LA-UR-19-27092

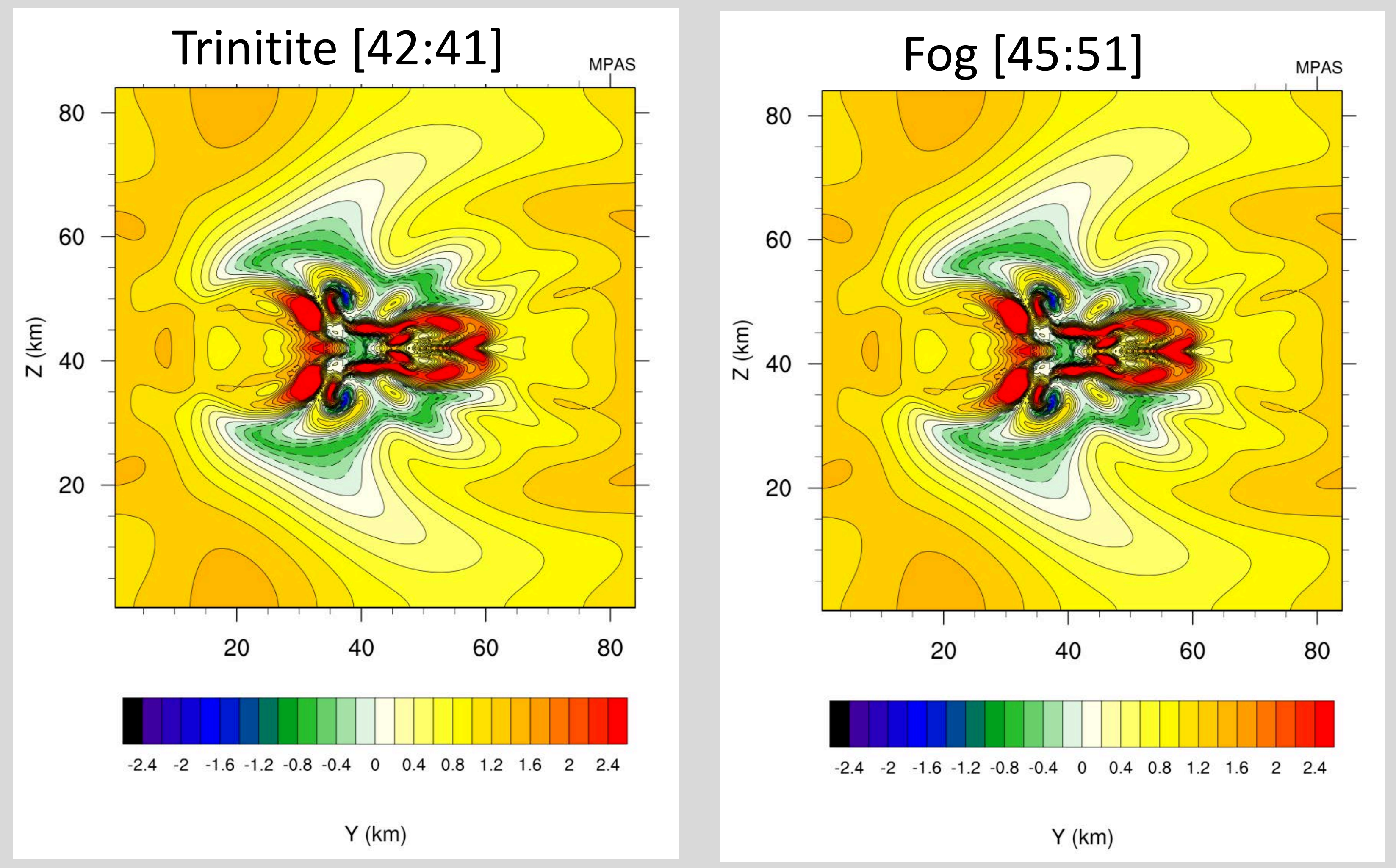
Building Complex Software Applications Inside of a Container

Calvin Seamons
Los Alamos National Laboratory
calvindseamons@lanl.gov

Abstract

High performance computing (HPC) scientific applications require complex dependencies, many of which are not supplied by the Linux operating system. Typically, HPC centers offer these dependencies through environment module-files that when loaded, modify the user environment to provide access to software installations. If a package doesn't exist on a system, customers must request them through system administrators or find alternatives. It is unrealistic for HPC centers to provide every unique dependency requested, thus the interest for user defined software stacks and containers are increasing. By building *Model for Prediction Across Scale (MPAS)* and its dependencies inside a Debian GNU/Linux 9 container image, we demonstrate that a common atmospheric simulation runs nearly identically on a Red Hat Enterprise Linux 7 Commodity Technology System (CTS1) cluster with the Intel® Core Broadwell™ architecture and a Cray System with the SuSE Enterprise Linux 12 + Cray Linux Environment (CLEv6.0) and an Intel® Core Haswell™ architecture, with no modifications to the container. This shows that it is possible to build complex software applications inside an unprivileged container and run it successfully across various super computers with different hardware components, Linux operating systems, and environments. Furthermore, the application computational results from their execution are essentially identical with 28bytes differing between 2.1GB output file. Containers offer customers versatility with nominal dependencies on the system they run on. This advancement potentially allows for tremendous portability across Linux HPC systems; by encapsulating complex dependencies it gives scientists the ability to run large scale simulations on HPC resources with their own preferred software.

Results of containerized MPAS



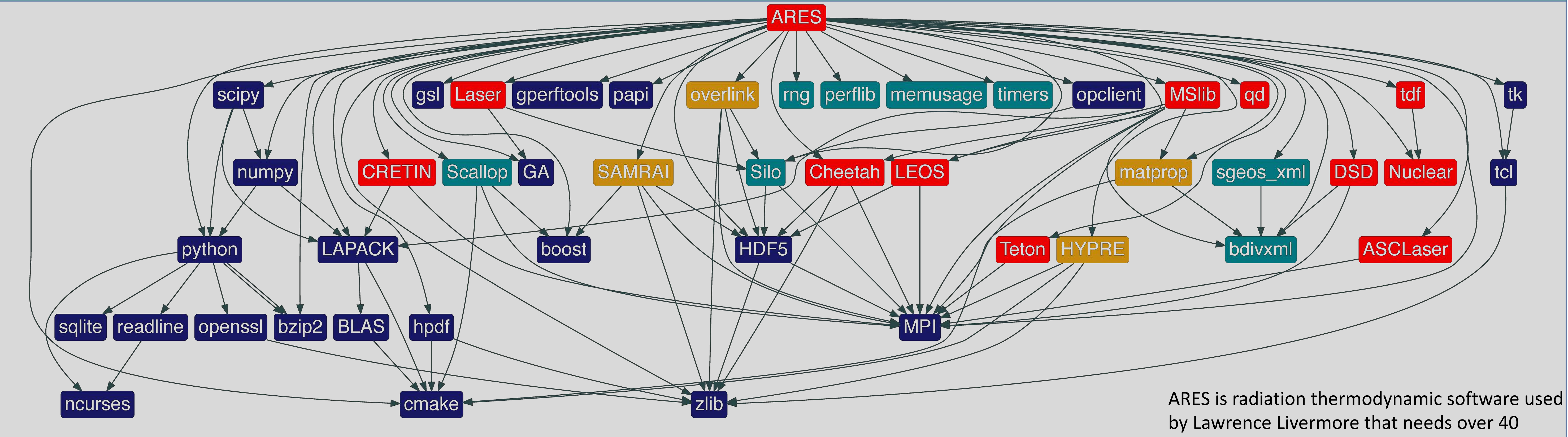
Both Fog and Trinitite produce the same result with **zero** modification to the container.

- Trinitite runs roughly 7% faster
- Linux Diff command show 28bytes differ on the output of 2.1GB output file

Dockerfile.MPAS

```
#Model for Prediction Across Scales (MPAS) "containerized" using docker.
#Author: Calvin Seamons
#Dockerfile being built using openmpi build from charliecloud team.
FROM openmpi

#apt update and installation of tools, compilers, and libraries needed.
RUN apt -y update
RUN apt -y install build-essential
RUN apt -y install wget curl git

#Creating directory for tarballs.
RUN mkdir -p /home/tars

#Entering tar directory for downloading.
WORKDIR /home/tars/

#Downloading tarballs from website.
RUN wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.16/src/hdf5-1.8.16.tar.gz  \
               https://parallel-netcdf.github.io/Release/pnetcdf-1.11.2.tar.gz                        \
               https://www.unidata.ucar.edu/downloads/netcdf/ftp/netcdf-c-4.7.0.tar.gz                \
               https://github.com/NCAR/ParallelIO/releases/download/pio2_4_3/pio-2.4.3.tar.gz         \
               https://www.unidata.ucar.edu/downloads/netcdf/ftp/netcdf-fortran-4.4.5.tar.gz          \
               https://github.com/MPAS-Dev/MPAS-Model/archive/v7.0.tar.gz                             \
                http://www2.mmm.ucar.edu/projects/mpas/test_cases/v7.0/supercell.tar.gz               \

#Untaring all tarballs into /usr/local/src.
WORKDIR /usr/local/src/
RUN  cat /home/tars/*.tar.gz | tar -xzf - -i

#Installing requirements for Python3.7
RUN apt -y install zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev
RUN apt -y install python

#Installing Spack.
WORKDIR /usr/local/src
RUN git clone https://github.com/spack/spack.git
WORKDIR /usr/local/src/spack
RUN git checkout v0.12.1
#RUN "." source share/spack/setup-env.sh

RUN ./bin/spack install zlib@1.2.11
RUN ./bin/spack install gcc@8.2.0

#TODO edit spack to allow dir specification and not use hash.
ENV PATH=/usr/local/src/spack/opt/spack/linux-centos7-x86_64/ \
     gcc-4.8.5/gcc-8.2.0-sxbf4jq6ghmoybsjlpqz2dm2qbbxzfyn/bin/:$PATH

#Installation of Parallel hdf5 (phdf5)
ENV LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/
WORKDIR /usr/local/src/hdf5-1.8.16
RUN CC=mpicc FC=mpif90 ./configure --with-zlib=/usr/local/zlib \
                    --disable-cxx          \
                    --enable-parallel      \
                    --enable-hl            \
                    --enable-fortran       \
                    --enable-fortran2003   \
                    --enable-direct-vfd    \
                    --enable-production    \
                    --enable-build-all     \
                    --enable-shared        \
                    --prefix=/usr/local
RUN make -j && make install -j

#Installation of PNetCDF
WORKDIR /usr/local/src/pnetcdf-1.11.2
RUN ./configure --enable-shared      \
          --prefix=/usr/local/  \
          --with-mpi=/usr/local/

RUN make check -j && make install -j

#Installation of NetCDF
Run apt -y install libcurl4-gnutls-dev
WORKDIR /usr/local/src/netcdf-c-4.7.0
RUN LDFLAGS='-L/usr/local/lib -lpnetcdf' \
   CPPFLAGS='-I/usr/local/include' ./configure --disable-dap          \
                              --disable-hdf4                          \
                              --enable-parallel-tests                 \
                              --enable-pnetcdf                        \
                              --enable-large-file-tests               \
                              --prefix=/usr/local/
RUN make -j && make install -j

#Installation of NetCDF-Fortran TODO may not be needed...
WORKDIR /usr/local/src/netcdf-fortran-4.4.5
RUN ./configure --enable-shared --prefix=/usr/local
RUN make -j && make install -j

#PIO building
WORKDIR /usr/local/src/pio-2.4.3
RUN CC=/usr/local/bin/mpicc   \
   FC=/usr/local/bin/mpif90  \
   CFLAGS=-std=c99 ./configure --enable-fortran --enable-shared

RUN make -j && make install -j

#MPAS Atmosphere Core Generated, all built core stored into mpas-cores
RUN mkdir /usr/local/src/mpas-cores

#Generation of init_atmosphere core
WORKDIR /usr/local/src/MPAS-Model-7.0
RUN PIO=/usr/local      \
   NETCDF=/usr/local    \
   PNETCDF=/usr/loca l \
   make gfortran CORE=init_atmosphere \
           USE_PIO2=true       \
           DEBUG=true          \
           PRECISION=single

#store and remove init_atmosphere core generation
RUN cp init_atmosphere_model ../mpas-cores
RUN make clean CORE=init_atmosphere

#Generation of atmosphere core.
RUN PIO=/usr/local      \
   NETCDF=/usr/local    \
   PNETCDF=/usr/local  \
   make gfortran CORE=atmosphere \
           USE_PIO2=true             \
           DEBUG=true                \
           PRECISION=single

#Store and remove atmosphere core generation
RUN cp atmosphere_model ../mpas-cores
RUN make clean CORE=atmosphere
```

ARES Dependency Trees



https://computing.llnl.gov/projects/spack-hpc-package-manager

ARES is radiation thermodynamic software used by Lawrence Livermore that needs over 40 dependencies

Running MPAS in a Container

1. Write a "dockerfile" that builds MPAS and it's dependencies
2. Create an Image
3. Flatten the Image into a tarball
4. Send the tarball to a production machine to execute

```
$ ch-grow –t mpas-debian9 –f Dockerfile.mpas .
$ ch-builder2tar mpas-debian9 /tmp
$ scp /tmp/mpas-debian9.tar.gz $user@fg-fey
```

5. Allocate nodes on a production cluster
6. Module load in desired packages (Charliecloud)
7. Unpack tarball across allocated nodes
8. Execute Charliecloud runtime environment

```
$ salloc –N1 --qos=standard --time=3:00:00

$ module load charliecloud

$ ch-tar2dir mpas-debian9.tar.gz /var/tmp

$ ch-run -w --unset-env='*' --cd=/usr/local/\
src/supercell/ --set-env=/var/tmp/mpas-debian9/\
ch/environment/var/tmp/mpas-debian9/ -- mpirun\
-np 32 ./atmosphere_model
```

Charliecloud development team is currently working on ch-grow being adapted for the front end. This would eliminate the transport of tarballs as images could be build and executed all on the production cluster.

Acknowledgements

Mentors: Jordan Ogas, Jennifer Green
Special Thanks: Dan Magee, Rob Aulwes, Kody Everson, and Trent Steen

Future Work

- Get MPAS container to work on multi-nodes
- Build other software applications inside containers
- Performance analysis
- GPU analysis and testing

Presented to Los Alamos National Laboratory HPC Mini-Showcase 2019