OAK RIDGE National Laboratory

Evaluating Integration and Performance of Containerized Climate Applications on GAEA

Subil Abraham
Ryan Prout
Matthew Davis

ORNL

Thomas Robinson
Christopher Blanton
Luis Sal-Bey

NOAA/GFDL

May 3, 2022

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Contents

- Background
- Evaluation
- Issues we ran into
- Future Work

BACKGROUND

The GAEA Supercomputer at ORNL

- ORNL manages the GAEA on behalf of NOAA.
  - Cray XC supercomputer, two clusters C3 (1496 nodes - peak 1.77 PF) and C4 (2656 nodes - peak 3.52 PF)
  - primarily used for climate simulations.
- Running important climate models like AM4 and ESM

[1]

The Need for Containers at NOAA

- NOAA GFDL's climate models must adhere to strict reproducibility guidelines and run at scale on GAEA.
- Containers useful for packaging up the model's software environment, remove dependence on host system's stack.
- Using Singularity since that works well in the HPC space without extra privileges, for running containers.
  - Can use Docker or Podman for building containers elsewhere and integrating into a CI pipeline

Challenges with Running Containers at Scale

- Climate models rely on MPI
  - Need to test hybrid and bind modes with given containers to see which work best on GAEA.
    - hybrid mode – container MPI libraries communicate with host MPI libraries over PMI.
    - bind mode – application in container directly uses the host MPI libraries.
  - Need to test if the container's MPI libraries are ABI compatible with the host's MPI libraries.
  - Need to hunt for the locations of the optimized Cray MPI libraries, and make sure those are visible to the application in the container.

Challenges with Running Containers at Scale

- Fully coupled climate models rely on Slurm's 'heterogenous jobs' feature
  - Two ways, shares an MPI_COMM_WORLD between two separate jobs or job steps

```
#SBATCH --clusters=c4
#SBATCH --time=00:10:00
#SBATCH -N 1
#SBATCH --mem=0
#SBATCH hetjob
#SBATCH -N 1
#SBATCH --mem=0

srun  -N1 -n4  ./mpitest : -N1
-n4 ./mpitest
```

```
#SBATCH --clusters=c4
#SBATCH --time=00:10:00
#SBATCH -N 2

srun  -N1 -n4  ./mpitest : -N1
-n4 ./mpitest
```

Output:

```
Hello from rank 4 on host nid00363
Hello from rank 3 on host nid00362
Hello from rank 6 on host nid00363
Hello from rank 5 on host nid00363
Hello from rank 7 on host nid00363
Hello from rank 1 on host nid00362
Hello from rank 2 on host nid00362
Hello from rank 0 on host nid00362
```

- This feature has not been tested for HPC containers, so we had to figure it out

Motivation and Goals

- Containers are the way forward, so need to get containerized climate applications close to native performance.
- Test the various ways of running containerized MPI applications to get that level of performance - identify if different ways suit different use cases.
  - Use benchmarks and real world climate models for testing hybrid and bind methods for MPI containers.
- Get containerized fully coupled climate models to run without needing any changes to the model code.
  - Need to get containers working with Slurm's heterogeneous jobs.

EVALUATION

Tests and Benchmarks

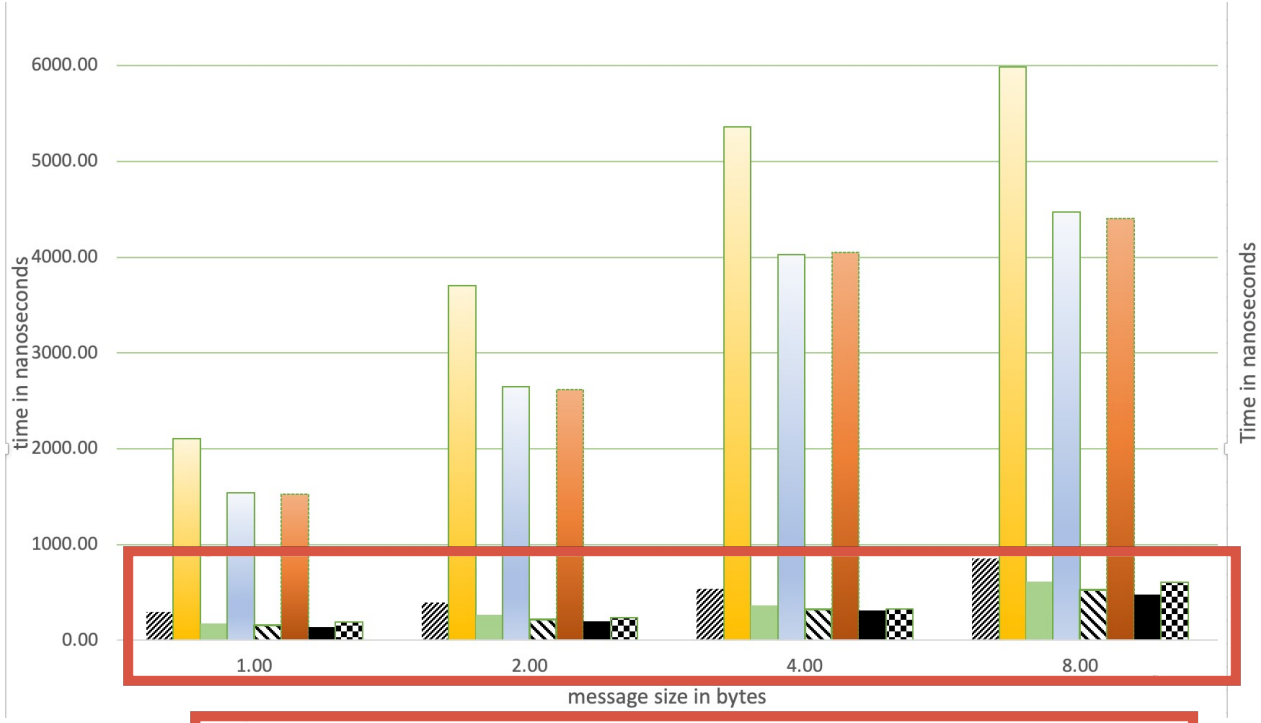
- OSU benchmarks
  - Standard benchmarks for testing MPI performance on a cluster
  - We use osu_allgather
- Aquaplanet [2]
  - Based on GFDL Atmosphere Model 4
  - Useful for studying large scale atmospheric phenomenon like tropical cyclone frequency
- ESM4 (Earth System Model 4.1) [3]
  - Fully coupled - couples atmosphere models AM4 and LM4P with ocean models MOM6 and SIS2
  - AM4+LM4P runs on a separate set of MPI ranks with its own communicator, same for MOM6+SIS2
  - The two parts communicate over MPI_COMM_WORLD
- Each of these were built using the container's compilers and libraries.

Containers and Host Environment

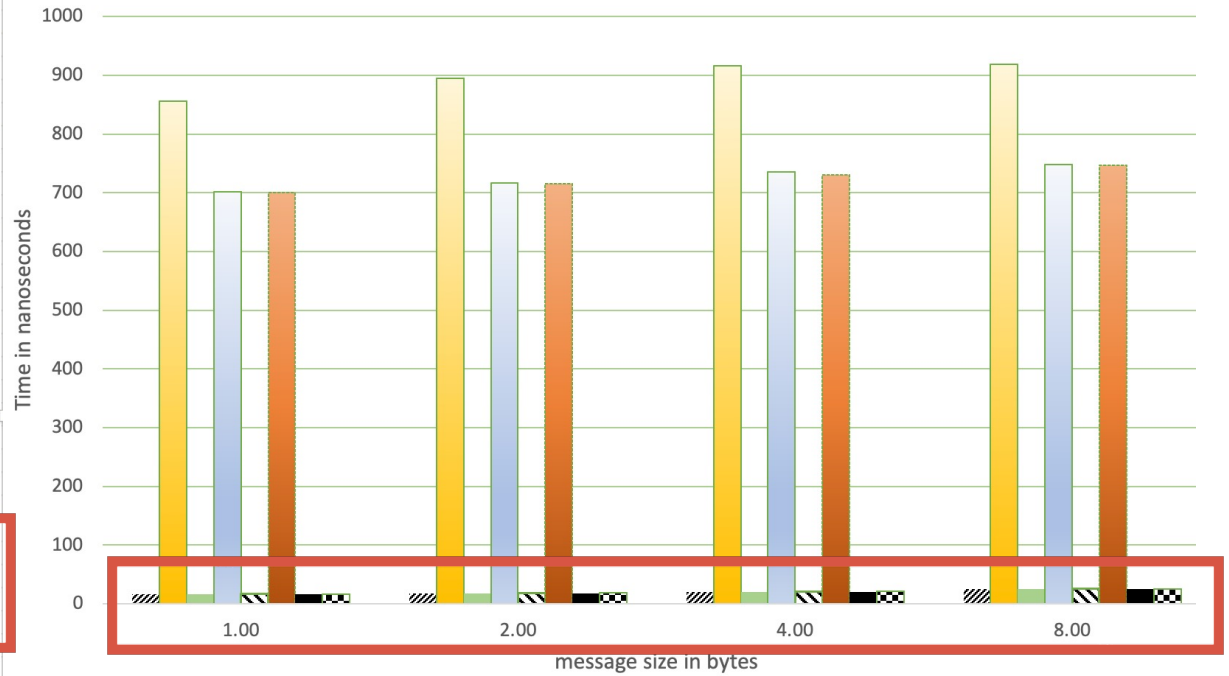
Environment	Packages
GNU MPICH container (hpcme_gnu_rhel8.sif)	RHEL 8.4 GCC 8.4.1 MPICH 3.3.2 Netcdf 4.8.0
Intel MPI container (ubuntu-intel_2022.1.1.sif)	Ubuntu 18.04 Intel 2021.5 Intel MPI 2019.10 GCC 7.5.0 Netcdf 4.8.0
Intel 2021 OneAPI container (intel2021.2_netcdfc4.7.4_ubuntu.sif)	Ubuntu 18.04 Intel 2021.2 Intel MPI 2021.2 GCC 7.5.0 Netcdf 4.7.4
GAEA Host Environment	Intel 19.0.5 (PrgEnv-intel) GCC 8.3.0 (PrgEnv-gnu) Cray MPICH 7.7.11 Intel MPI 2019.5

Results: osu_allgather on small and large scale

Large scale - 450 nodes, 32 tasks per node
c4 – 2 18-core Intel Broadwell CPUs, 64gb DDR4 memory



Small scale - 16 nodes, 32 tasks per node
t4 – 2 16-core Intel Haswell CPUs, 64GB DDR4 memory



- GNU MPICH-Host Cray MPICH bind
- Intel 2021 OneAPI-Host Cray MPICH bind
- Intel MPI-Host Cray MPICH bind
- Native PrgEnv-intel
- GNU MPICH-HOST Intel MPI bind
- Intel 2021 OneApi hybrid
- Intel MPI hybrid
- Native PrgEnv-gnu

Running the Climate Models: Aquaplanet and ESM4

- Run for a month of simulation time, useful for calculating simulated years per day (SYPD: number of simulated years a computer can run in a day).
- Focus on comparing main loop times of the models, removes initialization and finalization

Results: Aquaplanet

total cores	bare metal (s)	hybrid (s)	bind (s)
216	1533.07	1745.50	1587.56
384	1194.56	1467.48	1217.80
432	847.57	1131.60	850.40
576	632.96	946.75	635.71
864	467.62	722.46	464.47
1152	378.64	775.05	392.15
1728	283.91	572.44	302.46

Intel 2021 OneAPI container
intel2021.2 netcdfc4.7.4 ubuntu.sif
Ubuntu 18.04 Intel 2021.2
Intel MPI 2021.2
GCC 7.5.0 Netcdf 4.7.4

cores	bare-metal (s)	bind (s)
216	2026.35	1951.29
384	1597.43	1532.81
432	1108.32	1103.46
864	877.39	601.98

GNU MPICH container
hpcme gnu rhel8.sif
RHEL 8.4 GCC 8.4.1
MPICH 3.3.2 Netcdf 4.8.0

Results: ESM4 (with bind mode using Host Cray MPICH)

nodes	atmos ranks x threads	ocean ranks	Native (s)	Container (s)
30	216 x 2	648	2898.5	2899.6

```
binds = \
“/etc/hosts,/opt/cray,/etc/opt/cray,/var/opt/cray/alps,/lustre/f2/pdata/gfdl”

container = “intel2021.2_netcdfc4.7.4_ubuntu.sif“

srun --export=ALL --mpi=pmi2 --ntasks=216 --cpus-per-task=2 \
  singularity exec --bind $binds \
  $container \
  ./wrapper.sh ./fms_esm4.1_sis2_compile.x : \
--export=ALL --mpi=pmi2 --ntasks=648 --cpus-per-task=1 \
  singularity exec --bind $binds \
  $container \
  ./wrapper.sh ./fms_esm4.1_sis2_compile.x
```

intel2021.2 netcdfc4.7.4
ubuntu.sif
Ubuntu 18.04 Intel 2021.2
Intel MPI 2021.2
GCC 7.5.0 Netcdf 4.7.4

ISSUES WE RAN INTO

Issues we ran into: heterogenous jobs

- Using the host Cray library with bind mode hangs on MPI_Init in heterogeneous job allocations, but worked with heterogeneous job steps in a single allocation
  - This is fixed in newer Slurm versions, now works in both
- Intel's MPI library (in the Intel containers with hybrid mode) couldn't receive het task distributions from Slurm – produces MPI shared memory errors
- The GNU MPICH library (in GNU MPICH container with hybrid mode) works with heterogeneous job allocations (with #SBATCH hetjob), but hangs in heterogeneous job steps in single allocation

Issues we ran into: RPATH

- The GNU mpicc sets the RPATH in our container, which has higher precedence than LD_LIBRARY_PATH
  - The application's RPATH points to the container's MPI libraries
  - the host MPI libraries we set on the LD_LIBRARY_PATH in bind mode are not used when app is run
- workarounds
  - Use `-Wl,--enable-new-dtags` flags with gcc to set RUNPATH during compile
  - Set LD_PRELOAD with the host Cray MPI libraries
  - Invoke the dynamic linker with the `--inhibit-rpath` flag
  - Use the `--preload` flag on the newer GNU dynamic linker

Issues we ran into: Finding the MPI libraries

- Finding and binding MPI libraries and all the supporting libraries
  - Specifically needing the cray-mpich-abi module
  - Cray MPICH needs alps, ugni, xpmem, udreg; make sure to find and bind all the locations in the container
    - `--bind /etc/hosts,/opt/cray,/etc/opt/cray,/var/opt/cray/alps`
  - Load your modules and update the LD_LIBRARY_PATH in the container
    - `export LD_LIBRARY_PATH=$CRAY_LD_LIBRARY_PATH:$LD_LIBRARY_PATH`
    - Put this in a wrapper and call your application with the wrapper
  - e4s-cl [4] works towards fixing this, but it currently needs mpicc and mpirun, not suited for a Cray system

Issues we ran into: --export=NONE

- GAEA users use #SBATCH --export=NONE to start clean environment for jobs
- Using this causes the heterogeneous job with containers to hang where it would otherwise work
- Need to figure what in the exported environment allows the job to run

Future work

- Investigate some of the earlier odd behaviors (the spikes at 16 byte size, --export=NONE)
- Examine closer if container usage has any negative impact on the Lustre filesystem
- Auto discovering and mounting Cray libraries into container
- Portability of containers onto other HPC systems or cloud

Thank you!
Questions?
abrahams@ornl.gov

References

[1] https://www.noaa.gov/organization/information-technology/gaea

[2] Zhao, Ming, et al. "The GFDL global atmosphere and land model AM4. 0/LM4. 0: 1. Simulation characteristics with prescribed SSTs." *Journal of Advances in Modeling Earth Systems* 10.3 (2018): 691-734.

[3] Dunne, J. P., et al. "The GFDL Earth System Model version 4.1 (GFDL-ESM 4.1): Overall coupled model description and simulation characteristics." *Journal of Advances in Modeling Earth Systems* 12.11 (2020): e2019MS002015.

[4] https://github.com/E4S-Project/e4s-cl