical Report DESRES/TR—2008-01. http://deshawresearch.com. 2008.

[49] B. Roux, The membrane potential and its representation by a constant electric field in computer simulations. Biophys. J. vol. 95, pp. 4205–4216, 2008.

[50] A. D. MacKerell Jr., M. Feig, and C. L. Brooks III, “Extending the treatment of backbone energetics in protein force fields: limitations of gas-phase quantum mechanics in reproducing protein conformational distributions in molecular dynamics simulations,” J. Comput. Chem., vol. 25, pp. 1400–1415, 2004.

[51] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein, “Comparison of simple potential functions for simulating liquid water,” J. Chem. Phys., vol. 79, pp. 926–935, 1983.

[52] K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, et al., “Scalable algorithms for molecular dynamics simulations on commodity clusters,” Proc. of the ACM/IEEE Conf. on Supercomputing (SC06), New York, NY, 2006. IEEE.

[53] S. Berneche and B. Roux, “Energetics of ion conduction through the K^+ channel,” Nature, vol. 414, pp. 73–77, 2001.