Joe Hahn

Senior Data Scientist

joe.hahn@oracle.com

2022-July-28

Using OCI Data Science to analyze

high-performance computing output

High-performance computing (HPC) on Oracle Cloud Infrastructure (OCI) accelerates the delivery of complex time-consuming numerical simulations by running that simulation’s many compute tasks in parallel across the HPC cluster’s multiple compute nodes. Those compute nodes provide a reservoir of CPUs or GPUs, memory, and network bandwidth that allows you to swiftly complete a parallelized simulation that would be too time-consuming for serial execution. Typical use cases include physical simulations for product design, clinical research, financial modeling, oil and gas reservoir modeling, etc. Often you are also performing multiple HPC calculations so that key design parameters can be optimized. Examples include selecting an airplane’s shape so that safety and performance are maximized, or the design of a sophisticated financial instrument so that risk and revenue are properly balanced.

Optimizing those design parameters requires inspecting the output generated by those many HPC simulations. This blog post shows how easily you can use OCI’s Data Science service, which provides compute, memory, and a Jupyter server with the usual python libraries (numpy, pandas, matplotlib, etc) preinstalled so you can quickly develop custom Python code to visualize your HPC output. OCI Data Science also helps optimize your simulations’ design settings. This blog post walks you through the deployment an HPC cluster on OCI, the installation of a computational fluid dynamics (CFD) code on that cluster, the parallel execution of that code across the cluster, and then using Data Science to visualize the HPC output. Key steps are briefly summarized here, but see this [GitLab demo](https://github.com/oracle-nace-dsai/fargo3d-demo) for additional details.

Figure 1. Text


Figure . Cloud architecture for HPC and Data Science

## Deploy an HPC cluster

Figure 1 shows the cloud architecture used in this example. In the Oracle Cloud Marketplace, select **HPC Cluster Stack** to create an HPC cluster. Select the number of compute nodes, their shape, and their storage, and give the new cluster a name. This example builds the smallest possible HPC cluster, composed of two bare metal compute nodes of shape BM.HPC2.36 that provides 384 GB of memory and 36 compute cores on each node.

Clicking the **Create** button runs an OCI-provided Terraform script that deploys an HPC stack composed of two compute nodes and a bastion instance with all instances having all the standard HPC libraries preinstalled. You can also tailor that Terraform script further so that any additional libraries and custom settings will also be present at subsequent deployments. Deploying an HPC cluster within OCI is easy, and tailoring that cluster to a specific use case is straightforward.

Next, SSH into the bastion node, which serves as the cluster’s front door and is where the HPC job will be launched. This example runs the [FARGO3D](http://fargo.in2p3.fr/) code on the HPC cluster. FARGO3D is a CFD code that solves the differential equations that govern how flowing fluids evolve over time. FARGO3D is used for astrophysics research, which is irrelevant to most users of OCI’s HPC. Nevertheless, FARGO3D is easy to install, run, and visualize its output, and is why we are using it here. FARGO3D was developed to study the formation and evolution of recently formed planets orbiting within the circumstellar disk of gas and solids from which they formed. Our example simulates the mutual co-evolution of a just-formed planet while its gravity disturbs the dusty gas disk that also birthed it.

## Install and set up FARGO3D

Executing the HPC job that performs the parallel run of FARGO3D also requires a simple bash script posted at a Github repo. This walkthrough assumes that you clone that repo to your bastion node via:

git clone git@github.com:oracle-nace-dsai/fargo3d-demo.git

cd fargo3d-demo

and then download and unpack FARGO3D,

wget <http://fargo.in2p3.fr/downloads/fargo3d-1.3.tar.gz>

tar -xvf fargo3d-1.3.tar.gz

cd fargo3d-1.3

and then compile FARGO3D for a parallel execution via

make SETUP=fargo PARALLEL=1 GPU=0

The FARGO3D code comes with a file containing the initial conditions for the disk-planet system described above, but to make our HPC simulation more challenging, replace the FARGO3D-provided initial conditions with a modified version that samples the disk’s CFD grid 16 times more finely and extends the simulation’s timewise duration by five times. Overwrite the original fargo.par file with the modified version:

cp ../fargo\_big.par setups/fargo/fargo.par

The [slurm](https://slurm.schedmd.com/documentation.html) workload manager will be used to execute FARGO3D in parallel across the cluster’s compute nodes. So copy the slurm deployment script to the working directory and inspect:

cp ../MY\_SLURM\_JOB .

cat MY\_SLURM\_JOB

which displays:

#!/bin/bash

module load mpi/openmpi/openmpi-4.0.3rc4

time mpirun ./fargo3d -k ./setups/fargo/fargo.par

When this deployment script is executed, it will first load an openmpi module to use the Message Passing Interface (MPI) library that will execute FARGO3D’s various code tasks in parallel. The above script then tells mpirun to run FARGO3D in parallel using the now-modified initial conditions, with this job’s runtime also being tracked.

## Execute the slurm job

This experiment’s FARGO3D job is submitted to slurm via the following command:

sbatch --job-name=fargo3d --nodes=2 --ntasks-per-node=32 --cpus-per-task=1 \

--exclusive MY\_SLURM\_JOB

which tells slurm to execute 32 parallel tasks on each of this cluster’s compute nodes, with one CPU dedicated to each task. This slurm job completes in 30 minutes on this small HPC cluster, and inspection of [OCI’s cloud-compute cost](https://www.oracle.com/cloud/compute/pricing.html) tells us that this 30-minute-long HPC experiment accrued a $2.80 cloud bill.

Next, check how execution times and compute costs scale by rerunning this example with   
--nodes=1 and --ntasks-per-node=1, which instructs slurm to run this job using only 1 CPU on a single compute node. That single-threaded FARGO3D job completes in 19 hours, which would cost $3.05 had the run instead occured on a typical OCI instance of shape VM.Standard2.4. If lengthy run times are impacting your project’s time to deliver, consider migrating your parallelizable workloads to HPC where run times can be shorted dramatically with no incremental compute costs.

## Visualizing HPC output with OCI Data Science

Next, deploy and configure a Data Science (DS) instance so that it can communicate with the HPC cluster, and consult [this document](https://github.com/scacela/oci-datascience-setup-steps/blob/main/Data%20Science%20Setup%20Steps.docx) for the required steps. This experiment originally intended to use OCI’s File Storage System to manage the HPC cluster’s output, but the DS-FSS connectivity will not be available until a few months hence, so instead use OCI’s command line interface (OCI CLI) to manually copy the HPC output to an OCI Object Storage bucket where the Data Science instance can see it; see also figure 1 and these more [detailed instructions](https://github.com/oracle-nace-dsai/fargo3d-demo).

Then navigate to the Data Science instance’s Jupyter server where you can develop custom Python code to read, analyze, and visualize the HPC output inside this [Jupyter notebook](https://129.213.160.170/jhahn/fargo3d-demo/blob/master/hello_world.ipynb), with Figs. 2–4 displaying highlights from that analysis.

Graphical user interface

Description automatically generated

Figure . Example Jupyter notebook that reads and visualizes the output generated by the FARGO3D simulation.

OCI Data Science provides a Jupyter notebook where you can prototype and develop custom python code to analyze and visualize HPC output. The [notebook](https://129.213.160.170/jhahn/fargo3d-demo/blob/master/hello_world.ipynb) shown in Fig. 2 displays the final state of the FARGO3D simulation of a Jupiter-mass planet orbiting within a circumstellar disk. That FARGO3D simulation uses a polar (r, ϕ) coordinate grid to track the mutual evolution of the disk-planet system, and the heatmap in Fig. 2 shows the gas disk’s density displayed with disk radius r increasing upwards and disk azimuth ϕ to the right. So this visualization scheme ‘unwraps’ the disk and reprojects it onto a rectangular grid. The planet, which has its own circumplanetary gas disk, is at the bright spot in lower middle, and the horizontal dark band is an annulus of low-density gas that was depleted by the planet’s gravitational perturbations of the disk. Tilted density-bands are spiral density waves that the planet also excites within the disk.

Chart

Description automatically generated

Figure . Code block that displays the simulated disk’s gas density versus radial distance at various times.

Figure 3 shows radial profiles of the disk’s azimuthally averaged density ρ at various simulation times. The planet’s gravitational perturbations open an annular gap in the gas disk concentric with the planet’s orbit at r=1. The density peak at r=1 is due to residual disk gas that persists along the planet’s orbit.

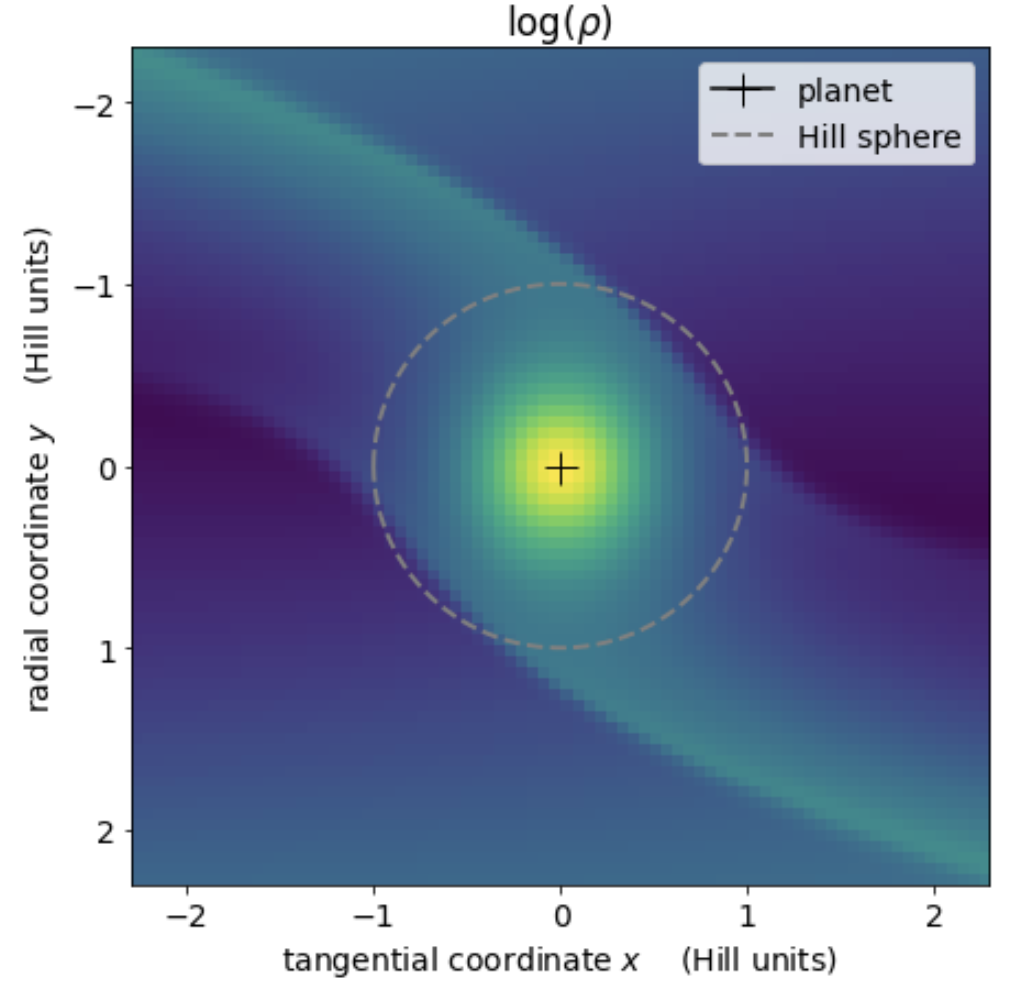


Figure . Heatmap of the circumplanetary gas disk.

The Fig. 4 heatmap zooms in on the immediate vicinity of planet that resides at the + and is surrounded by a circumplanetary disk of gas and solids being supplied by two streams of circumstellar matter that go into orbit about the planet when inside that planet’s Hill radius, which is where the planet’s gravity dominates over stellar gravity.

## Recommendations

When the serial execution of your use case’s numerical simulations is excessive and slows your solution’s delivery, investigate whether parallel execution of that code is a possibility. If so, then get those parallelized simulations executed much more swiftly using Oracle Cloud Infrastructure's HPC offering, which this example shows to be very cost-effective. And if you also need custom code to investigate that HPC output, consider developing your visualizations using Oracle’s Data Science service using the simple cloud architecture shown in Figure 1. For more information about the topics mentioned in this blog post, see the following resources:

* [Oracle Cloud Infrastructure](https://www.oracle.com/cloud/)
* [High-performance computing](https://en.wikipedia.org/wiki/High-performance_computing)
* [Computational fluid dynamics](https://en.wikipedia.org/wiki/Computational_fluid_dynamics)
* [FARGO3D](http://fargo.in2p3.fr/)
* [Message Passing Interface](https://www.mcs.anl.gov/research/projects/mpi/) (MPI)
* [OCI Command Line Interface (CLI)](https://docs.oracle.com/en-us/iaas/Content/API/Concepts/cliconcepts.htm)
* [slurm](https://slurm.schedmd.com/documentation.html)