Issues in the Typical NBI Implementations

- Creation of a domain-specific API for making highly reusable components covering the most compute-heavy parts of particle simulations.
- Increase modularity and maintainability of the performance-critical parts while simplifying prototyping for the end-user.

Expected Benefits

Generalized Particle Simulations

Applications for high-performance computation of pairwise interactions between large numbers of particles are present in biochemistry, material science, nanotech, astrophysics, fluid mechanics, etc. For big, heterogeneous clusters, data transfer and reduction tasks often dominate the compute loads. N-body solvers integrated into a package can be usable outside of these applicability transcends several domains.

Swapback Backends

The implementation of NBI calculators is tightly coupled to the simulation packages, which makes adding alternative NBI backends a huge challenge. A standardized interface for the features would allow explicit sharing of the components, enabling greater proliferation of methods across different classes of n-body solvers thus speeding up development.

Rapid Prototyping Custom Programs

Researchers often write programs that require NBI backends but end up writing their own as reusability of high-performance kernels in popular packages is poor. Exposing the NBI calculators in a common API would guarantee high performance in custom codes with reusable building blocks.

Architecture Optimized Schedules

The diversity of hardware/accelerators in HPC is rapidly rising. Hardware optimizations are package specific and this know-how isn’t easily transferable among users/developers. Execution schedules that optimally overlap computation, communication and reduction tasks for specific architectures, algorithms and workloads can easily be modularized to reuse.

Alternative Parallelization Paradigms

MPI is de facto choice for multimode parallelism, and this choice is tightly coupled with the software design as well. The API aims to clearly separate the model/architecture, parallelism and accelerator concerns thereby allowing concurrent development using alternatives like task-based parallelism.

Sustainable Feature Development

The exascale future brings great challenges. New algorithms, a plethora of new hardware and new features to cover emerging domains. Current APIs are foreseen to play a critical role in increasing developer participation to speed up the transition from research to scalable, high performance software.

Issues in the Typical NBI Implementations

- Building a collection of standardized interfaces analogous to BLAS or PETSc for NBI.
- Enable integration of ‘ascale-friendly’ backends in both custom programs and software packages.
- Reuse/share architecture-specific optimizations across the research and developer community.

With Proposed NBI API

A researcher wishing to work with custom physics and post-processing methods often ends up writing even the NBI code. API aims to guarantee high performance for the n-body engine, thus reducing development overhead.

Schedule abstraction enables quicker integration as well as implementations that optimize for method and hardware. For the end-user, it aims to allow easy deployment of community built kernels in their custom MD programs.