Co-design for Particle Applications at Exascale

Samuel Temple Reeve, Jean-Luc Fattebert, Stephen DeWitt, Pablo Seleson, David Joy, and Stuart Slattery, *Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA*

Aaron Scheinberg, *Jubilee Development, Arlington, MA, 02476, USA*

Rene Halver, *Forschungszentrum Jülich, 52425, Jülich, Germany*

Christoph Junghans, Christian F. A. Negre, Michael E. Wall, Yu Zhang, and Anders M. Niklasson, *Los Alamos National Laboratory, Los Alamos, NM, 87545, USA*

James Belak, *Lawrence Livermore National Laboratory, Livermore, CA, 94550, USA*

Susan M. Mniszewski and Danny Perez, *Los Alamos National Laboratory, Los Alamos, NM, 87545, USA*

Co-design across the Exascale Computing Project has been critical for both enabling science applications and bringing disparate communities together. Developing and porting applications to the various high-performance computing architectures on pre-exascale and exascale computers has been quite challenging due to the diversity of hardware features and software stacks. The Co-design Center for Particle Applications (CoPA) has developed and enhanced the Cabana and Parallel, Rapid O(N), and Graph-Based Recursive Electronic Structure Solver (PROGRESS)/Basic Matrix Library (BML) libraries to facilitate the creation of new particle applications, make existing particle applications exascale capable, and allow teams to explore new capabilities. Particle methods from the atomistic, mesoscale, and continuum through cosmological scales have been built with Cabana, along with new possibilities for application coupling. Similarly, the PROGRESS/BML library has enabled quantum particle applications with linear algebra solvers to use advanced hardware. Across these CoPA-developed libraries, the co-design abstraction layer combines performance portability with math library support to facilitate the separation of concerns and directly support science runs.

The Co-design Center for Particle Applications (CoPA)[1] has developed computational tools to enable the scientific readiness of particle-based applications on exascale architectures. Under the particle "motif," CoPA focuses on several "submotifs," including short-range particle interactions [e.g., those that often dominate molecular dynamics (MD) and peridynamic (PD) methods], long-range particle interactions (e.g., electrostatic MD and gravitational N body), particle-in-cell (PIC) methods, and electronic structure solvers and quantum MD (QMD) algorithms. Representative particle applications from the Exascale Computing Project (ECP)[2] are addressed within CoPA and help drive the co-design process as well as other applications.

Particle-based simulation approaches are ubiquitous in computational science and engineering. The particles might represent, for example, the atomic nuclei of quantum or classical MD methods, gravitationally interacting bodies, or material points in mesoscale/continuum simulations. In each case, particles interact with the surrounding environment through the local electronic structure by direct particle–particle interactions at short ranges and/or particle–mesh interactions between a particle and a local grid field that represents longer range effects.

Digital Object Identifier 10.1109/MCSE.2024.3384052
Date of publication 3 April 2024; date of current version 30 September 2024.

CoPA has developed both libraries and proxy applications (apps) to enable the exascale readiness of application partners through a co-design process. The goal of the co-design activity has been to integrate the software stack with emerging hardware technologies while developing software components that embody the most common application motifs. Two main library directions have emerged: one focused on QMD and another for most other particle methods. The Cabana particle library[3] targets nonquantum methods, and two are provided for QMD: the Parallel, Rapid O(N), and Graph-Based Recursive Electronic Structure Solver (PROGRESS) and the Basic Matrix Library (BML)[4] libraries. Each strives for performance portability, flexibility, and scalability, on both CPU and GPU architectures, by providing optimized data structures and layouts, data movement, algorithms, and parallel communication in the context of the submotifs they address.

Cabana focuses on short-range and long-range particle interactions for mesh-free applications (e.g., MD, PD, and N body) and hybrid particle–mesh (PIC) applications. PROGRESS and BML focus on algorithms for electronic structure and QMD applications. QMD is unique among particle methods, as it is computationally dominated by matrix operations, whereas the other submotifs are primarily limited by particle and PIC operations. Alongside the libraries, proxy apps are used to evaluate the viability of incorporating various algorithms, data structures, architecture-specific optimizations, and the associated tradeoffs; CoPA-developed proxy apps include ExaMiniMD, CabanaMD, ExaMPM [material point method (MPM)], CabanaPIC, HACCabana (N body), and ExaSP2 (QMD).

The Cabana library provides particle algorithm implementations and particle data structures. The algorithms span the space of particle operations necessary for supporting each relevant application type, spanning nearly all submotifs. Cabana users can leverage the algorithms and computational kernels provided by Cabana independent of whether they are also using the native data structures. This includes intranode (i.e., local and threaded) operations on particles and internode (i.e., communication between nodes) operations to form a hybrid parallel capability. Cabana uses the ECP Kokkos programming model for on-node parallelism together with the message passing interface (MPI), providing performance and portability on current and anticipated future exascale systems developed by the U.S. Department of Energy (DOE), including multicore CPUs and GPUs. Within Cabana, Kokkos is used for abstractions to memory allocation, array-like data structures, and parallel loop concepts, which allow a single source code to be written for multiple architectures.

The PROGRESS/BML QMD libraries provide increased productivity in the implementation and optimization of O(N) and O(N^3) complexity algorithms with a design in which the matrix operations are separate from the solver implementations. The computational framework relies on two main libraries: PROGRESS and BML. Electronic structure codes call the solvers in the PROGRESS library, which, in turn, rely on BML. The BML library provides basic matrix data structures and linear algebra operations. These linear algebra matrix operations are optimized based on the format of the matrix and the targeted architecture. Applications can also directly implement new and experimental algorithms using BML when they are not available in PROGRESS. The overarching goal is to construct a flexible library ecosystem that helps to quickly adapt and optimize electronic structure applications on exascale architectures.

The rest of this article provides more details on the Cabana particle library and PROGRESS/BML QMD libraries along with notable application examples.

CABANA PARTICLE LIBRARY

The Cabana library has been developed to support a wide range of particle-based applications across scientific domains, including fully mesh-free and hybrid particle–grid schemes (https://github.com/ECP-CoPA/Cabana). This capability is a product of co-design, where Cabana sits directly between general exascale parallel programming models and the application physics, adding an additional layer for separating concerns. Cabana provides performance-portable particle data structures, algorithms, and parallel communication as well as structured grids and particle–grid interpolation. Each scientific application uses a subset of this functionality, focusing on the physics across the particles and/or grid with as little emphasis as possible on the particle bookkeeping and underlying parallelism. Cabana does not support adaptive or unstructured meshes (although Cabana particles can be used with external packages that provide this), nor the complex matrix operations necessary for the quantum MD applications supported by PROGRESS/BML discussed throughout the next section.

Cabana's design strategy begins with Kokkos, a library for performance portability widely used for targeting multiple-hardware-vendor threaded back ends,[5] shown in Figure 1. Kokkos is not only a required dependency but is fundamental to Cabana; both Kokkos and Cabana are C++ libraries with heavy use of template metaprogramming. Cabana combines Kokkos with MPI for multinode performance portability across hardware architectures as a single source implementation. The

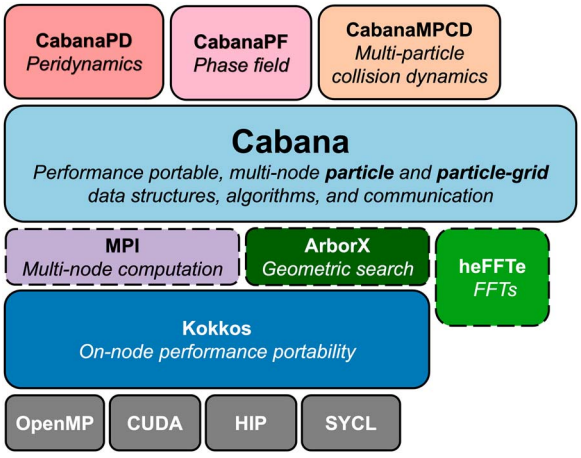

FIGURE 1. Cabana software stack, showing dependencies (optional ones shown with dashed outlines) and applications. FFT: fast Fourier transform; MPI: message passing interface.

intent is for the end user to primarily write Cabana code alongside calls to Kokkos and MPI as needed for operations either already clearly implemented by these programming models or for features not covered by the Cabana domain model.

One direct example of this idea is Cabana::neighbor_parallel_for, which extends Kokkos::parallel_for for the common need of iterating over the neighbors of all particles (e.g., a force update). Importantly, this includes flexible options for different types of threaded parallelism (over particles only or over particles and neighbors). However, it is also common to directly use Kokkos::parallel_for to update each particle independently, as in the case of an integration step (sometimes referred to as a particle "push"). Many optional library dependencies are also a part of Cabana; MPI is the primary example, while ArborX and heFFTe (both also developed throughout the ECP for spatial search and fast Fourier transforms, respectively) are highlighted in Figure 1, as they are used within the applications discussed here.

The following sections describe examples of using Cabana to create a mesh-free PD code, a phase–field code (using only the grid subpackage), and a hybrid particle–grid multiparticle collision dynamics (MPCD) code.

CabanaPD: Fracture Mechanics

Cabana has been leveraged to build a brand new fracture mechanics application, CabanaPD (https://github.com/ORNL/CabanaPD). A mesh-free PD approach has been implemented with multiple particle interaction models for elasticity and brittle fracture as well as infrastructure to generate initial structures, precracks, and boundary conditions. Each particle interacts via bonds with its neighborhood, which can contain hundreds to thousands of particles. Although the method is entirely mesh free, the particle–grid subpackage of Cabana is used for MPI communication. Bond breaking facilitates dynamic fracture simulation and can successfully model complex fracture phenomena, such as crack initiation, propagation, and branching. Ongoing developments of CabanaPD include the simulation of impact problems, failure of fiber-reinforced composite laminates, and crack formation in metals due to fusion-relevant transient thermal loads. CabanaPD has been tested with up to tens of billions of particles and thousands of neighbors per particle. Scaling on Oak Ridge Leadership Computing Facility (OLCF) Summit and Frontier with up to 1000 nodes has shown good parallel efficiency on both GPUs and CPUs.

Figure 2 shows an example simulation with CabanaPD, beginning with two prenotches in a steel plate that is hit by an impactor between those prenotches. Cracks then grow with a characteristic angle (highlighted in the figure with the color given by the strain energy density). In the future, Cabana could be leveraged to couple the mesh-free PD approach with mesh-free or grid-based classical continuum mechanics methods within CabanaPD for more efficient and targeted fracture simulations.

CabanaPF: Microstructure Prediction

CabanaPF is a new phase–field application built on Cabana (https://github.com/ORNL/CabanaPF). CabanaPF

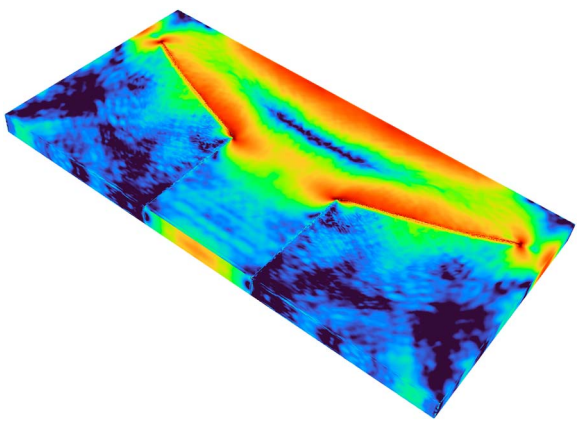

FIGURE 2. CabanaPD simulation of crack propagation in a prenotched steel plate, colored by strain energy density, highlighting the characteristic crack angle. Run on one node of Frontier MI250X GPUs.

uses a Fourier pseudospectral spatial discretization and a semi-implicit Euler time integration scheme to solve the Cahn–Hilliard equation, a fourth-order nonlinear partial differential equation at the core of many phase–field models. CabanaPF currently uses the structured grid capabilities of Cabana exclusively, using the interface to heFFTe, the MPI domain decomposition, and the Cabana::grid_parallel_for extension of Kokkos. CabanaPF has been verified and tested on both the Summit and Frontier supercomputers at Oak Ridge National Laboratory, in particular for the PFHub community benchmark problem 1a (https://pages.nist.gov/pfhub), a test problem for spinodal decomposition. In this problem, two initially mixed phases separate, tracked by the single conserved phase–field variable representing the solute composition.

Although phase–field is a grid-based method, and phase–field applications are nearly always written with a grid-based library or in isolation, the development of CabanaPF is strongly motivated by the potential for tight coupling with particle methods in multiphysics settings, which is directly enabled by Cabana. Applications such as the dendritic solidification of metal alloys are particularly relevant. In recent years, large-scale simulations combining phase–field solidification models with fluid flow models have demonstrated the effects of natural and forced convection of material microstructures (e.g., Takaki et al.[6]). With Cabana-based particle methods (including MPM and PD), this line of research can be extended to include the mechanical deformation of the metal dendrites due to fluid–structure interactions and external loads, void formation at dendrite roots due to density changes during phase transitions, and the initiation of fracture. Such capabilities would provide unprecedented insight into the complex interplay of phenomena occurring during solidification and, particularly, the far-from-equilibrium solidification in additive manufacturing.

CabanaMPCD: Hydrodynamics

CabanaMPCD implements MPCD, a particle-based description of hydrodynamic interactions in an incompressible fluid. MPCD is a stochastic collision scheme in which the fluid-describing particle gets rotated in velocity space in such a way that the momentum is conserved. CabanaMPCD uses the core particle portion of Cabana as well as the particle–grid subpackage for a fundamentally hybrid method. As discussed in the previous section, Cabana also provides a path forward for future work in direct coupling between MPCD and MD. The code exibits reasonable scaling behavior for sufficiently large test cases on CPUs as well as GPUs.

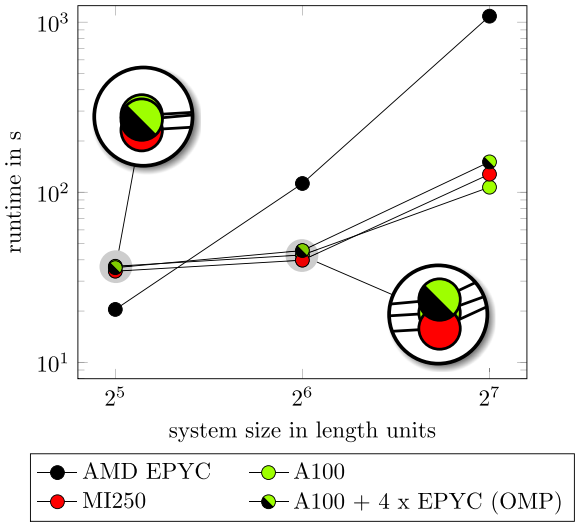

FIGURE 3. Size scaling results of the CabanaMPCD code on different GPUs and CPUs, run at the Jülich Supercomputing Centre. OMP: open multiprocessing.

Figure 3 shows the performance as a function of the system size (where the increase in linear size corresponds to an increase by a factor of eight in the number of unknowns) on a single node of the Jülich Supercomputing Centre JUWELS machine. This includes CPU-only (AMD EPYC Rome, with 48 cores with two OpenMP threads each), GPU-only (with four NVIDIA A100 or four AMD MI250X GPUs), and combined CPU + GPU execution to fully utilize all available hardware (four A100 and four EPYC MPI processes using 11 OpenMP threads each, which was the best performing setup for MPI ranks versus OpenMP threads, reserving the four remaining CPU cores for the GPU). A significant speedup is achieved on the GPU; however, combining GPU and CPU execution did not lead to improved performance in the current implementation. Additional details of the work can be found in Halver et al.,[7] including implementation strategy and roofline analysis showing the code is memory bound.

Other Cabana Applications

Cabana also enabled the exascale transition of the X-point gyrokinetic code (XGC) plasma physics code. This was initially done primarily with Fortran interfaces to Cabana particle data structures, now almost entirely converted into C++.[8] XGC is a notable Cabana application as a full production code that regularly scales to full DOE leadership machines (including OLCF Summit and Frontier) and as an unstructured mesh with embedded Cabana particles. Continuing work involves

moving performant XGC particle communication routines into Cabana and adding new particle sampling methods, which are useful to wide arrays of particle and, particularly, PIC applications.

A suite of codes for additive manufacturing is also in active development with Cabana for the performance-portable simulation of the 3-D printing process. These include relatively low-fidelity heat transfer (with a focus on the minimum time to solution) as well as full fidelity process simulation and microstructure simulation methods. As described throughout the section, these applications include particle-only, grid-only, and hybrid codes. In this context, Cabana continues to facilitate fast implementation by minimizing code that the application developer must write outside of physics kernels and coupling between methods in multifidelity or multimodal workflows by supporting both structured grids and particle systems.

There is additionally active development in Cabana-based MD, PIC, MPM, discrete element dynamics, and N-body cosmology, which are not described here in detail. Many other particle methods—e.g., smooth particle hydrodynamics—could leverage Cabana and, in turn, provide ideas and performant algorithms to current Cabana-based particle codes.

PROGRESS/BML LIBRARIES

Quantum MD application codes involve many different types of numerical operations to derive the atomic forces acting on each atom and propagate the atoms along their trajectories. These operations depend on the specific quantum model used for the electronic structure, its discretization or numerical basis set, some materials properties (metallic or not), and the often iterative solver used to solve the resulting equations. The most popular models for QMD include density functional theory and semiempirical quantum chemistry or tight-binding theory, and these are the ones we have been focusing on.

With the PROGRESS and BML libraries (https://github.com/lanl/qmd-progress and https://github.com/lanl/bml, respectively), we are targeting the major kernel found in many of these approaches: computing the so-called single-particle density matrix (DM). For an insulator, this is simply the projector onto the subspace spanned by the eigenvectors associated with the N lowest eigenvalues of the Hamiltonian operator. For a metallic system, this is slightly different, and, instead of a simple projector, this becomes a weighted projection, with weights of one for the lowest eigenvalues and zero for the highest as well as a smoothly varying occupation centered around the Fermi level in between given by the Fermi–Dirac distribution function.

The most straightforward way of tackling that problem is to solve a dense eigenvalue problem and build the DM from the computed eigenvectors. However, alternative solvers have been explored by the community over the last three decades to reduce the computational complexity from O(N^3) for a dense eigensolver to O(N) when taking advantage of the sparsity in the Hamiltonian and DM matrices. Many of those algorithms rely on approximating the DM as a polynomial of the Hamiltonian matrix, primarily recursive or Chebyshev polynomials. Truncation of small matrix elements is done at every step to preserve sparsity and reduce the computational cost to O(N). The truncation threshold needs to be carefully controlled to preserve accuracy. Such methods rely heavily on sparse–sparse matrix multiplications. It turns out that these algorithms also work very well for dense O(N^3) computations on GPUs due to the simplicity of the operations and the high degree of concurrency found in matrix–matrix multiplications.

To facilitate interfacing with various legacy codes in the field, many of which are Fortran-based, BML was implemented in C with a thin Fortran interface. PROGRESS mostly consists of Fortran code with a thin C interface. Beyond that, the implementation strategy adopted on the BML side is a hybrid one that depends on both the matrix format and the targeted architecture.

On the CPU side, optimized dense linear algebra libraries (BLAS, LAPACK, and ScaLAPACK) are used wherever possible for the dense format to achieve the best performance. For sparse formats, on the other hand, the code does not make much use of any third-party package. OpenMP threading is used throughout to take advantage of the multiple-CPU-cores architectures widely available.

On the GPU side, we rely heavily on third-party implementations of the numerically intensive kernels for both the dense and the ELLPACK formats (the only formats with GPU support at this time). The dense-matrices GPU implementation is based on the MAGMA library, which offers many of the needed linear algebra operations, from matrix–matrix multiplications to dense eigensolvers.[12] MAGMA is also used for memory management on the GPU and data transfer between the CPU and the GPU. To minimize data transfer, the primary location of the matrices and their coefficients in our implementation is on the GPU, and data transfer is done only when needed, for instance for input–output, setup, and diagnostics-related operations.

For the sparse ELLPACK format, the strategy consists of using the offloading capability of OpenMP4.5 (and beyond) for memory management and data

transfer as well as for offloading and parallelizing some loops on the GPU. This strategy has the advantage of enabling portable C code without even relying on another language or third-party library, as it is supported by many C compilers. However, the current OpenMP capabilities did not provide an acceptable level of performance for some critical kernels and led us to rely on vendor-optimized libraries for some kernels. Figure 4 summarizes the software stack integrating BML and PROGRESS within an electronic structure application.

In electronic structure, it is often difficult to take advantage of extra compute nodes due to the O(N^3) computational complexity of the solvers and the O(N^2) memory footprint. With O(N) approaches, on the other hand, one can scale up the atomistic size with the number of resources and expect the same time to solution. Using the graph-based matrix partitioning algorithm implemented in PROGRESS, large-scale applications are now possible (see the "Biomolecular MD Application" section). The technique is based on dividing the atomistic system of interest into a set of smaller linear algebra problems that can be solved independently and whose solutions can be combined together at the end.[9] The algorithm implemented in PROGRESS partitions the system based on the graphs associated with the matrices involved in the solver.

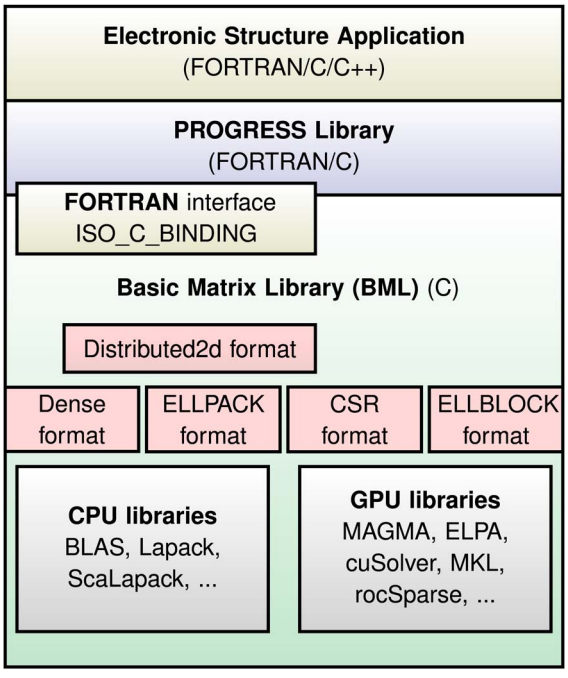

FIGURE 4. Software stack showing the PROGRESS and BML libraries and their integration within an electronic structure application. ...: symbol represents other relevant libraries.

While new high-performance computing resources can enable larger problems, in QMD, one is also interested in speeding up the calculation of moderate sizes, involving matrices of sizes ranging from 500 to 5000, to extend the length scale of these simulations and enhance sampling of the phase space and improve the quality of the results. Some algorithms based on the polynomial expansion of the DM can speed up solvers in this case. In addition, generating high-quality trajectories and sampling of that phase space can be enhanced by running multiple independent replicas of the system.[11] An example of such a use case is discussed in the next section.

Exascale Atomistics for Accuracy, Length, and Time (EXAALT) Application

The EXAALT project is an application project under the ECP that focuses on the development of ultrascalable MD approaches. The overarching goal of EXAALT is to provide a flexible simulation capability that can enable simulations in as much of the theoretically accessible accuracy/length/time exascale simulation space as possible. To do so, the EXAALT software stack combines three main simulation codes: 1) the ParSplice Accelerated MD code, 2) the large-scale atomic/molecular massive parallel simulator (LAMMPS) classical MD code, and 3) the Los Alamos Transferable Tight-Binding for Energetics (LATTE) semiempirical electronic structure code (see Figure 5). LATTE itself relies on PROGRESS and BML to efficiently implement linear algebra solvers. For small to intermediate spatial scales and long times, ParSplice[11] implements replica-based parallel-in-time acceleration techniques where trajectories generated by concurrent and independent MD simulations can be rigorously assembled to generate a single *dynamically correct* MD trajectory.

When high-accuracy simulations are required (e.g., to describe complex chemical reactions), the LAMMPS code is coupled with the LATTE engine to compute atomistic forces using electronic structure information. A targeted application is uranium dioxide (UO_2), which is a common fuel in nuclear reactors. Of specific interest is the understanding of how radiation-induced defects evolve, diffuse, and react within the material. Due to the complex chemistry and the long times required for the defects to evolve, the combination of ParSplice, LAMMPS, LATTE, PROGRESS, and BML is essential. This regime is especially challenging, as the modest system sizes that are required to keep the wall-clock time of each MD timestep low enough to enable long-time simulations lead to matrix sizes that are too small to take full advantage of modern GPUs,

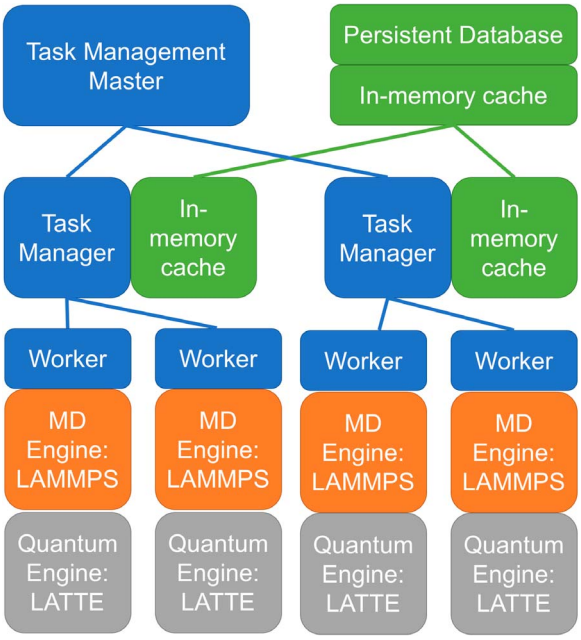

FIGURE 5. ParSplice accelerated MD workflow takes advantage of parallel computing resources by running many simulations simultaneously and assembling them into a longer MD. MD: molecular dynamics.

even when the state-of-the-art solvers provided by PROGRESS/BML are used. While this would lead to poor performance for conventional MD simulations, in the context of ParSplice AMD simulations, the figure of merit is the total MD throughput summed over all MD instances that simultaneously execute.

We have found that, on the nodes of Frontier, where eight MI250X graphic compute dies (GCDs) are available, GPU oversubscription leads to good aggregate performance in conjunction with PROGRESS/BML. Indeed, running 16 LATTE instances per node, each with four OpenMP threads per instance and oversubscribing each GPU/GCD with four/two LATTE instances, respectively, leads to performance improvements of about 5× relative to CPU-only runs for a UO_2 system with only 1344 electronic orbitals. In this case, the performance and flexibility of the PROGRESS/BML solvers in conjunction with the parallel-in-time ParSplice algorithm were essential to significantly increase the simulation throughput, which directly translates to a commensurate extension of the simulation time scales. When extrapolated to the entirety of Frontier, this level of performance would translate into an aggregate ParSplice simulation rate of up to 1 ps/wall-clock s, which is unprecedented for quantum MD simulations.

Biomolecular MD Application

MD simulations have become a cornerstone of drug discovery, revealing insights into microscopic processes involved in human disease at atomic detail. Most simulations only include classical effects and do not capture quantum-mechanical phenomena. Biomolecular MD simulations typically involve tens of thousands of atoms, for which long-duration quantum MD simulations historically have been out of reach computationally. However, quantum phenomena, such as chemical reactions and charge equilibration dynamics, can be crucial for modeling processes important for biomolecular function and disease.

Recently developed methods, using the graph-based partitioning of the system and distributed algorithms,[10] have decreased the time to solution for quantum electronic structure calculations, bringing long-duration quantum MD simulations of entire protein systems within reach. The graph-based approach allows partitioning of linear-scaling electronic structure calculations into smaller dense matrix algebra problems that are distributed using MPI.

Figure 6 shows a 64,112-atom protein system that was simulated on a CPU cluster by Negre et al.,[10] using a graph-based prototype QMD code, which

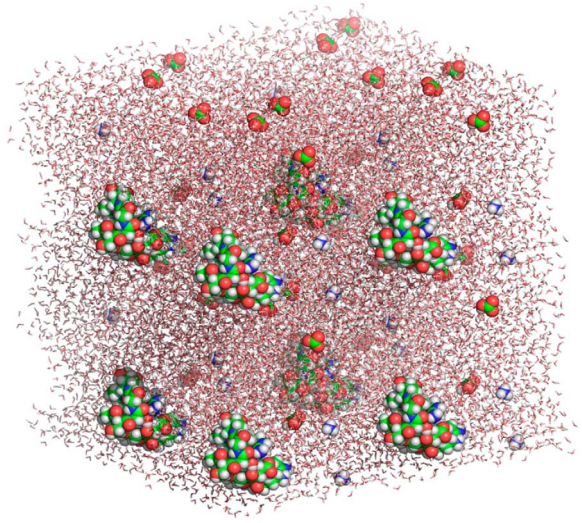

FIGURE 6. Eight protein molecules in a solution of ammonium bicarbonate and water in a periodic box with a total of 64,112 atoms. Using graph-distributed methods that had been used previously to perform distributed simulations on CPU clusters,[10] the PROGRESS/BML libraries enabled quantum molecular dynamics simulations of this biomolecular system to be performed on 128 nodes of Frontier, making use of all GPUs on each node. (Source: Negre et al.[10]; used with permission.)

implements self-consistent-charge density functional tight-binding theory to describe the electronic structure. This previously published result did not make use of GPUs; however, by design, the graph-based code uses the PROGRESS/BML libraries and, therefore, should be able to easily take advantage of the GPU extensions of the PROGRESS/BML libraries developed under the CoPA project. We tested the effectiveness of this design by building our graph-based QMD code against PROGRESS/BML libraries with GPU back-end matrix algebra solvers. The resulting build was used to run quantum MD simulations of the 64,112-atom system in Figure 6 on Frontier using all eight MI250X GCDs on each of 128 nodes. This test demonstrates that distributed electronic structure codes using the PROGRESS/BML libraries can run on hybrid exascale machines without the need to explicitly write any separate GPU code.

CONCLUSION

CoPA's co-design process has focused on library implementations, algorithm development, and interactions with particle applications represented within the Center. The Cabana library addresses particle applications with short-range, long-range, and particle–grid interactions. The PROGRESS/BML libraries address applications with a quantum-mechanical description of interaction. Having members with expertise in each submotif as application partners has allowed us to create these libraries as well as proxy apps with the necessary capabilities and performant implementations for all stakeholders. The applications highlighted in this article show the breadth of the usefulness of these libraries for developing new codes and extending existing particle codes to current and coming exascale architectures.

Key lessons learned for Cabana included how to recast different, seemingly application-specific algorithms into general particle or grid kernels as well as how to ensure performant and scalable Kokkos + MPI applications on a GPU (in particular, avoiding unnecessary memory allocations). For PROGRESS/BML, the realization that current OpenMP offload capabilities were not going to provide expected performance on GPUs led to the exploration of hybrid strategies combining OpenMP with platform-specific third-party libraries.

Crucially, the approaches used within CoPA can also be used to extend to future architectures beyond exascale. However, both Cabana and PROGRESS/BML rely heavily on external dependencies (Kokkos and OpenMP, respectively) whose support of next-generation hardware is necessary for applications based on CoPA libraries to use these architectures. Whether neuromorphic, field-programmable gate array, machine-learning-focused hardware, or otherwise, our co-designed libraries require performant implementations from the primary parallelism layers.

In addition, Cabana development will require bidirectional contributions into and out of the library to increase the sustainability of the software ecosystem. Cabana code that is not specific to particle/grid applications or MPI communication should be moved to Kokkos where possible. Conversely, increasing amounts of application code should be generalized and moved into Cabana to benefit other Cabana-based codes. PROGRESS/BML can also be extended using additional back-end libraries that enable the acceleration of linear algebra operations on future architectures. However, with sufficient development of external open source, performance-portable linear algebra libraries, PROGRESS/BML may be able to remove other dependencies (contingent upon sufficient functionality and performance on various architectures).

Development priorities for Cabana moving forward include better direct support for particle MPI communication through the grid, expansion of sparse (logically dense) grid capabilities, and load-balancing support. All of these efforts will make implementation easier and improve the performance and scalability of Cabana-based applications. The work on PROGRESS/BML enables linear-scaling electronic structure codes, such as our graph-based QMD code, to access GPU acceleration without the need to explicitly write any separate GPU code. This GPU acceleration is a key feature that will enable long-duration quantum MD simulations, in particular for large biomolecular systems for biomedically important applications.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy (DOE) by Los Alamos National Laboratory, Lawrence Livermore National Laboratory, and Oak Ridge National Laboratory under Contracts DE-AC52-06NA25396, DE-AC52-07NA27344, and DE-AC05-00OR22725, respectively, as part of the Co-design Center for Particle Applications, supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the DOE Office of Science and the National Nuclear Security Administration (NNSA). This research used resources of the Oak Ridge Leadership Computing Facility. Assigned: Los Alamos Unclassified Report 23-32396.

This article has been authored by UT-Battelle, LLC, under Contract DE-AC05-00OR22725 with the DOE.

REFERENCES

1. S. M. Mniszewski et al., "Enabling particle applications for exascale computing platforms," *Int. J. High-Perform. Comput. Appl. (IJHPCA)*, vol. 35, no. 6, pp. 572–597, 2021, doi: 10.1177/10943420211022829.
2. F. Alexander et al., "Exascale applications: Skin in the game," *Philos. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 378, no. 2166, 2020, Art. no. 20190056, doi: 10.1098/rsta.2019.0056.
3. S. Slattery et al., "Cabana: A performance portable library for particle-based simulations," *J. Open Source Softw.*, vol. 7, no. 72, 2022, Art. no. 4115, doi: 10.21105/joss.04115.
4. N. Bock et al., "The basic matrix library (BML) for quantum chemistry," *J. Supercomput.*, vol. 74, no. 11, pp. 6201–6219, 2018, doi: 10.1007/s11227-018-2533-0.
5. C. R. Trott et al., "Kokkos 3: Programming model extensions for the exascale era," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 4, pp. 805–817, Apr. 2022, doi: 10.1109/TPDS.2021.3097283.
6. T. Takaki, S. Sakane, M. Ohno, Y. Shibuta, and T. Aoki, "Large–scale phase–field lattice Boltzmann study on the effects of natural convection on dendrite morphology formed during directional solidification of a binary alloy," *Comput. Mater. Sci.*, vol. 171, 2020, Art. no. 109209, doi: 10.1016/j.commatsci.2019.109209.
7. R. Halver, C. Junghans, and G. Sutmann, "Using heterogeneous GPU nodes with a Cabana-based implementation of MPCD," *Parallel Comput.*, vol. 117, Sep. 2023, Art. no. 103033, doi: 10.1016/j.parco.2023.103033.
8. A. Scheinberg et al., "Kokkos and Fortran in the exascale computing project plasma physics code XGC," in *Proc. Super Comput. Conf.*, 2019, pp. 1–3.
9. A. M. N. Niklasson et al., "Graph-based linear scaling electronic structure theory," *J. Chem. Phys*, vol. 144, no. 23, 2016, Art. no. 234101, doi: 10.1063/1.4952650.
10. C. F. A. Negre, M. E. Wall, and A. M. N. Niklasson, "Graph-based quantum response theory and shadow Born-Oppenheimer molecular dynamics," *J. Chem. Phys.*, vol. 158, no. 7, 2023, Art. no. 074108, doi: 10.1063/5.0137119.
11. D. Perez, E. K. Cubuk, A. Waterland, E. Kaxiras, and A. F. Voter, "Long-time dynamics through parallel trajectory splicing," *J. Chem. Theory Comput.*, vol. 12, no. 1, pp. 18–28, 2016, doi: 10.1021/acs.jctc.5b00916.
12. J. Dongarra et al., "Accelerating numerical dense linear algebra calculations with GPUs," in *Numerical Computations with GPUs*, V. Kindratenko, Ed., Cham, Switzerland: Springer-Verlag, 2014, pp. 1–26.

SAMUEL TEMPLE REEVE is a staff scientist at Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA. His research interests include computational science and high-performance computing (HPC) software development for materials science. Reeve received his Ph.D. degree in materials engineering from Purdue University. He is a member of TMS, SIAM, and US-RSE. Contact him at reevest@ornl.gov.

JEAN-LUC FATTEBERT is a research scientist in the Computational Sciences and Engineering Division at the Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA. His research interests include HPC and quantum molecular dynamics simulations. Fattebert received his Ph.D. in applied mathematics from the Swiss Federal Institute of Technology. He is a member of SIAM. Contact him at fattebertj@ornl.gov.

STEPHEN DEWITT is a research scientist at Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA. His research interests include simulating the multiscale response of materials to complex processing conditions, with an emphasis on phase-field methods for microstructural evolution. DeWitt received his Ph.D. in applied physics from the University of Michigan. Contact him at dewittsj@ornl.gov.

PABLO SELESON is a research scientist in the Computer Science and Mathematics Division at Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA. His research interests include computational fracture modeling with peridynamics and surrogate modeling. Seleson received his Ph.D. in computational science from Florida State University. He is a member of SIAM, USACM, IACM, and ASCE. Contact him at selesonpd@ornl.gov.

DAVID JOY is a recent graduate of Auburn University and a Science Undergraduate Laboratory Internships participant at Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA. Joy received his B.S. degree in software engineering from Auburn University. Contact him at dhj0005@auburn.edu.

STUART SLATTERY is a senior scientist at Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA. His research interests include high performance computing, scientific software, and multiphysics simulation. Slattery received his Ph.D. degree in nuclear engineering and engineering physics from the Universit of Wisconsin-Madison. He is a member of SIAM. Contact him at slatterysr@ornl.gov.

AARON SCHEINBERG is a computational scientist and consultant with Jubilee Development, Arlington, MA, 02476, USA. His research interests include exascale computing, scientific application performance, and particle-based methods. Scheinberg received his Ph.D. degree in planetary science from Massachusetts Institute of Technology. Contact him at aaron@jubileedev.com.

RENE HALVER is a staff scientist at the Jülich Supercomputing Centre, part of Forschungszentrum Jülich, 52425, Jülich, Germany. His research interests include load-balancing schemes and performance portability approaches for particle-based simulation methods in HPC. Halver received his M.Sc. degree in technomathematics from the University of Applied Sciences Aachen. Contact him at r.halver@fz-juelich.de.

CHRISTOPH JUNGHANS is the group leader of the Applied Computer Science group at Los Alamos National Laboratory, Los Alamos, NM, 87545, USA. His research interests include scientific software development and engineering over molecular dynamics methods and multiscale simulation techniques. Junghans received his Ph.D. degree in physics from Johannes Gutenberg University Mainz. Contact him at junghans@lanl.gov.

CHRISTIAN F. NEGRE is a computational chemist at Los Alamos National Laboratory, Los Alamos, NM, 87545, USA. His research interests include the simulation of electron quantum dynamics and quantum molecular dynamics with a focus on a diverse array of applications. Negre received his Ph.D. degree in computational chemistry from the National University of Cordoba. Contact him at cnegre@lanl.gov.

MICHAEL E. WALL is a scientist in the Computer, Computational, and Statistical Sciences Division at Los Alamos National Laboratory, Los Alamos, NM, 87545, USA. His research interests include biomolecular quantum molecular dynamics simulations and protein crystallography. Wall received his Ph.D. degree in physics from Princeton University. Contact him at mewall@lanl.gov.

YU ZHANG is a scientist in the Theoretical Division at Los Alamos National Laboratory, Los Alamos, NM, 87545, USA. His research interests include nonadiabatic molecular dynamics, light–matter interactions and quantum computation and quantum information science. Zhang received his Ph.D. degree in physics from the University of Hong Kong. Contact him at zhy@lanl.gov.

ANDERS M. NIKLASSON is a scientist in the Theoretical Division at Los Alamos National Laboratory, Los Alamos, NM, 87545, USA. His research interests include the development of next-generation quantum-based molecular dynamics schemes using low-complexity algorithms as well as new mixed precision algorithms tailored for AI-accelerated hardware. Niklasson received his Ph.D. degree in physics from Uppsala University. Contact him at amn@lanl.gov.

JAMES BELAK is a senior scientist in the Materials Science Division at Lawrence Livermore National Laboratory, Livermore, CA, 94550, USA. His research interests include the application of HPC to equilibrium and nonequilibrium problems in materials physics, most recently additive manufacturing. Belak received his Ph.D. degree in physics from Colorado State University. He is a member of APS. Contact him at belak1@llnl.gov.

SUSAN M. MNISZEWSKI is the principal investigator (PI) of the Co-design Center for Particle Applications and a senior scientist in the Computer, Computational, and Statistical Sciences Division at Los Alamos National Laboratory, Los Alamos, NM, 87545, USA. Her research interests include HPC for quantum molecular dynamics simulations, computational co-design, and quantum computing. Mniszewski received a B.S. degree in computer science from Illinois Institute of Technology. She is a Member of IEEE, ACM, ACS, and SIAM. Contact her at smm@lanl.gov.

DANNY PEREZ is a scientist in the Theoretical Division at Los Alamos National Laboratory, Los Alamos, NM, 87545, USA, and the principal investigator (PI) of the Exascale Atomistics for Accuracy, Length, and Time project. His research interests include massively parallel methods for long-time simulations and large-scale workflows for machine learning. Perez received a Ph.D. degree in physics from Universite de Montreal. Contact him at danny_perez@lanl.gov.