Sandia National Laboratories

Fugaku and A64FX Update

APRIL 2021

PRESENTED BY

SI HAMMOND, MATTHEW CURRY, KEVIN DAVIS, VINH QUANG DANG, OKSANA GUBA, ROB HOEKSTRA, JIM LAROS , KEVIN PEDRETTI, DAVID POLIAKOFF, SIVA RAJAMANICKAM, CHRISTIAN TROTT, LUC VERGIAT-BERGER AND ANDREW YOUNGE

Unclassified Unlimited Release

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND20XX-XXXX P

U.S. DEPARTMENT OF ENERGY

NNSA

INTRODUCTION

- Been a busy few months with our A64FX/Fugaku enablement work


Improving CMake to support the Fujitsu compiler


First builds of Kokkos and extending Kokkos SIMD to support A64FX SVE operations


First builds of Trilinos to support NALU-CFD (Sandia CFD application)


Developing ATSE Software Stack for A64FX


Containers for ATSE on A64FX Nodes (using podman)


Supercontainers scaling study on Fugaku (ECP, in progress)


LAMMPS and SPARTA Runs in coming year (in progress)

Thank you to RIKEN for the collaboration and support

INOUYE A64FX TESTBED

- Inouye A64FX testbed is part of the ASC Advanced Architecture Testbed project at Sandia
  - Open to researchers across the NNSA labs (and some external partners)
- Provides Fujitsu, Arm and GCC compiler toolchains
  - Still working on installing the Cray/HPE compiler toolchain locally
- Integrated into Sandia's continuous integration build and test environment
  - Critical for Trilinos, Kokkos and several application ports to check code changes don't break compatibility
  - Runs sequence of tests overnight and throughout the day
- Helped to prepare our codes for Fugaku and to support SVE and ATSE development activities

CMAKE ISSUES WITH FUJITSU A64FX COMPILER

- Early experimentation with Sandia's A64FX Inouye platform showed some issues with correct detection of Fujitsu compiler when using Cmake
  - Changes detection of correct C++ flags and OpenMP
  - Problematic for Trilinos and Kokkos (without workarounds)

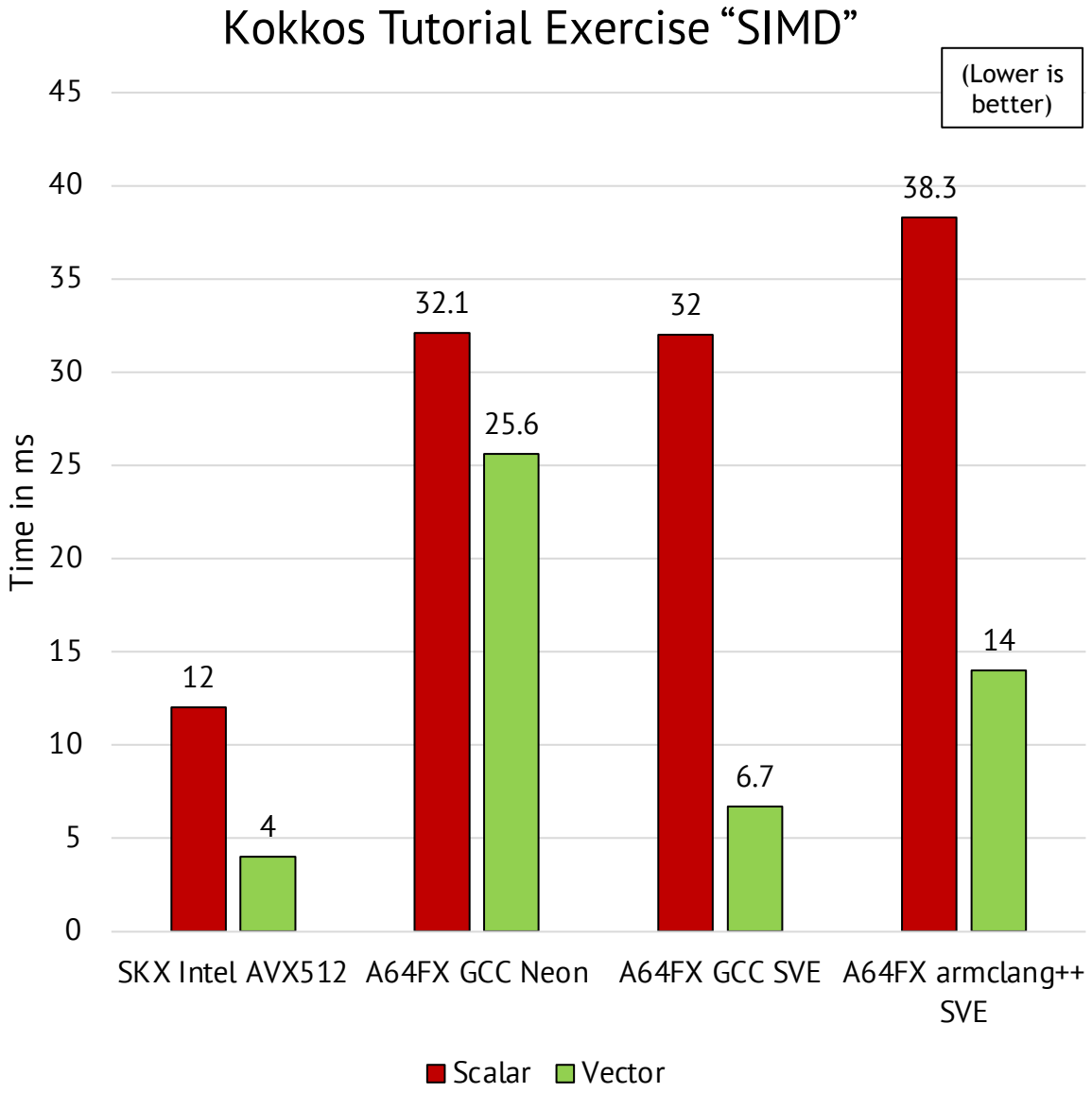
https://cmake.org/

- Engaged with Chuck Atkins at KitWare to fix this issue
  - Used Sandia's Inouye testbed to develop changes/fixes
  - https://gitlab.kitware.com/chuck.atkins/cmake/-/tree/fujitsu-compiler-support

- Will be integrated into a future CMake release

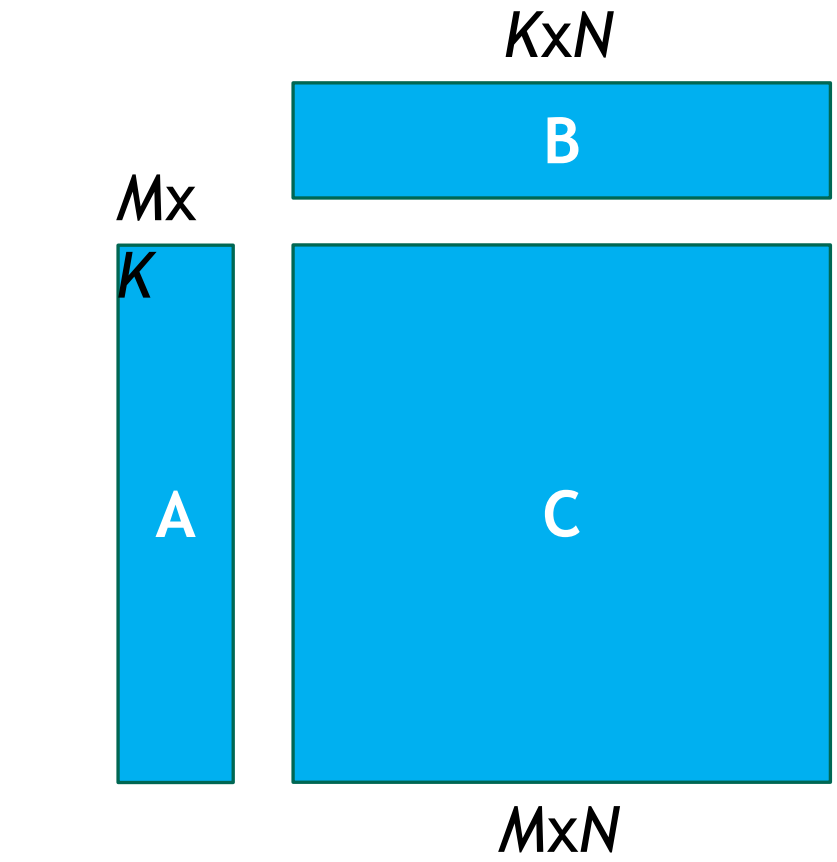
KOKKOS CORE SUPPORT FOR A64FX

- Added Fujitsu compiler as a supported compiler Kokkos version 3.3
  - Challenging due to CMake and compiler detection issues (now fixed)
- Added sve-vector-bits flag (for GCC) in Kokkos version 3.4
  - Improves performance in a number of our internal micro-benchmarks and reduces code sequence length
- Added support for A64FX SVE in the Kokkos SIMD library
  - Clang and GCC support ”vector_size” attribute
  - Fujitsu compiler utilizes “omp simd”
  - Significant improvement over using Neon instructions



KOKKOS KERNELS - ADELUS DENSE LU SOLVER

- ADELUS – performance portable dense LU solver for next-generation distributed-memory HPC platforms
  - Uses partial pivoting LU factorization for double real/complex dense linear systems using MPI
  - Leverages Kokkos and Kokkos Kernels for performance portability
- Key requirement in each MPI process is the use of BLAS functionalities for local matrices (e.g. GEMM for updating the matrix)
- Preliminary evaluated multi-threaded GEMM on a single node
- Comparisons:
  - Lassen – ESSL 6.2.1 on a 44-core POWER9 CPU (GCC 8.3.1)
  - Fugaku - ArmPL 20.1 on a 48-core A64FX (GCC 10.2)
  - Fugaku – SSL2 on a 48-core A64FX CPU (FCC 4.5.0)

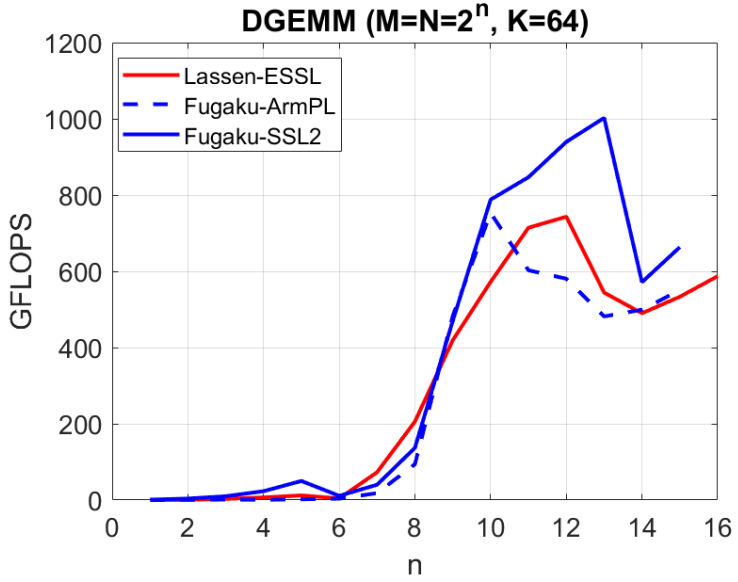


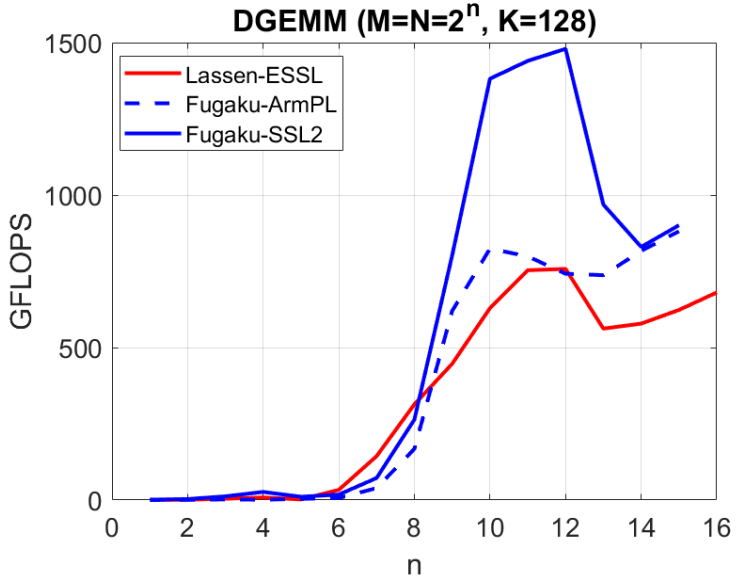
M=*N*
K=block_size parameter used in delay-updating

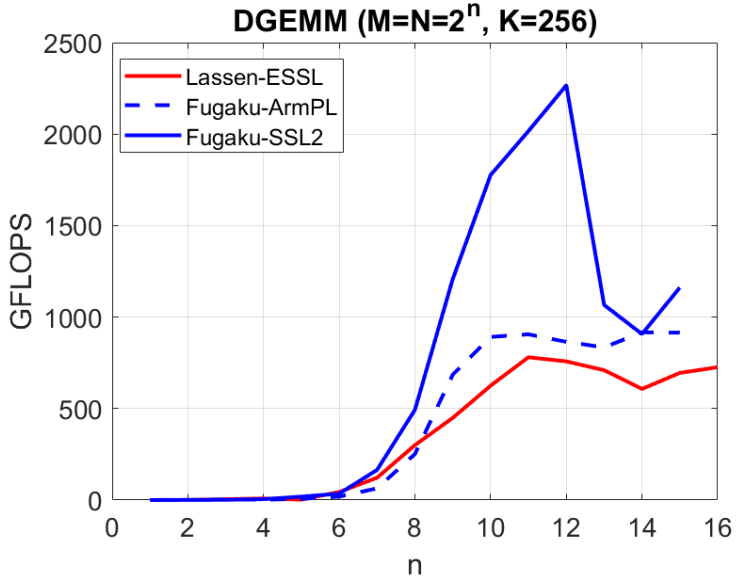
Availability: https://github.com/trilinos/Trilinos/tree/master/packages/adelus

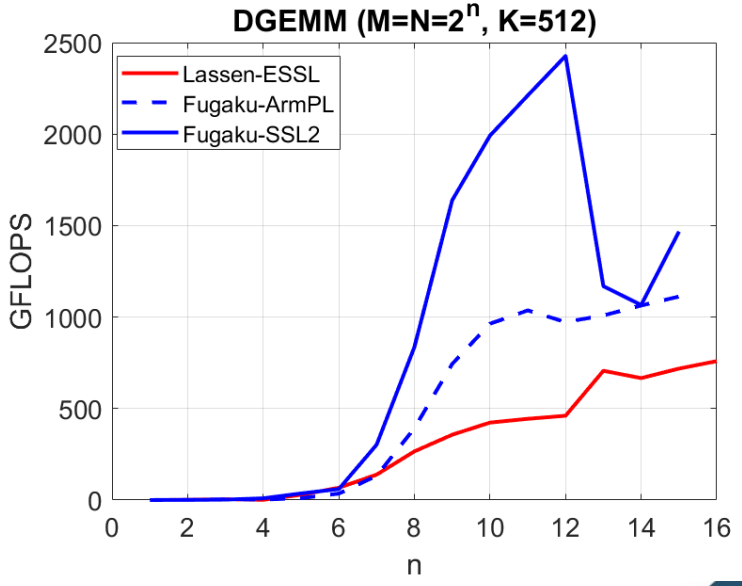
ADELUS – DGEMM PERFORMANCE COMPARISON

- K is chose similarly to typical block sizes in ADELUS (K = {64, 128, 256, 512})
- M(=N) varied: 2, 4, 8, 16, .. 16384, 32768, 65536
  - ArmPL and SSL2 cannot handle M=N=65536 due to 32GB memory limit on Fugaku
- Both ArmPL and SSL2 outperform ESSL with double precision for large matrix sizes
- SSL2 provides best performance among the three libraries benchmarked



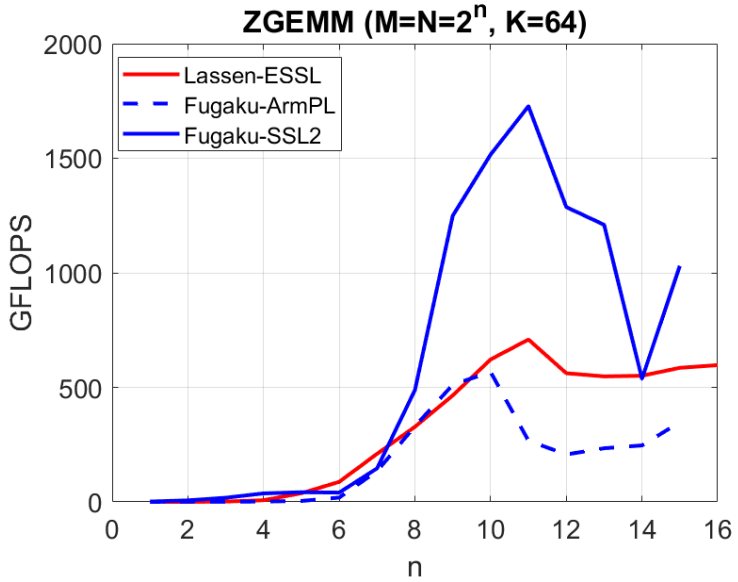


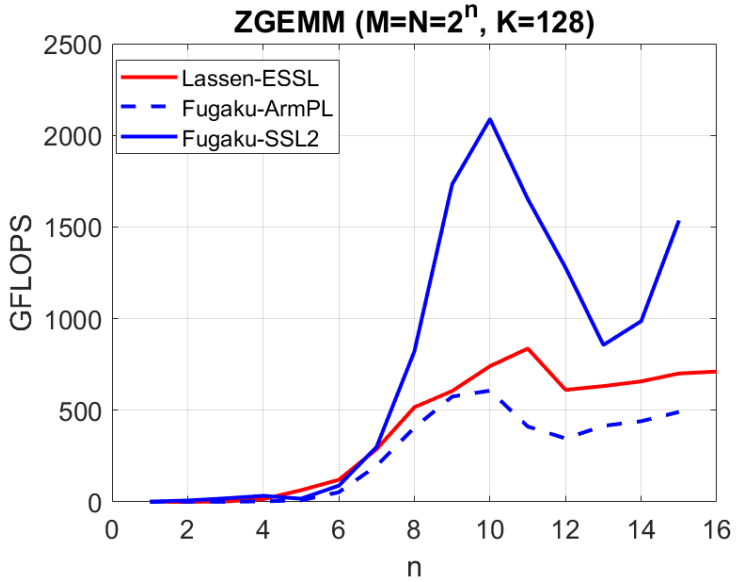


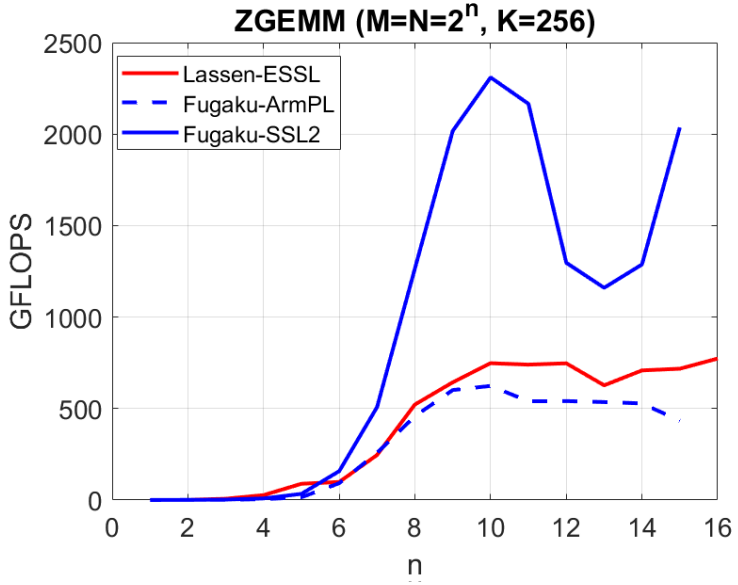


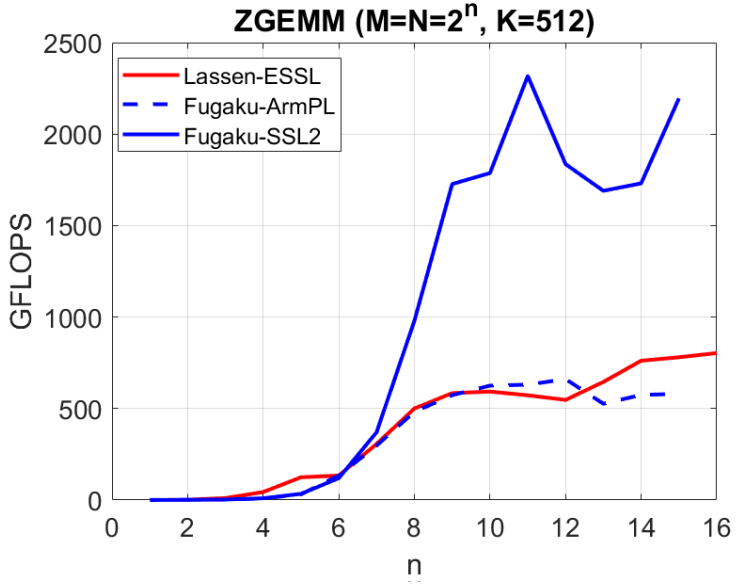
ADELUS – ZGEMM PERFORMANCE COMPARISON

- K is chosen similarly to typical block sizes in ADELUS (K = {64, 128, 256, 512})
- M(=N) varied: 2, 4, 8, 16, .. 16384, 32768, 65536
  - ArmPL and SSL2 cannot handle M=N=65536 due to 32GB memory limit on Fugaku
- Both ArmPL does not perform well for complex double precision benchmarks
- SSL2 again provides the best performance among the three libraries benchmarked









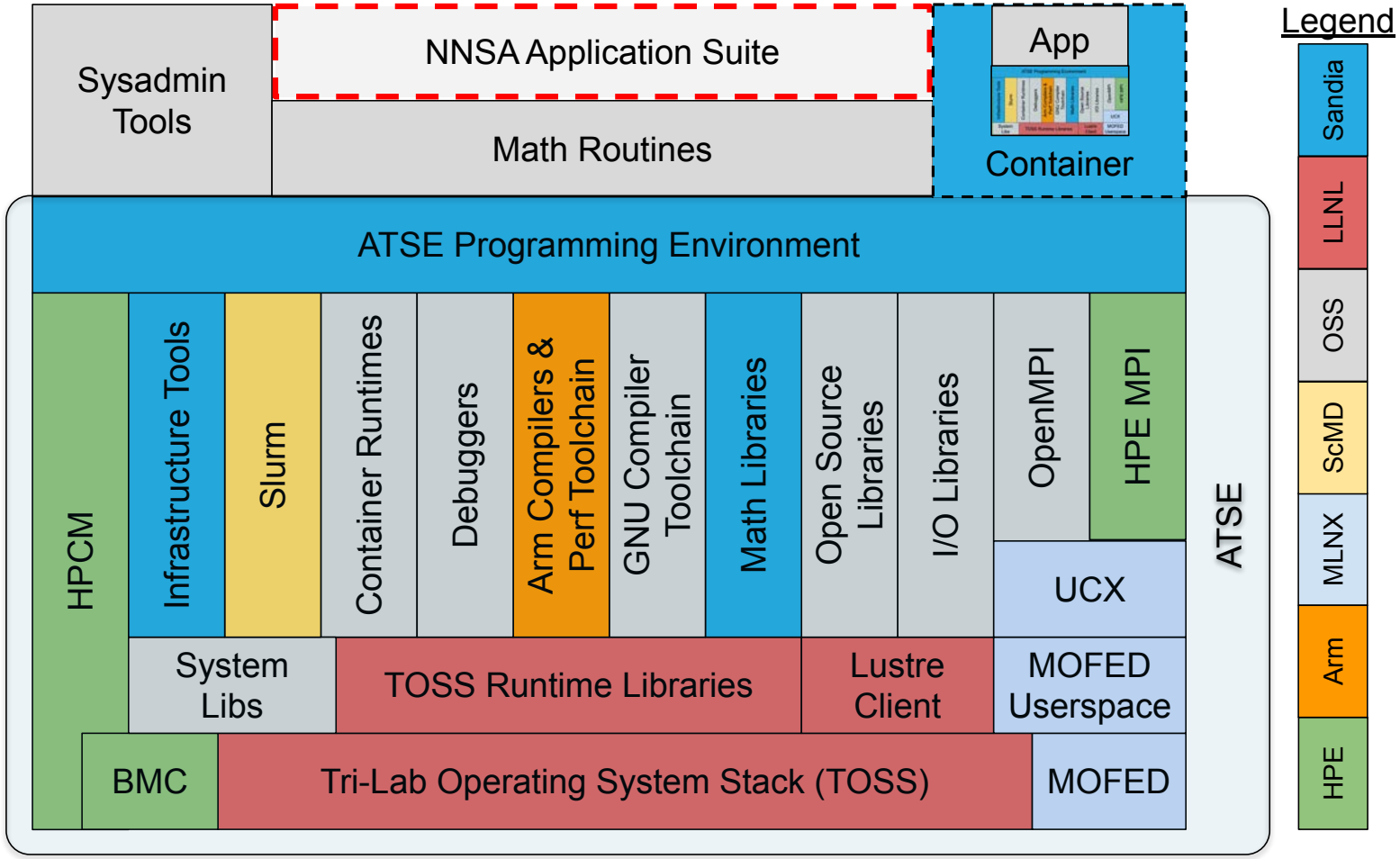
TRILINOS BUILDS ON A64FX

- Early testing of Trilinos builds on A64FX are now complete
  - Using configurations for NALU-CFD (similar to NALU-ExaWind, but optimized for NNSA use cases)
  - Builds full solver stack with multiple packages
- Builds with Kokkos using OpenMP backend
  - GCC, Arm and initial builds with Fujitsu compiler
  - Several challenges with the Fujitsu compiler (test failures)
  - Utilizes Kokkos A64FX configuration parameters to generate build configuration
- Regular continuous integration testing for main components at Sandia
  - Helps to reduce bugs

https://trilinos.github.io/

ATSE: ADVANCED TRI-LAB SOFTWARE ENVIRONMENT

- ATSE is a collaboration with HPE, OpenHPC, and ARM
- Many pieces to the software stack puzzle
- HPE's HPC Software Stack
  - HPE Cluster Manager
  - HPE MPI (+ XPMEM)
- Arm
  - Arm HPC Compilers
  - Arm Math Libraries
  - Arm Allinea Tools
- Open source tools and libs
  - Slurm, OpenMPI, TPL stack, etc.
  - OpenHPC and Spack versions
- Mellanox-OFED & HPC-X
- RedHat 7.x for aarch64 – TOSS



ATSE CONTAINER USE CASES

- Test new releases prior to roll out (Sysadmins)
  - Containers available for ATSE 1.2.0, 1.2.1, 1.2.2, 1.2.3, 1.2.4, and 1.2.5 (current)
  - Test new stack at scale against real apps, identify issues early
- Full-stack rewind and fast forward (End-users and Sysadmins)
  - Debug issues
  - Try out newer software ahead of roll out
- Off-platform Build Environment
  - Replicate a near exact environment that users can run elsewhere, saving platform cycles
  - Piloting now at Sandia for Sierra code suite, using the Astra container for off-platform build and test
- Consistent Look and Feel Across HPC Platforms
  - ATSE container available for Arm and x86_64 (new); creating optimized A64FX container for SNL Inouye testbed
  - Common set of TPLs optimized for the target environment (e.g., compilers and MPI)

ATSE CONTAINER PODMAN AND SINGULARITY EXAMPLES

Podman Example

```
# Login and pull an atse container image from the upstream container registry
$ podman pull containers.sandia.gov/atse/atse-container/atse-arm-openmpi4-astra:1.2.5

# Run an interactive shell
$ podman run -ti --rm -v /home/${USER}:/home/${USER} \
       containers.sandia.gov/atse/atse-container/atse-arm-openmpi4-astra:1.2.5
```

Singularity Example

```
# Import an atse container image and flatten to a flattened singularity image file
$ singularity build atse-1.2.5.simg \
      docker://containers.sandia.gov/atse/atse-container/atse-arm-openmpi4-astra:1.2.5

# Run an interactive shell
$ singularity shell atse-1.2.5.simg
```

PLANNED SUPERCONTAINER SCALING STUDIES

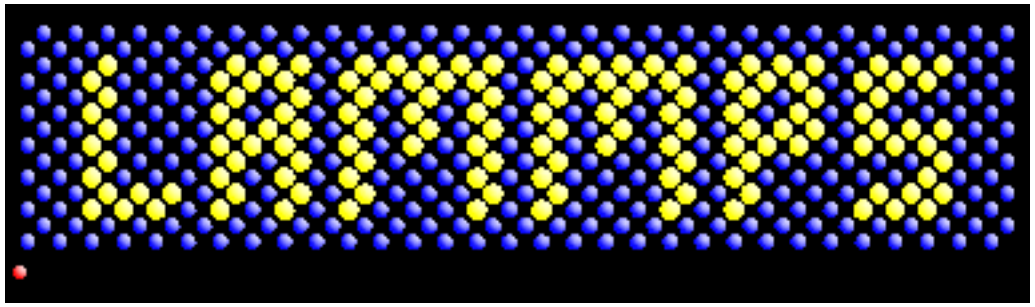
- Interested in new configurations for Singularity and Charliecloud container runtimes
  - Configurations could have significant impact on performance at extreme scale
  - Newer modifications & configs untested beyond 2000 nodes
- Need to investigate container runtime configurations at scale = Fugaku
  - Initial testing with Inouye A64FX system at Sandia
  - Once configurations validated on Inouye, reproduce on Fugaku
  - Leverage ATSE when possible for builds
- Evaluate overhead of Singularity Image Format (SIF) vs “sandbox” images at scale
- Evaluate overhead of container startup via SUID vs user namespaces at scale
  - May require administrative help
  - Hope to leverage existing Singularity for Fugaku via Pacific Teck & Sylabs
- Planned usage of containerized HPCG and other open mini-apps

More info: https://www.exascaleproject.org/accelerating-the-adoption-of-container-technologies-for-exascale-computing/

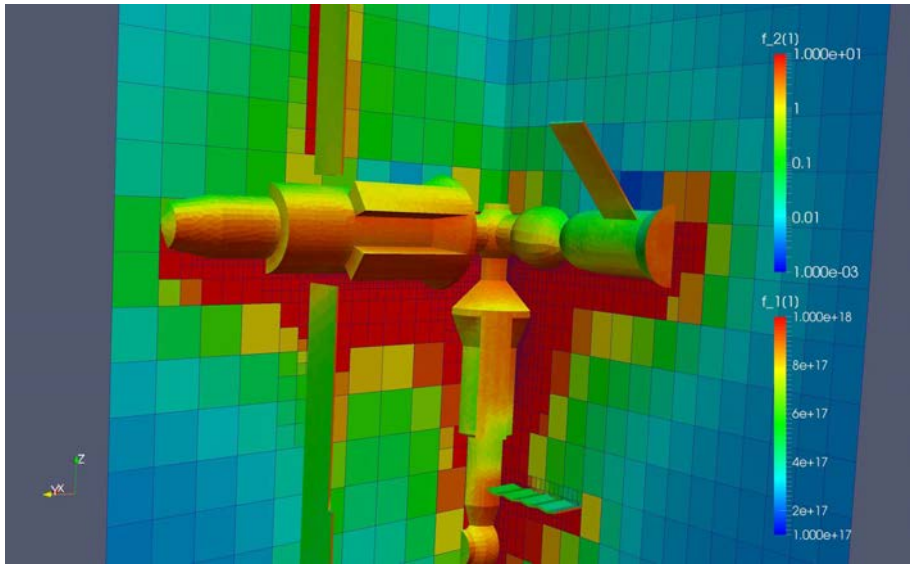
PLANS FOR THE COMING YEAR

- Plans to study large-scale runs of LAMMPS and SPARTA on Fugaku
  - Utilizes Kokkos improvements for performance
  - Builds in local testing using Inouye – basic initial runs are looking good

- Teams have started to identify the problems/inputs for these runs



https://lammps.sandia.gov



https://sparta.sandia.gov


Sandia
National
Laboratories