2020-08-13 JEDI Containers and Cloud Platforms

Yannick opened the meeting with several announcements.

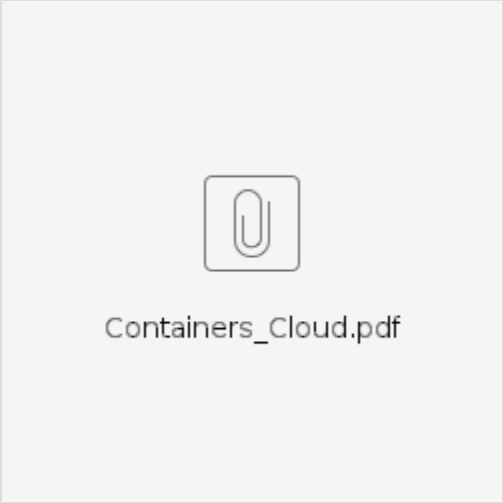
Today's topic is software containers and Mark M will present.

An OOPS PR is under review which will allow the removal of an unused section of YAML from JEDI configuration. This involves updates to the models, but the model code can be updated at your convenience. A reviewer from each model group has been assigned. If you haven't reviewed yet, please take a look and raise any concerns.

We are currently working to resolve issues with moving to the c++14 standard. There are a few PR's utilizing c++14 features that are held up for now (eg. Wojciech's Parameter class updates). Mark M (later in the meeting) added that the issue is that Intel compilers use the GNU C++ headers and libraries, and some HPC systems have old GNU versions (predating c++14). The solution is to install newer versions of GNU (>= 8), but it is not required to actually change the default GNU version (just need to have the new headers and libraries accessible to Intel). Mark added that there is a ZenHub issue in jedi-stack tracking this. Please feel free to contribute to the discussion in the ZenHub issue.

Please update your GitHub profiles and fill out the "real name" and "company" entries. Some GitHub user names are cryptic, plus we want to know your organization for gathering statistics for reporting purposes.

Yannick then handed the meeting over to Mark M who presented the following slides.



The following is a summary of the presentation. Please see the slides for details.

The presentation was divided into two main sections: A status report on the current state of containers for JEDI followed by results from a benchmark test of container runtime performance.

Mark noted that there were many contributors to this effort, and a notable result is that we are the first to run Singularity across nodes on either the S4 HPC system or the Discover system. So, the sys admins at both sites have been eager to work with us to get this working and have been a great help. AWS colleagues have also been extremely helpful. Mark added that the Singularity community and support team are very good.

Containers are a means of packaging specific versions of libraries and environments that can be unpacked on many platforms to create a consistent "look" across those platforms. We use Docker, Singularity and Charliecloud implementations of containers, and our build system is based on Docker where the other containers can be auto generated from the Docker container. We have several repositories for managing this work: JCSDA/jedi-stack (build processes), JCSDA/docker and JCSDA/docker_base (source configuration for Docker) and JCSDA/containers (build processes for Singularity and Charliecloud).

The JEDI containers that we build come in a variety falling in three categories: development, application and tutorial. Development containers are for developing JEDI source code, application containers are for running pre-compiled JEDI applications (3Dvar, H(x), etc.) and tutorial are for online and academy training. The application and tutorial containers are private for now, but will be made public once the upcoming JEDI release occurs.

A couple questions came up at this point. Sergey asked if we are restricted to using Singularity on the cloud. Technically no, but Singularity is proving to be the best match for running on the cloud.

Rahul asked if there were instructions for building containers. The docker, docker_base and containers repos all contain instructions and build scripts /processes. Mark is happy to help if you run into any issues.

Chris H asked if the containers come with run scripts, and Mark answered that only the tutorial containers include run scripts. There is a wide variety of lmod modules, environments, file system structures across the various HPC systems that would make this difficult to manage. Also, files in application containers are read-only whereas most users will want to have the freedom to continually modify their config files, input files, parallel partitions, etc. Mark pointed out that Singularity gives you access to your home directory so you can keep a set of run scripts there for use on a particular system. In addition to your home directory, the directory that you launch Singularity from (e.g. work space on an HPC system) is automounted in the container. So, it's easy to access any config and data files you have on your system. Singularity can also be configured to mount other HPC scratch disks for use inside the container.

The concept of "supercontainers" was introduced which are application containers that are designed to run across multiple nodes on supercomputer systems including cloud-based clusters. Mark noted that the hpc container maker from NVIDIA is very nice and is what we use. Mark explained a "best practice" when using supercontainers in an MPI multi-task job is to run mpiexec natively where each task fires up a Singularity container. This is termed "hybrid mode" and helps to attain compatibility with the various job queue managers such as SLURM. See slides for details.

As part of the supercontainer presentation, Mark showed an example using SLURM on AWS. Mark O asked why an approach using the '-e' option to Singularity (for isolating the container environment) was not being used. Mark M explained that using '-e' was what he initially did to develop this flow, it simply wouldn't work this way because the host MPI must communicate with the container via environment variables that are set at run time.

Chris H mentioned that Mark's presentation has been extremely helpful in regard to some issues he is trying to solve on Hera, and he should be able to get these resolved now.

Mark switched over to the benchmark portion of the presentation at this point. The benchmark test case is an FV3-GFS 3DVar run using c192 resolution and ingesting ~12 million observations. This application uses 864 MPI tasks and is in a 12x12x6 layout. Mark ran the benchmarks on Discover, S4 and AWS which were chosen simply because these had Singularity available at the time. The results showed that there is essentially no overhead for using Singularity compared to running natively outside the container. See the slides for details. One surprising result was that on S4 the inferior hardware (S4-ivy on the plot) ran faster than the superior hardware (S4 on the plot). Mark is working with the S4 IT folks to understand this result. Mark pointed out that the data depicted in the bar charts were developed by running the benchmark 10 times and showing the mean runtimes with each bar along with the associated standard deviations on the black error range markers.

Chris H asked if OpenMP was used in the benchmark, and the answer was no.

Sergey asked why is AWS slower than Discover. Mark responded that the expectation (based on the hardware) is that AWS would be faster than Discover, and that the AWS IT team is currently investigating this result. One suspect is the file system that was used (NFS versus FSX Lustre), but initially this hasn't shown a significant difference. Mark O suggested another suspect which is the "hypervisor" layer on AWS (Discover would not have this layer).

Rahul asked if the benchmark could be run on Hera and Orion just to see how they compare with the HPC platforms in the study. Steve V asked to add Cheyenne to that list. Mark replied that this would be a good thing to do, and cautioned that he will need to make sure the source code used for benchmarking on Hera, Orion and Cheyenne is the same as the source code inside the Singularity container (which dates from April, 2020) to get a good apples-to-apples comparison. Chris H added that there exists a Singularity module on Orion, and Sergey added that Singularity might be available on Hera as well. Previously Singularity was installed on Hera but only accessible to system administrators and managers, though this may have changed.

Dan asked if Mark tried the partitions on Discover with Skylake processors and an Omnipath interconnect. The answer is no, but again Mark noted that this would be a good benchmark to run.

Mark concluded with notes related to the talk:

- For AWS we have a script, called jedinode.py, that can used to get access to a single node for development
- AWS provides a feature called "Parallel Cluster" that can be used for creating multi-node clusters for HPC applications
- Singularity is creating an open source version on GitHub which will be called HPCNG (HPC Next Generation)
- The outlook of running Singularity natively (without Vagrant) on the Mac looks a bit tenuous

Steve V commented that Docker can be run natively on the Mac, and Mark agreed that this is a good option but cautioned that the Docker interface is not as nice as Singularity.

Daniel Abdi asked if Charliecloud could be configured to run on multiple nodes. Mark replied that in theory yes, but when he took a look at doing this he couldn't get it to work (this was a while ago - it might be worth another try with the insights gained from the work presented here). Mark noted that HPC IT groups tend to avoid Docker due to its security holes.

Hui asked if it's okay to continue developing on S4 and Discover outside the container. Mark replied that development outside the container is fine. He added that the consistent environment provided by the container becomes more valuable in other contexts besides development. For example, using a container to run experiments and gather data for an article to be published in a peer-reviewed journal allows the authors to publish the container version /specs making it much easier for others to reproduce the published results.