MetaMorph: A Modular Library for Democratizing the Acceleration of Parallel Computing across Heterogeneous Devices

Paul Sathre, Wu Feng
{sath6220, feng} @cs.vt.edu
Virginia Tech --- Center for Synergistic Environments for Experimental Computing

SC|14: ACM/IEEE International Conference on High-Performance Computing, Networking, Storage, and Analysis, New Orleans, LA, Nov. 2014.

Introduction

Heterogeneity is becoming a fact of life in HPC, largely driven by demands for increased parallelism and power efficiency over what traditional CPUs can provide.

However, extracting the full performance of heterogeneous systems is non-trivial and requires architecture expertise.

Retrofitting existing codes for heterogeneity is tedious and error-prone, architecture experts are in short supply, and accelerators are moving targets.

Therefore, a single API for transparently executing optimized code on accelerators with minimal intervention is needed for scientific productivity.

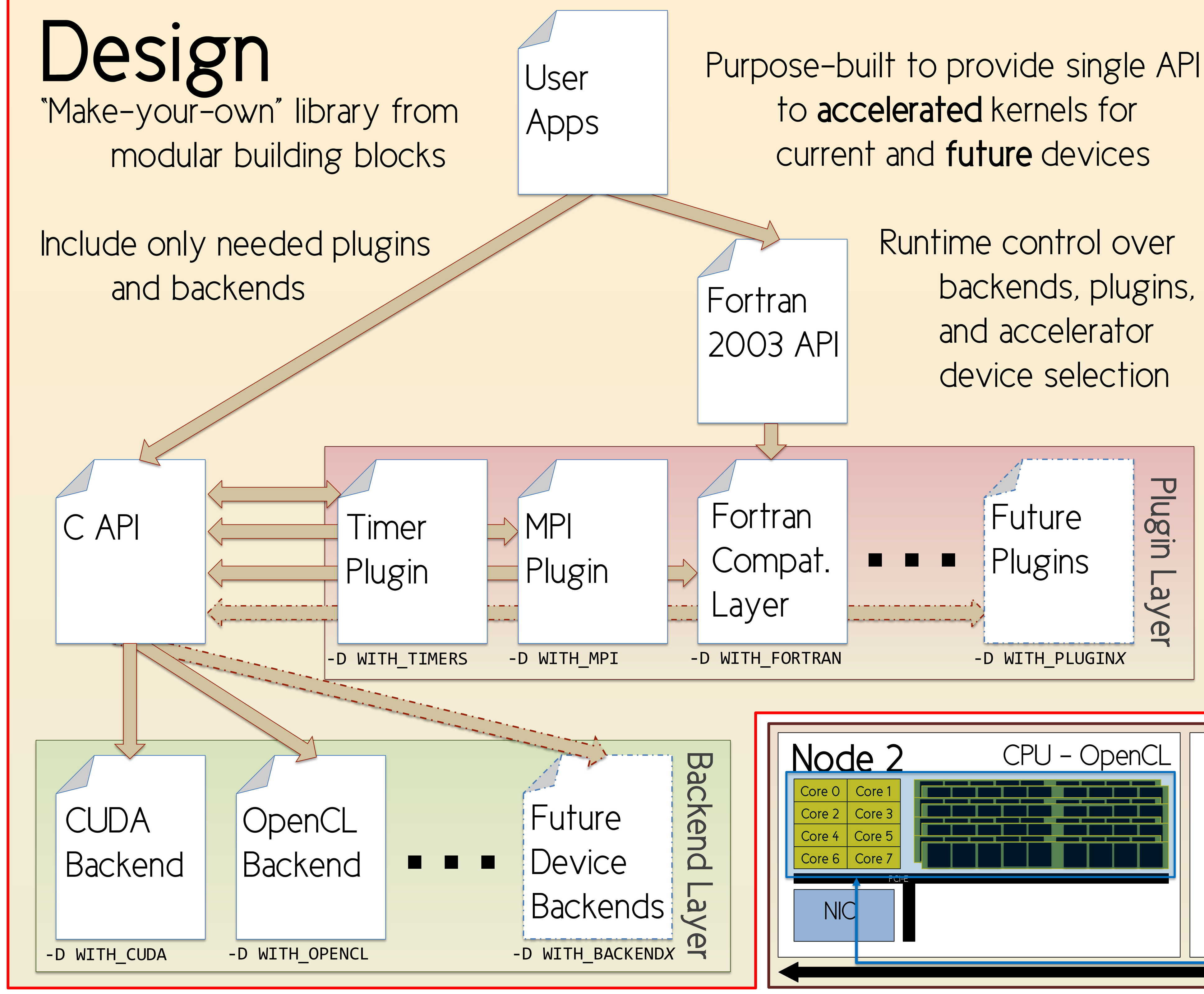
Design

"Make-your-own" library from modular building blocks

Purpose-built to provide single API to **accelerated** kernels for current and **future** devices

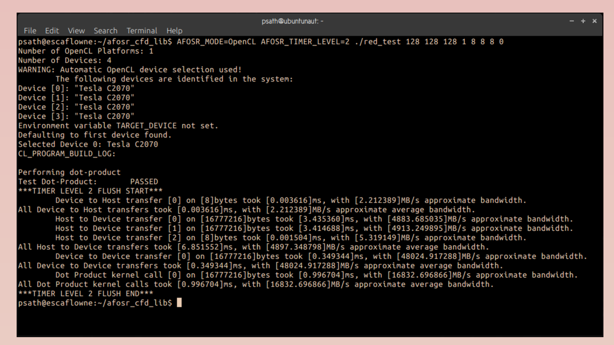
Include only needed plugins and backends

Runtime control over backends, plugins, and accelerator device selection



Plugins

Automatic Profiling



If compiled in, all kernels and transfers are timed behind the scenes automatically

Environment variable controls verbosity

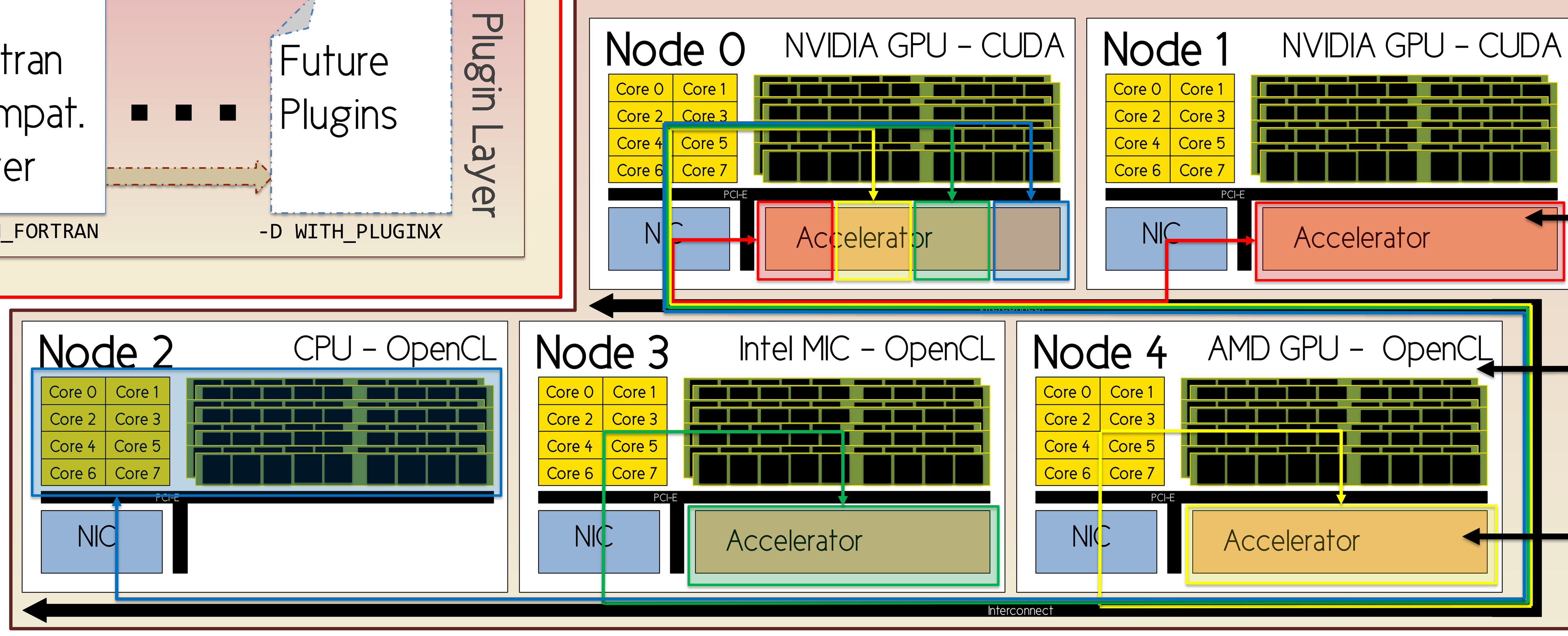
Fortran Compatibility

Verbose pass-by-reference C API, usable with ISO_C_BINDINGS

Fortran 2003 wrapper to verbose API provides simplified, unified calling convention for supported real and integer types

Uses C API internally, so all runtime control variables work equivalently

MPI



On-device packing of ghost regions

GPU Direct option, if available with MPI/ backend

Fallback to host-staged transfers otherwise

Transparently exchange device buffers between processes, *regardless of backend*

Accelerated Backends

The heavy-lifters of the library, selected at runtime by a "mode" environment variable from those included at compile-time

Include implementations of **all** C API-supported kernels for a single accelerator model

Standalone libraries in their own right can be used and distributed separately from the top-level API, as long as they are API compliant, *supporting community development of closed- or open-source alternatives*

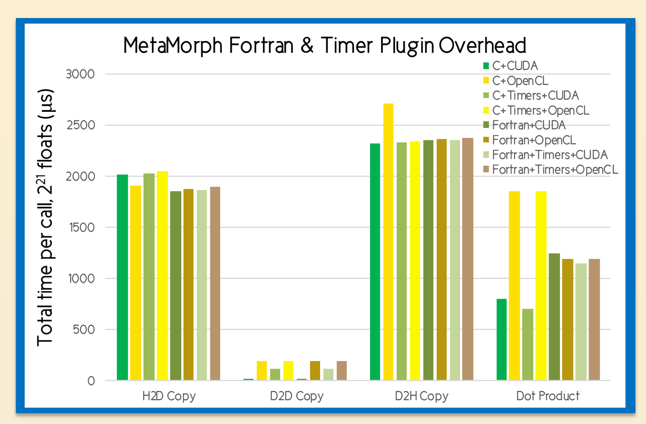
Currently support both CUDA and OpenCL, providing access to the most popular accelerators

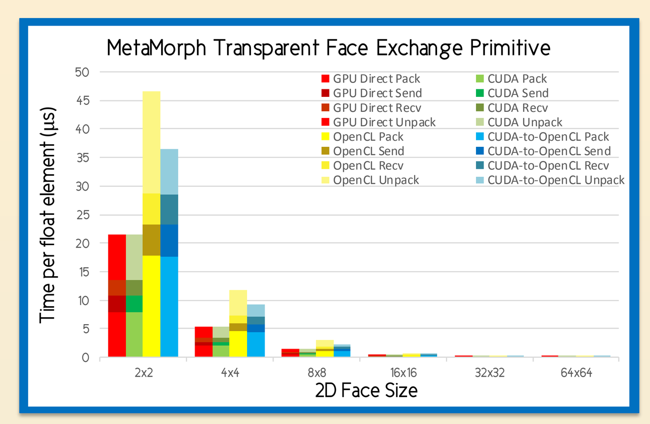
Currently support simple operations on subsets of 3D dense matrices: *reduction-sum, dot-product, 2D transpose, pack/unpack of subregions*

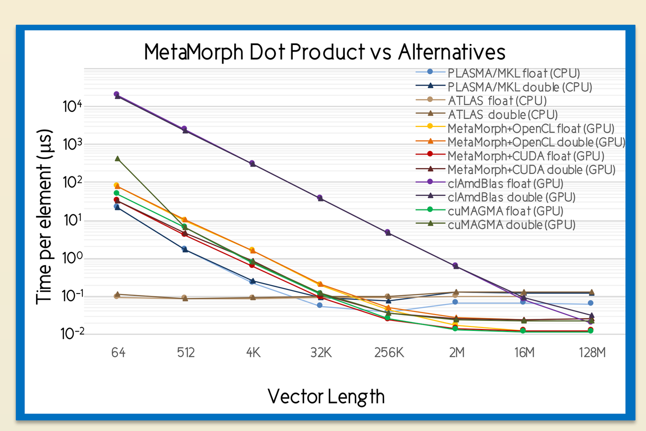
More kernels from computational fluid dynamics in the pipeline; extensions to other domains to follow, simply a matter of adding necessary kernels

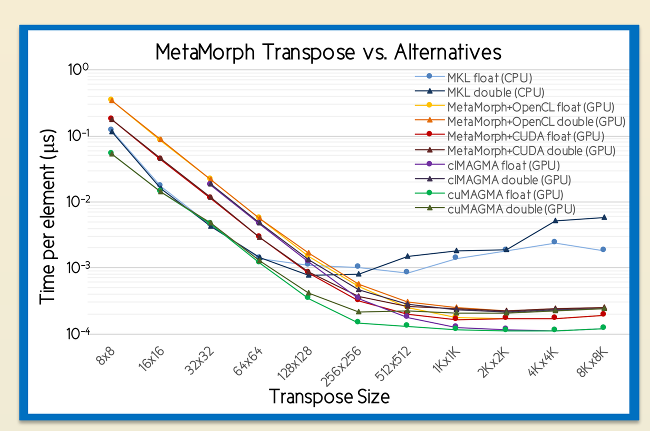
Performance

All tests performed in-node on a single system containing: 2x Intel Xeon X5550 Quad-core CPUs, 20GB RAM, and 4x Tesla C2070 GPUs









Related Efforts

Solver Frameworks

OpenFOAM [1]

Pros: Support for useful pre- and post-processing (mesh generation and visualization); many solvers for many domains

Cons: No internal accelerator support; framework-centric development; cumbersome API and "case" construction

PARALUTION [2]

Pros: Many matrix storage formats; many solvers; many preconditioners; support for OpenMP, CUDA, and OpenCL on CPUs/GPUs and MIC; plugins for Fortran and OpenFOAM

Cons: Framework-centric development; interop. with existing code low; no MPI support (yet), asynchronous operations only on CUDA; lack of non-destructive copy to/from C arrays

Solver Libraries

MAGMA [3]

Pros: Full BLAS and LAPACK support for CUDA, OpenCL, and MIC; support for several factorizations and eigenvalue problems; smart scheduling of hybrid CPU/GPU algorithms with QUARK directed acyclic graph scheduler; Multi-GPU methods

Cons: CUDA, OpenCL, and MIC variants are separate implementations; no internal MPI support; MKL/ACML dependency poorly documented and cumbersome

Trilinos [4]

Pros: Massive set of capability areas beyond linear algebra, solvers, and meshes; built-in distributed memory support; some preliminary CUDA/MIC work (e.g. Kokkos, Phalanx, Tpetra packages)

Cons: Redundancies of capability between packages; breadth of packages difficult to navigate for newcomers

Future Work

Continue expanding the API's provided set of kernels and backends with other primitive operations underlying fluid simulations. i.e. *Krylov solvers, stencil computations,* and *various preconditioners*

Generalize operations to work on non-3D data, and add primitives for computations on unstructured grids

Generate a third automatically runtime-scheduled backend to transparently execute code across entire node, a la *CoreTSAR* [7].

VirginiaTech Invent the Future

This work was funded in part by the Air Force Office of Scientific Research (AFOSR) Basic Research Initiative from the Computational Mathematics Program via Grant No. FA9550-12-1-0442.

[1] H. Jasak, A. Jemcov, and Z. Tukovic, "OpenFOAM: A C++ library for complex physics simulations."
[2] D. Lukarski, "PARALUTION project v0.7.0," 2012.
[3] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, and I. Yamazaki, "Accelerating Numerical Dense Linear Algebra Calculations with GPUs," in *Numerical Computations with GPUs.* Springer, 2014, pp. 3–28.
[4] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, "An overview of the Trilinos project," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, 2005.
[5] AMD. clMath (formerly APPML). Accessed 2014.10.17. [Online]. Available: http://developer.amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-math-libraries/
[6] D. K. Panda. MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RoCE. Network-Based Computing Laboratory, The Ohio State University. Accessed 2014.10.17. [Online]. Available: http://mvapich.cse.ohio-state.edu/
[7] T. Scogland, W.-c. Feng, B. Rountree, and B. de Supinski, "CoreTSAR: Adaptive Worksharing for Heterogeneous Systems," in *Supercomputing*, ser. Lecture Notes in Computer Science, J. Kunkel, T. Ludwig, and H. Meuer, Eds. Springer International Publishing, 2014, vol. 8488, pp. 172–186.

SyNeRG
http://synergy.cs.vt.edu/