THEME ARTICLE: Scientific Impact of the Exascale Computing Project

Co-design for Particle Applications at Exascale

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

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

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

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

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

Abstract—Co-design across the Exascale Computing Project (ECP) has been critical for both enabling science applications and bringing disparate communities together. Developing and porting applications to the various high-performance computing (HPC) 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 PROGRESS/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 atomistic, mesoscale, 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 separation of concerns and directly support science runs.

XXXX-XXX © 2023 IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

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 “sub-motifs” 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 mesocale/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 the particle-mesh interactions between a particle and a local grid field that represents longer range effects.

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 non-quantum 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 sub-motifs 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 sub-motifs are primarily limited by particle and particle-in-cell operations. Alongside the libraries, proxy apps are used to evaluate the viability of incorporating various algorithms, data structures, architecture-specific optimizations, and the associated trade-offs; 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 sub-motifs. 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 MPI, providing performance and portability on current and anticipated future exascale systems developed by the US Department of Energy, 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 paper 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 book-keeping 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 backends [5], shown in Figure 1. Kokkos is not only a required dependency, but fundamental to Cabana; both Kokkos and Cabana are C++ libraries with heavy use of template meta-programming. Cabana combines Kokkos with MPI for multi-node performance portability across hardware architectures as a single source implementation. The 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 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 peridynamics code, a phase-field code (using only the grid subpackage), and a hybrid particle-grid multi-particle collision dynamics (MPCD) code.

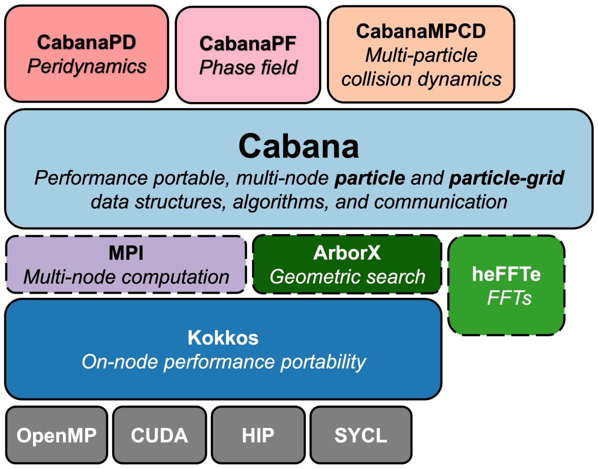
CabanaPD
Peridynamics
CabanaPF
Phase field
CabanaMPCD
Multi-particle collision dynamics
Cabana
Performance portable, multi-node particle and particle-grid data structures, algorithms, and communication
MPI
Multi-node computation
ArborX
Geometric search
heFFTe
FFTs
Kokkos
On-node performance portability
OpenMP
CUDA
HIP
SYCL

FIGURE 1. Cabana software stack, showing dependencies (optional shown with dashed outlines) and applications.

CabanaPD: fracture mechanics

Cabana has been leveraged to build a brand new fracture mechanics application, CabanaPD (https://github.com/ORNL/CabanaPD). A mesh-free peridynamic approach has been implemented with multiple particle interaction models for elasticity and brittle fracture as well as infrastructure to generate initial structures, pre-cracks, 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 OLCF Summit and Frontier with up to 1,000 nodes has shown good parallel efficiency on both GPUs and CPUs. Figure 2 shows an example simulation with CabanaPD, beginning with two pre-notches in a steel plate that is hit by an impactor between those pre-notches. 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 peridynamic

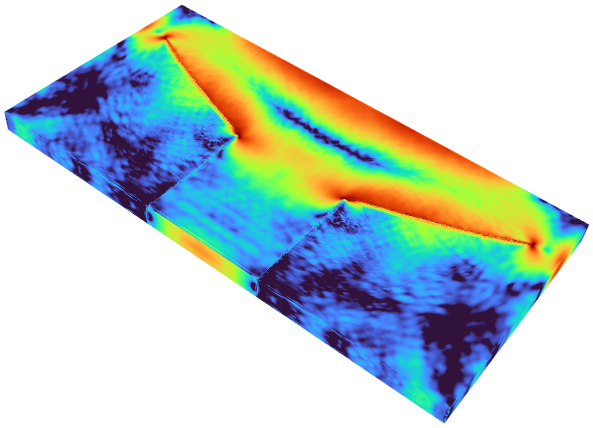

FIGURE 2. CabanaPD simulation of crack propagation in a pre-notched steel plate, colored by strain energy density, highlighting the characteristic crack angle. Run on one node of Frontier MI250X GPUs.

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 uses a Fourier pseudospectral spatial discretization and a semi-implicit Euler time integration scheme to solve the Cahn-Hilliard equation, a fourth-order nonlinear PDE 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 ORNL, 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 multi-physics settings which is directly enabled by Cabana. Applications such as 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., [6]). With Cabana-based particle methods (including MPM, PD) this line of research can be extended to include 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 multi-particle collision dynamics (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 behaviour for sufficiently large test cases, on CPUs as well as GPUs. Figure 3 shows the performance as a function of system size (where the increase in linear size corresponds to an increase by a factor of 8 in the number of unknowns) on a single node of the Jülich Supercomputing Centre JUWELS machine. This includes CPU only (AMD EPYC Rome, 48 cores with 2 OpenMP threads each), GPU only (with 4 NVIDIA A100 or 4 AMD MI250X), as well as combined CPU+GPU execution to fully utilize all available hardware (4 A100 and 4 EPYC MPI processes using 11 OpenMP threads each, which was the best performing setup for MPI ranks vs OpenMP threads, reserving the 4 remaining CPU cores for the GPU). A significant speedup is achieved on 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 [7], including implementation strategy and roofline analysis showing the code is memory-bound.

Other Cabana applications

Cabana also enabled the exascale transition of the XGC plasma physics code. This was initially done primarily with FORTRAN interfaces to Cabana particle

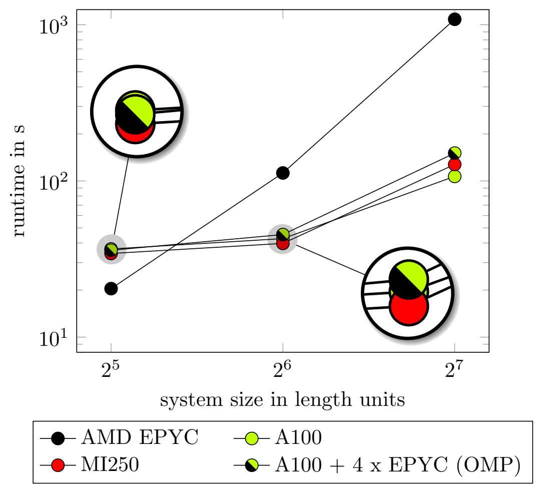

FIGURE 3. Size scaling results of the CabanaMPCD code on different GPU and CPUs, run at the Jülich Supercomputing Centre.

data structures, now almost entirely converted into C++ [8]. XGC is a notable Cabana application as a full production code which 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 are also in active development with Cabana for performance portable simulation of the 3D printing process. These include relatively low-fidelity heat transfer (with a focus on 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 multi-fidelity 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 (DEM), 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) as well as the often iterative solver used to solve the resulting equations. The most popular models for QMD include Density Functional Theory (DFT) and semi-empirical 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, https://github.com/lanl/bml), 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 projections with weights 1 for the lowest eigenvalues, 0 for the highest, and 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. But 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 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 both on the matrix format and the targeted architecture.

On the CPU side, optimized dense linear algebra libraries (BLAS, LAPACK, 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, both for 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. In order 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 I/O, setup, and diagnostics related operations. For the sparse ELLPACK format, the strategy consists in 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 Section "Biomolecular MD application" below). 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.

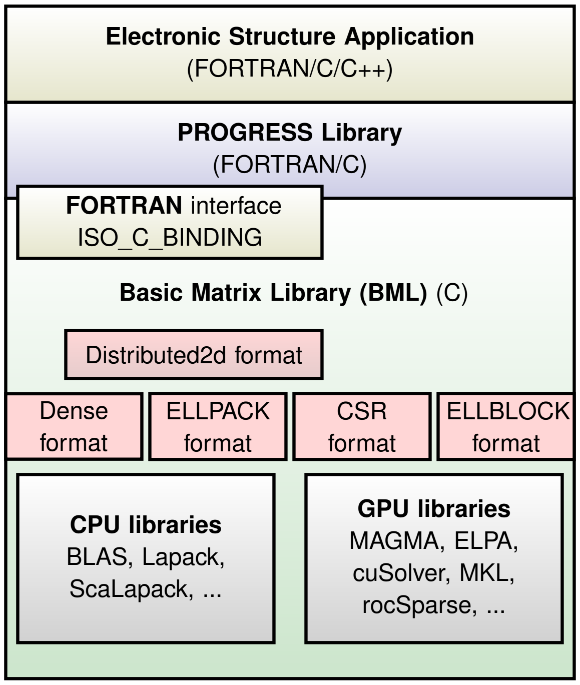

FIGURE 4. Software stack showing the PROGRESS and BML libraries and their integration within an electronic structure application.

While new HPC resources can enable larger problems, in quantum molecular dynamics, one is also interested in speeding-up 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 speedup 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.

EXAALT application

The EXAALT (EXascale Atomistics for Accuracy, Length, and Time) project is an application project under ECP which focuses on the development of ultra-scalable 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: i) the ParSplice Accelerated MD code, ii) the LAMMPS classical-MD code, and iii) the LATTE semi-empirical 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, 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 MI-250X GCDs are available, GPU oversubscription leads to good aggregate performance in conjunction with PROGRESS/BML. Indeed, running 16 LATTE instances per node, each with 4 OpenMP threads per instance and oversubscribing each GPU/GCD with four/two LATTE instances, respectively, leads to performance improvements of about 5x 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 timescales. When extrapolated to the entirety of Frontier, this level of performance would translate into an aggregate ParSplice simulation rate of up to 1ps/wall-clock second, which is unprecedented for quantum MD simulations.

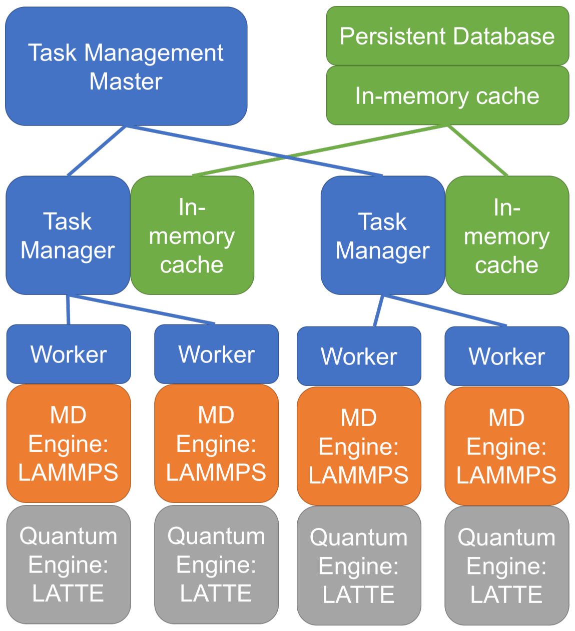

FIGURE 5. ParSplice accelerated MD workflow takes advantage of parallel computing resources by running simultaneously many simulations and assemble them into a longer MD.

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 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

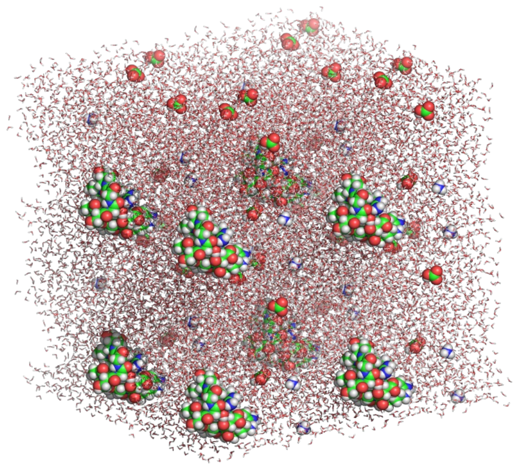

FIGURE 6. Eight protein molecules in a solution of ammonium bicarbonate and water in a periodic box with a total of 64,112 atoms. Using graph-distributed methods that had been used previously to perform distributed simulations on CPU clusters [10], the PROGRESS/BML libraries enabled quantum molecular-dynamics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

CPU cluster in Ref. [10], using a graph-based prototype QMD code, which implements a self-consistent-charge density functional tight-binding theory (SCC-DFTB) 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 backend 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-ranged, long-ranged, and particle-grid interactions. The PROGRESS/BML libraries address applications with a quantum mechanical description of interaction. Having members with the expertise in each sub-motif 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 devloping 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 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, FPGA, 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 in order 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 in order to benefit other Cabana-based codes. PROGRESS/BML can also be extended using additional backend libraries that enable 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 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 by LANL, LLNL, and ORNL 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 U.S. DOE Office of Science and the NNSA. This research used resources of the Oak Ridge Leadership Computing Facility (OLCF) Assigned: Los Alamos Unclassified Report (LA-UR) 23-32396

REFERENCES

1. S. M. Mniszewski, J. Belak, J.-L. Fattebert, C. F. A. Negre, S. Slattery, A. A. Adedoyin, *et al.*, "Enabling particle applications for exascale computing platforms," *International Journal of High-Performance Computing Applications (IJHPCA) Special Issue: ECP Co-design and computational motifs*. vol. 35 no. 6 pp. 572–597, 2021. doi:10.1177/10943420211022829
2. F. Alexander, A. Almgren, J. Bell, A. Bhattacharjee, J. Chen, P. Colella, *et al.* "Exascale applications: Skin in the game," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Science. vol. 378, no. 2166, 2020. doi:10.1098/rsta.2019.0056*
3. *S. Slattery, S. Reeve, C. Junghans, D. Lebrun-Grandie, R. F. Bird, G. Chen,* et al., *"Cabana: A Performance Portable Library for Particle-Based Simulations," Journal of Open Source Software. vol. 7 no. 72, pp. 4115, 2022. doi:10.21105/joss.04115*
4. *N. Bock, C. F. A. Negre, S. M. Mniszewski, J. Moyd-Yusof, B. Aradi, J.-L. Fattebert,* et al., *"The Basic Matrix Library (BML) for Quantum Chemistry," The Journal of Supercomputing. vol, 74, pp. 6201–6219, 2018. doi:10.1007/s11227-018-2533-0*
5. *C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood,* et al., *"Kokkos 3: Programming Model Extensions for the Exascale Era"* IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 805-817, 2022, doi:10.1109/TPDS.2021.3097283*
6. *T. Takaki, S. Sakane, M. Ohno, Y. Shibuta, 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" Computational Materials Science, pp. 109209, 2020 . doi:10.1016/j.commatsci.2019.109209*
7. *R. Halver, C. Junghans, G. Sutmann, "Using heterogeneous GPU nodes with a Cabana-based implementation of MPCD", Parallel Computing 117, 2023. doi:0.1016/j.parco.2023.103033*
8. *A. Scheinberg, G. Chen, S. Ethier, S. Slattery, R. Bird, P. Worley, and C. Chang, "Kokkos and Fortran in the exascale computing project plasma physics code XGC". In Proceedings of Sc19 Conference. 2019.*
9. *A. M. N. Niklasson, S. M. Mnizsewski, C. F. A. Negre, M. J. Cawkwell, P. J. Swart, J. Mohd-Yusof and T. C. Germann and M. E. Wall and N. Bock, E. H. Rubenssson, H. Djidjev. "Graph-based linear scaling electronic structure theory,"* J. Chem. Phys. *144, 234101, 2016. doi:10.1063/1.4952650.*
10. *C. F. A. Negre, M. E. Wall, A. M. N. Niklasson. "Graph-based quantum response theory and shadow Born-Oppenheimer molecular dynamics,"* J. Chem. Phys. *158, 074108, 2023. doi:10.1063/5.0137119.*
11. *D. Perez, E.K. Cubuk, A. Waterland, E. Kaxiras, A.F. Voter. "Long-Time Dynamics through Parallel Trajectory Splicing," J. Chem. Th. Comp. , pp. 18-28, 2016. doi:10.1021/acs.jctc.5b00916*
12. *J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, I. Yamazaki. "Accelerating Numerical Dense Linear Algebra Calculations with GPUs," In: Kindratenko, V. (eds) Numerical Computations with GPUs, pp. 1-26, 2014, Springer, Cham. doi:10.1007/978-3-319-06548-9_1*

Samuel Temple Reeve is a staff scientist at Oak Ridge National Laboratory working in computational science and HPC software development for materials science. 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. His current research interests include high-performance computing and quantum molecular-dynamics simulations. Contact him at fattebertj@ornl.gov.

Stephen DeWitt is a Research Scientist at Oak Ridge National Laboratory focused on simulating the multiscale response of materials to complex processing conditions, with an emphasis on phase-field methods for microstructural evolution. Contact him at dewittsj@ornl.gov.

Pablo Seleson is a Research Scientist in the Computer Science and Mathematics Division at Oak Ridge National Laboratory working on computational fracture modeling with peridynamics and surrogate modeling. Contact him at selesonpd@ornl.gov.

David Joy is a recent graduate of Auburn University and a Science Undergraduate Laboratory Internships (SULI) participant at Oak Ridge National Laboratory. Contact him at dhj0005@auburn.edu.

Stuart Slattery is a Senior Scientist at Oak Ridge National Laboratory working in computational science and HPC software development for engineering applications. Contact him at slatterysr@ornl.gov.

Aaron Scheinberg is a computational scientist and consultant focusing on exascale computing, scientific application performance, particle-based methods, magnetic fusion simulations, and GPU programming. Contact him at aaron@jubileedev.com.

Rene Halver is a staff scientist at the Jülich Supercomputing Centre, part of Forschungszentrum Jülich, where he is researching load-balancing schemes and performance portability approaches for particle-based simulation methods in HPC. 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. His reasearch interests span from scientific software development and engineering over molecular dynamics methods to multi-scale simulation techniques. Contact him at junghans@lanl.gov.

Christian F. Negre is a Scientist at Los Alamos National Laboratory. He is a computational chemist specializing in the simulation of electron quantum dynamics and quantum molecular dynamics, with a specific focus on a diverse array of applications, including crystallization, metallic nanoparticles, molecular electronics, photovoltaic systems, and biophysics. 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. His current research interests include biomolecular quantum molecular-dynamics simulations and protein crystallography. Contact him at mewall@lanl.gov.

Yu Zhang , is a Scientist in the Theoretical Division at Los Alamos National Laboratory. His current research interests include nonadiabatic molecular dynamics, light-matter interactions, quantum computation, and quantum information science. Contact him at zhy@lanl.gov.

Anders M. Niklasson , is a Scientist in the Theoretical Division at Los Alamos National Laboratory. His current research interests include the development of next generation quantum-based molecular dynamics schemes using low-complexity algorithms such as recursive sparse matrix algebra methods, graph-based electronic structure theory, as well as new mixed precision algorithms tailored for AI-accelerated hardware.

Danny Perez, is a Scientist in the Theoretical Division at Los Alamos National Laboratory and the PI of the EXXALT project. His current research interests include massively-parallel methods for long-time simulations and large-scale workflows for machine learning. Contact him at danny_perez@lanl.gov.

James Belak is a Senior Scientist in the Materials Science Division at Lawrence Livermore National Laboratory. His career has centered around the application of High Performance Computing to equilibrium and non-equilibrium problems in Materials Physics, most recently additive manufacturing. Contact him at belak1@llnl.gov.

Susan M. Mniszewski is the PI of the Co-design Center for Particle Applications (CoPA) and a Senior Scientist in the Computer, Computational, and Statistical Sciences Division at Los Alamos National Laboratory. Her research includes high performance computing for quantum molecular dynamics simulations, computational co-design, and quantum computing. Contact her at smm@lanl.gov.