Modest scale HPC on Azure using CGYRO

Igor Sfiligoi
University of California San Diego
La Jolla, CA, USA
isfiligoi@sdsc.edu

Jeff Candy
General Atomics
La Jolla, CA, USA
candy@fusion.gat.com

***Abstract*—Fusion simulations have traditionally required the use of leadership scale HPC resources in order to produce advances in physics. One such package is CGYRO, a premier tool for multi-scale plasma turbulence simulation. CGYRO is a typical HPC application that will not fit into a single node, as it requires several TeraBytes of memory and O(100) TFLOPS compute capability for cutting-edge simulations. CGYRO also requires high-throughput and low-latency networking, due to its reliance on global FFT computations. While in the past such compute may have required hundreds, or even thousands of nodes, recent advances in hardware capabilities allow for just tens of nodes to deliver the necessary compute power. We explored the feasibility of running CGYRO on Cloud resources provided by Microsoft on their Azure platform, using the infiniband-connected HPC resources in spot mode. We observed both that CPU-only resources were very efficient, and that running in spot mode was doable, with minimal side effects. The GPU-enabled resources were less cost effective but allowed for higher scaling.**

Keywords—HPC, Fusion, CGYRO, Cloud, Azure

I. Introduction

Fusion energy research has made significant progress over the years, yet many fundamental aspects of its physics are still not well understood. While experimental methods are essential for gathering new operational modes, simulations are used to validate basic theory, plan experiments, interpret results on present devices, and ultimately to design future devices. The current frontier is the simulation of multiscale turbulence, i.e. turbulence spanning both ion and electron scales seamlessly, which requires an arbitrary-wavelength formulation of the equations and algorithms, making them significantly more challenging and expensive. CGYRO [1], the focus of this report, is an Eulerian gyrokinetic solver specifically designed and optimized for collisional, electromagnetic, multiscale simulation.

CGYRO has been in use by the fusion community for several years. At the time of writing, a cutting-edge simulation requires several TeraBytes of memory and O(100) TFLOPS worth of compute to complete in a reasonable timeframe. Moreover, it heavily relies on global FFT computations, which require high-throughput and low-latency networking, thus making it a typical HPC application. CGYRO has been ported and extensively used on several leadership-class HPC centers, including ALCF Mira/Theta, NERSC Cori, OLCF Titan and Summit, CSCS Piz Daint and TACC Stampede2.

While leadership-class HPC centers provide for a very pleasant and effective work environment, they are heavily sought for and typically over-subscribed. We thus wanted to explore other options, and Cloud computing has recently emerged as a potential candidate. Microsoft Azure in particular is currently offering several infiniband-enabled instance types, with theoretical performance characteristics very similar to those advertised by the HPC centers.

In this work we present our experience in running a cutting-edge CGYRO simulation on Azure resources, as well as several other benchmarking runs. We initially started by provisioning on-demand resources, but quickly switched to spot ones, once it became apparent that the preemption rate was low enough to be cost effective, at the scales we were interested in.

One interesting property of Cloud computing is the explicit cost structure, that allows for direct comparison of very different compute architectures. We thus run the simulations on both CPU-only and GPU-enabled instance types and computed their relative cost-effectiveness.

II. Microsoft Azure Cloud Setup with CycleCloud

The CGYRO deployment and operational model relies on the presence of an HPC batch system and a shared file system, both of which are normally provided at an HPC center. Microsoft Azure however does not natively provide an HPC batch system capability, so we had to adopt a third-party solution. We chose Microsoft CycleCloud [2], since it is at least supported by the same vendor, even though it is not tightly integrated into the Azure infrastructure. Note that Microsoft offers the CycleCloud software for free, although that played only a minor role in our decision.

Installing CycleCloud was relatively easy, although documentation on proper configuration was lacking. Once installed and configured, the administrator can deploy an HPC batch system, complete with a shared filesystem, using a point-and-click Web interface. Note that the software was installed as an Azure instance, using the image in the Azure marketplace.

The HPC batch system chosen was SLURM, mostly due to the author's familiarity with that system; for completeness, PBS, LSF and Grid Engine are also options. CycleServer allows for push-button deployment of a fully configured SLURM submit node, with ssh access and auto-scaling capabilities. On job submission, additional worker nodes are automatically provisioned and fully configured, with the user's home area shared between all nodes. Nodes are automatically de-provisioned after a few minutes of inactivity. The administrator can provide any arbitrary image for both the submit node and the worker nodes, making it easy to fully customize the setup.

The out-of-the-box setup comes complete with the gcc 9 compiler, and MPI binaries and libraries, which works quite well for the CPU-only HPC instances. For GPU-enabled instances we required the use of PGI compilers and GPU-enabled MPI libraries, which is not something CycleCloud explicitly

This work was supported by the U.S. Department of Energy under awards DE-FG02-95ER54309 and DE-SC0017992.

supports. This required us to use a completely independent setup, which is straightforward for an expert but is likely beyond most researchers' grasp.

Two other features required changes that are not supported through the normal Web interface; requesting spot instances for worker nodes and using a dedicated, high-performance disk partition for the home area. Both tasks can be accomplished through the use of the CycleCloud API and are documented in the advanced use sections.

III. Suitability of Azure Resources for HPC

Azure offers three instance types that provide infiniband networking; NDv2 instances provide eight NVIDIA Tesla V100 32GB GPUs, HC instances provide dual Intel Xeon Platinum 8168 CPUs and HBv2 instances provide dual AMD EPYC 7002 CPUs. The first two come with 100 Gbps EDR and the last comes with 200 Gbps HDR. As a result, the network latencies between nodes are very low, as can be seen in Fig. 1.

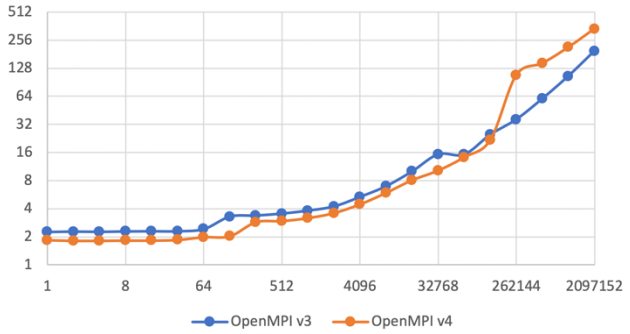

Fig. 1. Measured network latencies in us, as repoted by the osu_latency tool.

We compiled and benchmarked CGYRO on all three instance types, using gcc on CPU-only instances and PGI compilers on GPU-providing instances. OpenMPI v3 was used on all instances. We did not encounter any significant problems during setup or compilation. Note that we did also try other MPI implementations, including OpenMPI v4 and MVAPICH2; the performance difference was typically within 10%, with OpenMPI v3 generally providing the best results.

TABLE I. Observed run times and cost for a typical step in a cutting-edge CGYRO simulation by spot instance type on Azure

Instance	Nodes	Total time	Comm. time	Total cost
NDv2	24	161s	139s	$5.24
NDv2	8	369s	316s	$4.00
HBv2	36	272s	104s	$1.81
HBv2	18	441s	190s	$1.47
HC	35	416s	113s	$2.42
HC	18	763s	151s	$2.28

The measured run times for the cutting-edge CGYRO multi-scale simulation [3], along with the associated costs using Azure spot instances can be found in Table I. As can be seen, the GPU-providing NDv2 instances are drastically faster on a per-node basis, but AMD CPU-based NBv2 instances provide a much better value per dollar. The main reason for GPU-based instances to trail in cost-effectiveness is due to the amount of time spent in network communication; a single 100 Gbps EDR infiniband link is a poor match for the compute power of 8 V100 GPUs. Note the Intel CPU-based NC instances trail the other two in the benchmarks and are thus not a good fit for users using CGYRO. For smaller simulations, the situation was similar. Detailed results are presented in the poster.

Using spot instances implies lower availability and potential preemption during runtime. CGYRO is instrumented to deal with preemption by implementing checkpointing at a tunable interval; in our tests, we checkpointed every couple of hours, resulting in negligible overhead.

Availability of a large number of nodes can be a problem. We were never able to provision more than 24 NDv2 nodes or 36 HBv2 nodes. The experienced preemption rate was however very low for smaller node counts. We ran a simulation using 8 NDv2 nodes over a 5-day period with only 3 preemption events and had several 8 hour runs on 18 HBv2 nodes without a single preemption. Given that spot instances cost less than 20% compared to the more reliable on-demand instances, the trade-off is well worth it for the small node counts.

IV. Comparison with Leadership-class HPC Centers

In order to compare the performance of CGYRO on Cloud resources, we ran the same cutting-edge simulation also on OLCF Summit and NERSC Cori. As can be seen from Table II, Azure resources in spot mode can easily match the performance on 48 Intel Xeon Phi CPU nodes on Cori. The slowest result on Summit is however significantly faster than anything we measured on Azure, which is due to Summit nodes both having dual-rail 100 Gbps infiniband and only six NVIDIA Tesla V100 16 GB GPUs; we could not perform the CGYRO simulation with less than 32 nodes, due to GPU memory requirements.

TABLE II. Observed run times and cost for a typical step in a cutting-edge CGYRO simulation on on-prem HPC resources

System	Nodes	Total time	Comm. time
Summit	32	86s	67s
Cori	128	165s	62s
Cori	48	339s	160s

The situation was comparable for smaller simulations, with more details available in the poster.

Acknowledgment

This work was supported by the U.S. Department of Energy under awards DE-FG02-95ER54309 (General Atomics Theory grant), DE-SC0017992 (AToM SciDAC-4 project). Computing resources were provided by the Oak Ridge Leadership Computing Facility under Contract DE-AC05-00OR22725 (ALCC program) and the National Energy Research Scientific Computing Center under Contract DE-AC02-05CH11231.

References

[1] https://doi.org/10.1016/j.compfluid.2019.04.016

[2] https://docs.microsoft.com/en-us/azure/cyclecloud/

[3] https://github.com/scidac/atom-open-doc/blob/master/2020.11-SC20/multiscale_input/input.cgyro