BLAST Motivated Small Dense Linear Algebra Library Comparison

University of Colorado Boulder

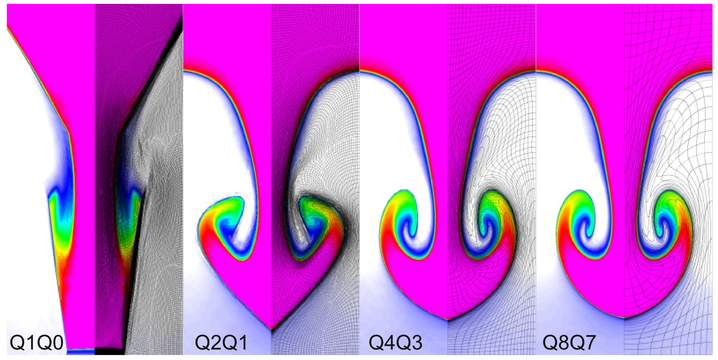
Pate Motter, Ian Karlin, Christopher Earl
pate.motter@colorado.edu, karlin1@llnl.gov, earl2@llnl.gov

Lawrence Livermore National Laboratory

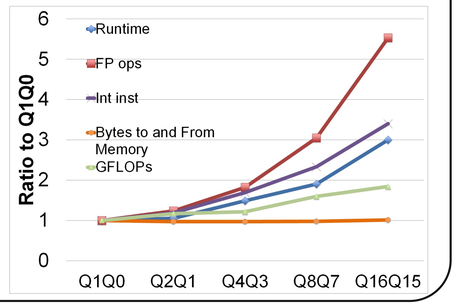
Abstract

The goal of the SHOCX project is to develop a new multiphysics coupled radiation-hydro code based on arbitrary-order finite elements. BLAST is the arbitrary Lagrangian-Eulerian (ALE) portion of this project and its finite-element formulation requires both global sparse and local dense matrix solves. The local dense matrix solves and sparse matrix assembly perform numerous calculations on small, dense matrices. Currently these operations are handled by MFEM, but using third party libraries such as Eigen, Armadillo, and Blaze might be more productive and faster. Using a benchmark we created to mimic the most expensive portions of BLAST, we explore the performance and maintainability of these libraries compared to MFEM and a version of MFEM optimized via compile-time templates. Our results show that for our benchmarks Armadillo, Blaze, and templated MFEM all show promising results. Eigen was initially promising with its available features, but was not performant enough to use it in BLAST.

BLAST and its motivation:

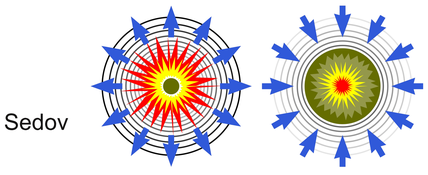


- Computing trends are increasing FLOPs faster than data motion capabilities
- BLAST is an arbitrary order, ALE finite element hydro code with curvilinear zones
- Higher orders result in:
  - Fixed amount of data motion per DoF
  - Greater computational intensity
  - More accurate answers
  - Fewer remap (Eulerian) steps

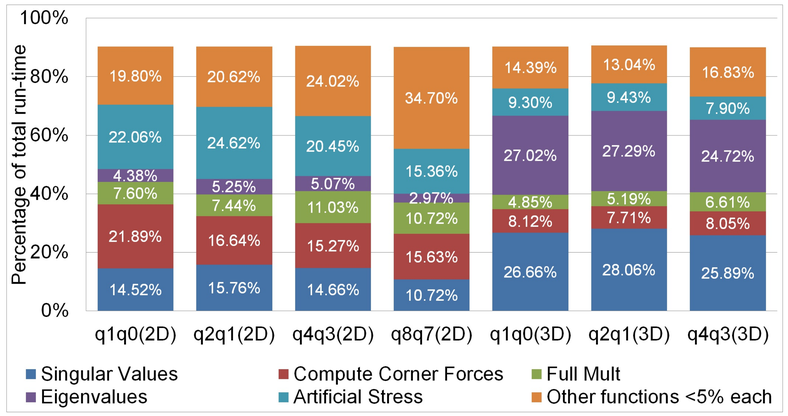


How we performed our testing:

- All tests were performed on LLNL's Cab cluster
- CPUs: Xeon Sandy Bridge
- Compilers: Intel v. 14.03
- Timings are the total execution time over 100M iterations of each benchmark
- The Noh and Sedov example problems were used as representative of the upper and lower bounds of mesh activity



Where Does BLAST Spend its Runtime?



- ~90% of the run-time is accounted for in ~12 functions (~25 loops)
- Most of the FLOPs occur in small dense linear algebra routines located within the MFEM finite element code at each quadrature point

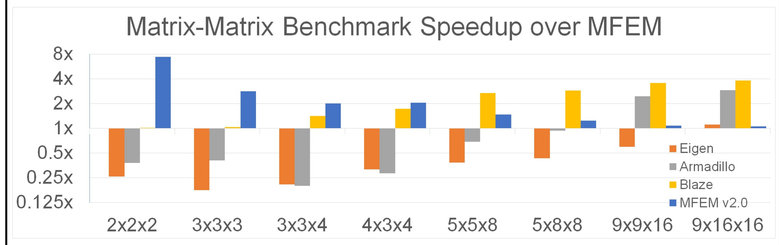
Determining a Suitable Library for BLAST's Needs

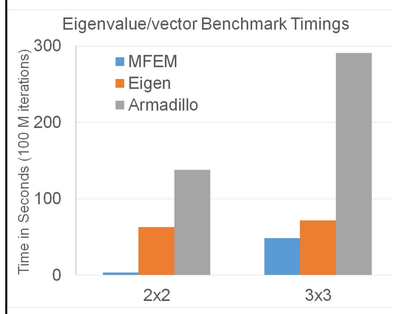
- Optimized for small, dense, fixed-size matrices
- Active development
- Easy to use

- No copying of data to new data structures
- Minimal re-coding
- Matrix decompositions
- Few if any additional dependencies
- Support for tensors
- Active community
- Easy to extend

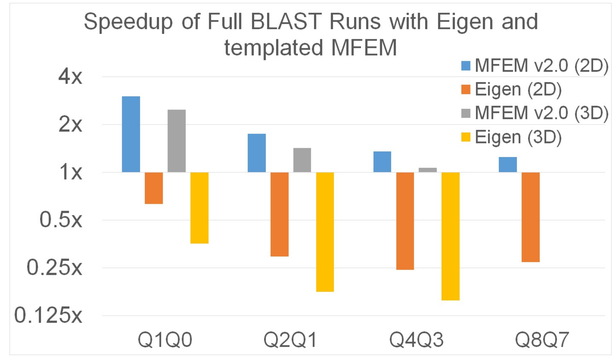
	Pros	Cons
LAPACK	- Well established - Large linear algebra function coverage	- Not optimized for small matrices
MAGMA	- Promising performance on GPUs	- BLAST is currently CPU only
Boost::uBlas		- Requires Boost - No active development
Elemental		- Focus on large, distributed matrices
Armadillo	- Matlab style syntax - Active development	- Small community
Blaze	- Very small footprint	- No support for eigen or singular values
Eigen	- Large user community - Active development	- Tensor class unsupported
MFEM	- Currently used within BLAST	- Primarily used only by LLNL

Comparing the potential libraries





- The synthetic benchmarks mimic the BLAST kernels taking a majority of the run-time
- Presented results are representative of a larger suite of tests
- Test sizes are representative of BLAST use cases
- Eigen's performance is not competitive
- Blaze and Armadillo show promise but need further testing



- Eigen is 1.5x - 4x slower than BLAST with MFEM
- MFEM v2.0 is 1x - 3x faster than BLAST with MFEM
- MFEM v2.0 produces its largest gains for low orders and 2D problems

Conclusions

- Most of BLAST's time is spent in small dense linear algebra routines
- Blaze, Armadillo and the templated implementation of MFEM are all promising.
- Many linear algebra libraries exist, but few do what BLAST specifically needs.

Future Work

- Expand the scale of and scope of these benchmarks
- Re-write important kernels of BLAST using Armadillo and Blaze
- Perform same tests on other architectures

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

LLNL-POST-675784