Running IFS on Microsoft Azure

Cathal O'Brien

Cathal.Obrien@ecmwf.int

ECMWF

Background

- This study wanted to evaluate objectively the pros and cons of HPC on the cloud

- Hoped to develop platform agnostic infrastructure to evaluate other cloud providers in the future

- constraints:
  - Just looking at Azure
  - Just looking at CPU instances, no GPUs

- Hypothesis:
  - Pros: flexible scaling, trying out different hardware
  - Cons: price

Presentation structure

- Overview of HPC on Azure
- Infrastructure for deploying IFS-capable clusters on the cloud
- Benchmarking results

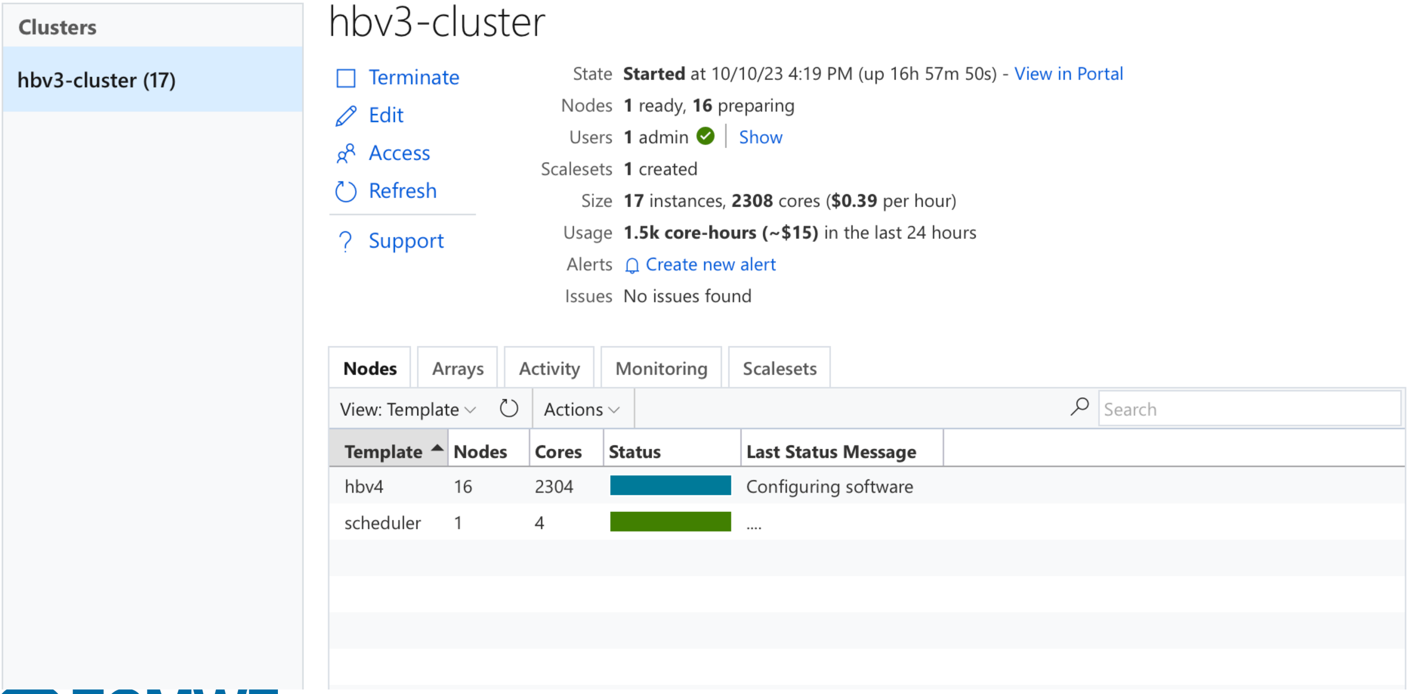
Overview of HPC on Azure

Azure HPC instances

VM	CPU (Cores / Node)	Memory per node (GB)	Interconnect (Gb/s)
HBv2	AMD Rome (120)	456	Infiniband HDR (200)
HBv3	AMD Milan-X (120)	448	Infiniband HDR (200)
HBv4	AMD Genoa-X (176)	704	Infiniband NDR (400)
Atos (Reference)	AMD Rome (128)	256	Infiniband HDR (200)

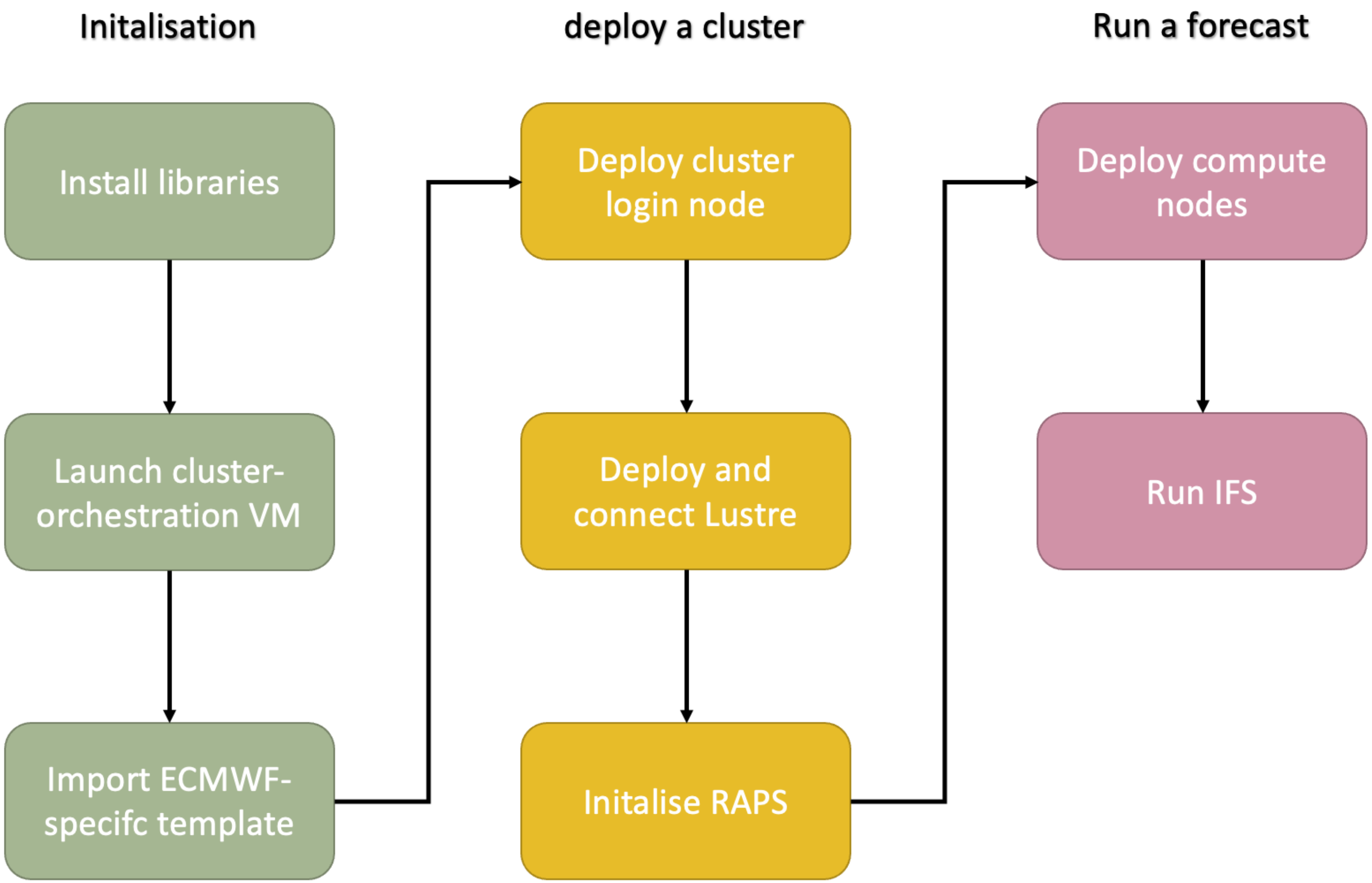
Cyclecloud

- Software for creating and managing HPC clusters on Azure
- Define cluster templates
  - VM Image, compute queues, etc…
- Persistent login nodes, compute nodes autoscale
- Web portal & command line interface



Infrastructure for deploying IFS-capable clusters on the cloud

Cluster creation pipeline



Initalisation

- Create Virtual Machine Image
  - Based on an existing Azure HPC Image loaded with OFED drivers etc
  - Install compilers and MPI libraries
  - Prebuild IFS build for all relevant stacks
- Provide user login details and billing info
- Deploy Cyclecloud using Terraform

Deploy a cluster

- Cluster based on custom template
  - Slurm scheduler
  - Spot partitions for HBv2, HBv3 & HBv4
  - on-demand partitions for longer jobs

- Launches the cluster and does some further configuration which couldn't go into the Image
  - Setup SSH keys
  - Configure git
  - Clone repos

```
[[nodearray hbv3]]
Extends = nodearraybase
MachineType = Standard_HB120-96rs_v3
ImageID = $ImageID
MaxCount = $MaxNodes
Azure.MaxScalesetSize = $HPCMaxScalesetSize
AdditionalClusterInitSpecs = $HPCClusterInitSpecs
Interruptible = true
MaxPrice = $SpotMaxPrice

        [[[configuration]]]
        slurm.default_partition = true
        slurm.hpc = true
        slurm.partition = hbv3
        slurm.use_pcpu = false
```

Cyclecloud template block defining a compute partition

Deploying Lustre on the cloud

Throughput (MB/s/TiB)	Block size (TiB)	Max capacity (TiB)	Price ($/GiB/month)	Price ($/Block/hour)
40	48	768	0.084	5.65
125	16	128	0.145	3.25
250	8	128	0.21	2.36
500	4	128	0.341	1.91

- Lustre can be linked to blob storage to populate it lazily with input data
- Script created to deploy Lustre in ~10 mins
- Load inputs into lustre before job starts
    - 700GB, ~10k files, ~15 mins
- Lustre on the cloud can – and should – be spun up on-demand

- Nice side benefit: getting input data on cluster in ~25 mins via Lustre vs hours using scp!

- IOR benchmarking from Microsoft available here

Run a forecast

- Start nodes
  - ~2 mins to find nodes (asking for 100 nodes will take longer)
  - ~4 mins to configure software
- We had an issue with ~10% of nodes being unhealthy
  - Instead of using Slurm we created a script to start the nodes
  - Allocates additional nodes, then runs a health-check and returns a subset of nodes which passes these tests and deallocates the rest
  - This specific issue is fixed now
- Once nodes are started, run jobs with Slurm as usual
  - Works OK but Slurm integration could certainly be better

```
[hpc_admin@hbv3-cluster-scheduler ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
hbv2         up   infinite     99  idle~ hbv3-cluster-hbv2-[1-99]
hbv3*        up   infinite     99  idle~ hbv3-cluster-hbv3-[1-99]
hbv4         up   infinite     38  idle~ hbv3-cluster-hbv4-[1-29,32-40]
hbv4         up   infinite      2  down* hbv3-cluster-hbv4-[30-31]
```

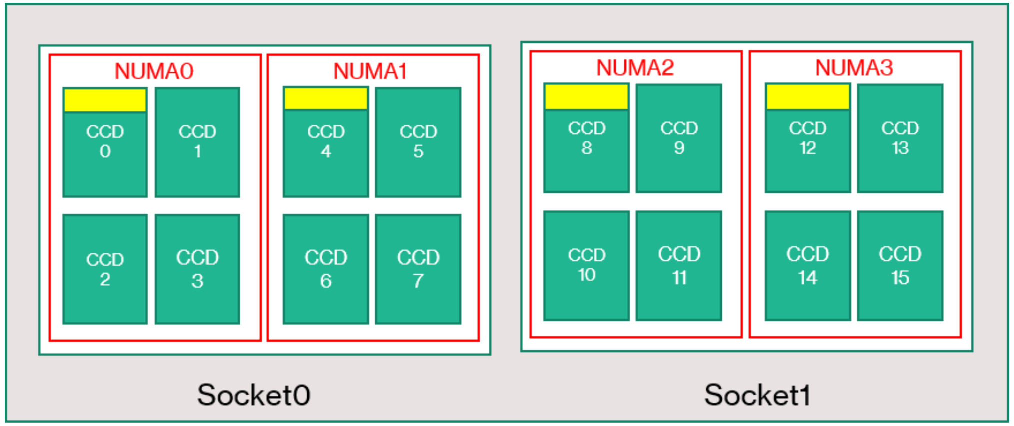
Benchmarking results

Task pinning on Azure HBv3

- Pinning locks processes to specifc cores on the node and is crucial for achieving good performance

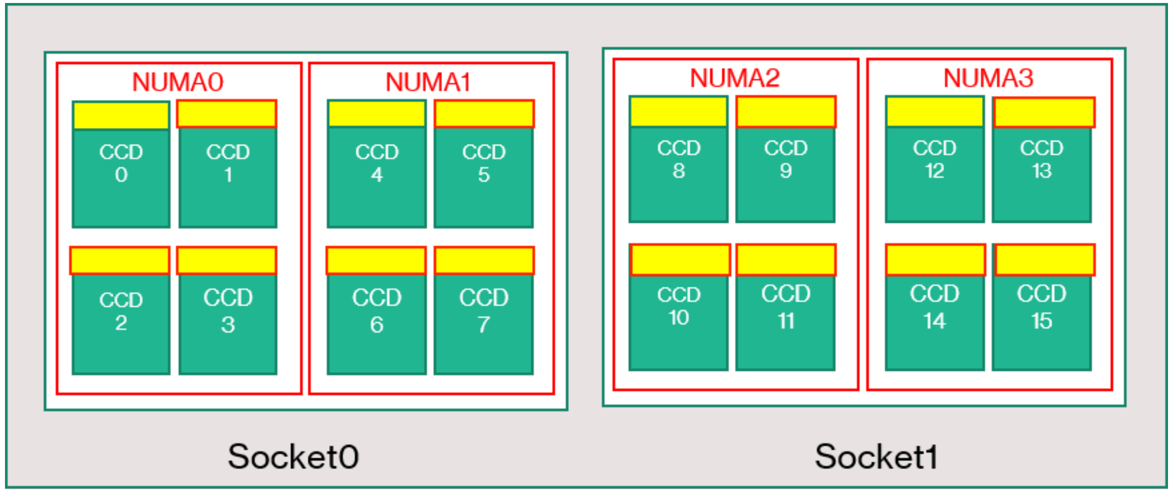
- Otherwise, OS can migrate MPI tasks across the node, decreasing performance
  - Cache invalidation
  - Acessing data in different NUMA domains

- Cloud hypervisor adds an extra difficulty here



Task pinning on Azure HBv3

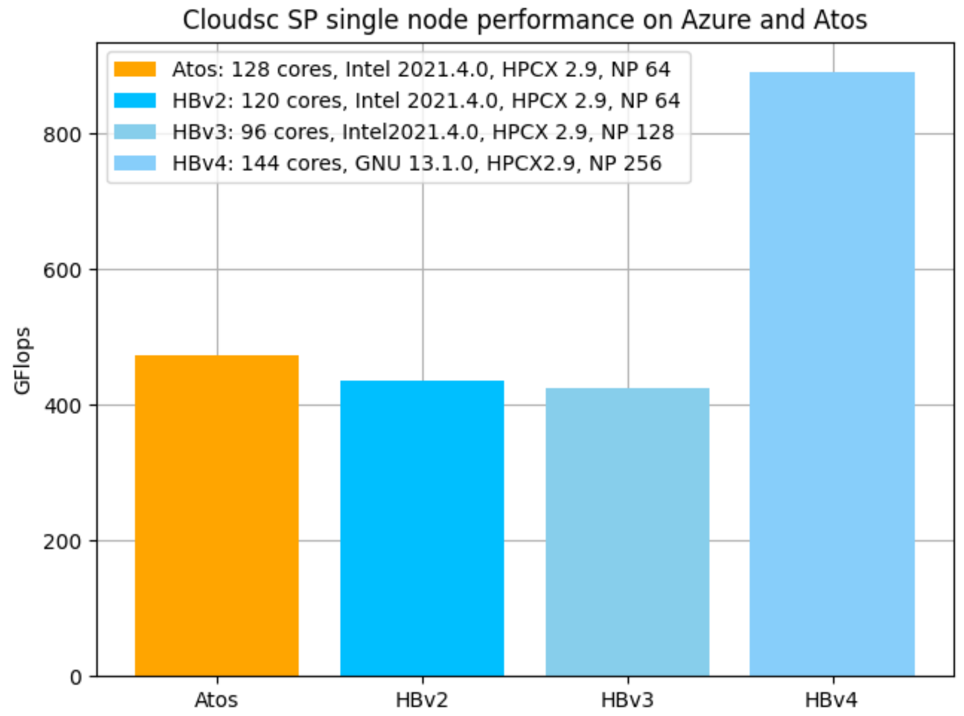
Size	vCPU	Processor	Memory (GiB)	Memory bandwidth GB/s
Standard_HB120rs_v3	120	AMD EPYC 7V73X	448	350
Standard_HB120-96rs_v3	96	AMD EPYC 7V73X	448	350
Standard_HB120-64rs_v3	64	AMD EPYC 7V73X	448	350
Standard_HB120-32rs_v3	32	AMD EPYC 7V73X	448	350
Standard_HB120-16rs_v3	16	AMD EPYC 7V73X	448	350

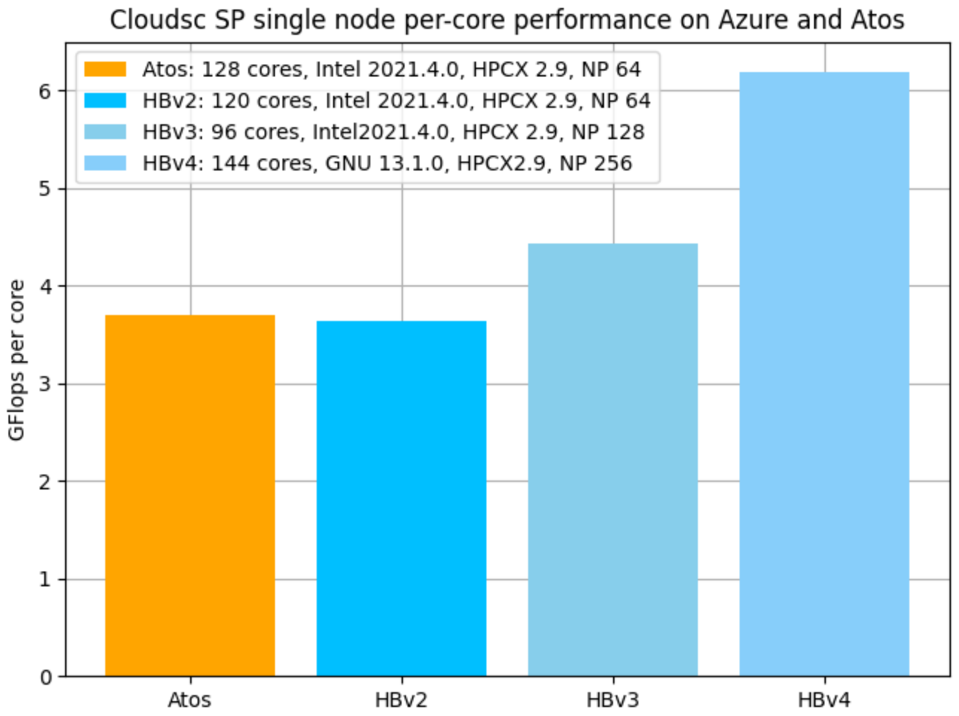


- Further reduces availible cores
  - HBv3: 128 => 120 => 96 cores
  - HBv4: 192 => 176 => 144 cores
  - HBv2: 128 => 120 cores (different HV layout)
- Still exclusive use of entire node
- Not present on all cloud providers

Cloudsc

- Cloud microphysics scheme extracted from IFS
- Available on Github
- Hybrid MPI - OpenMP, usually run on a single node for benchmarking
- Compute bound on most architectures
- Higher is better





ECRad

- Atmospheric radiation scheme

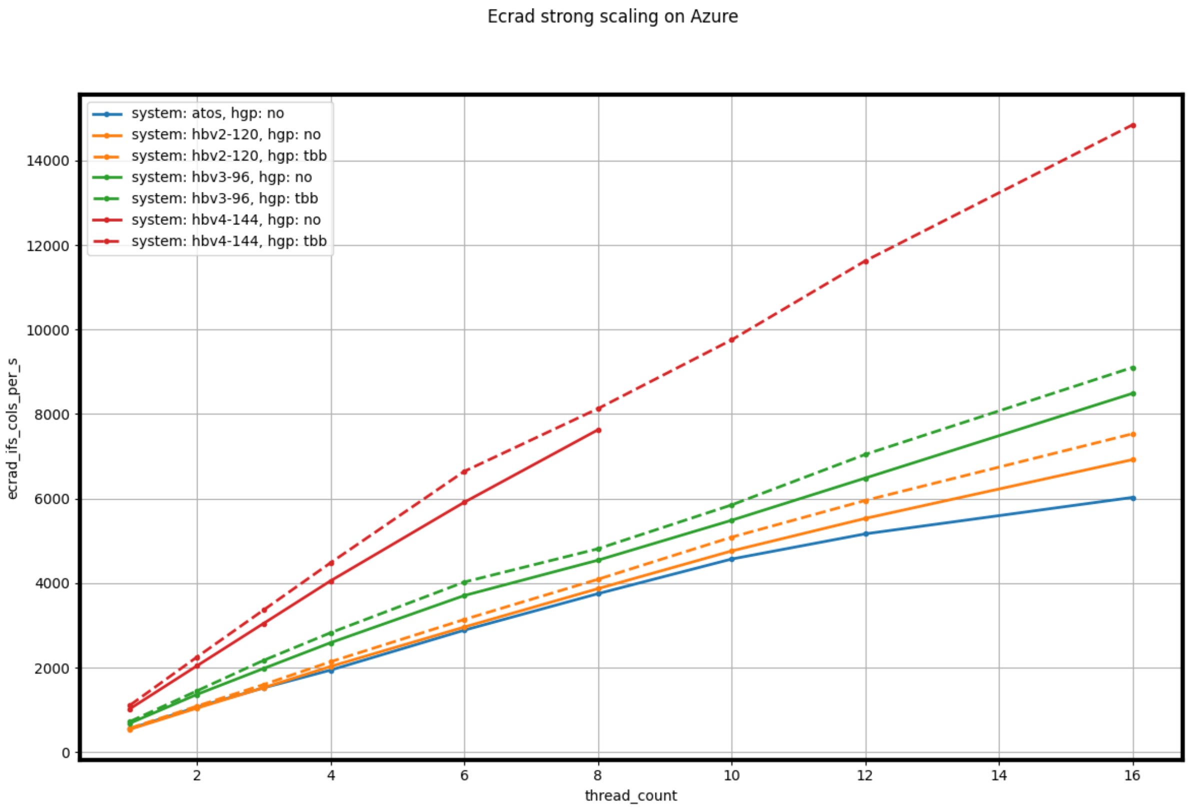
- Availible on [Github]

- Memory bound on most systems

- Pure OpenMP, runs within a single NUMA domain for First Touch reasons

- Higher is better

- Dashed line = 200GBs of Statically Allocated Huge Pages on Azure



ECTrans

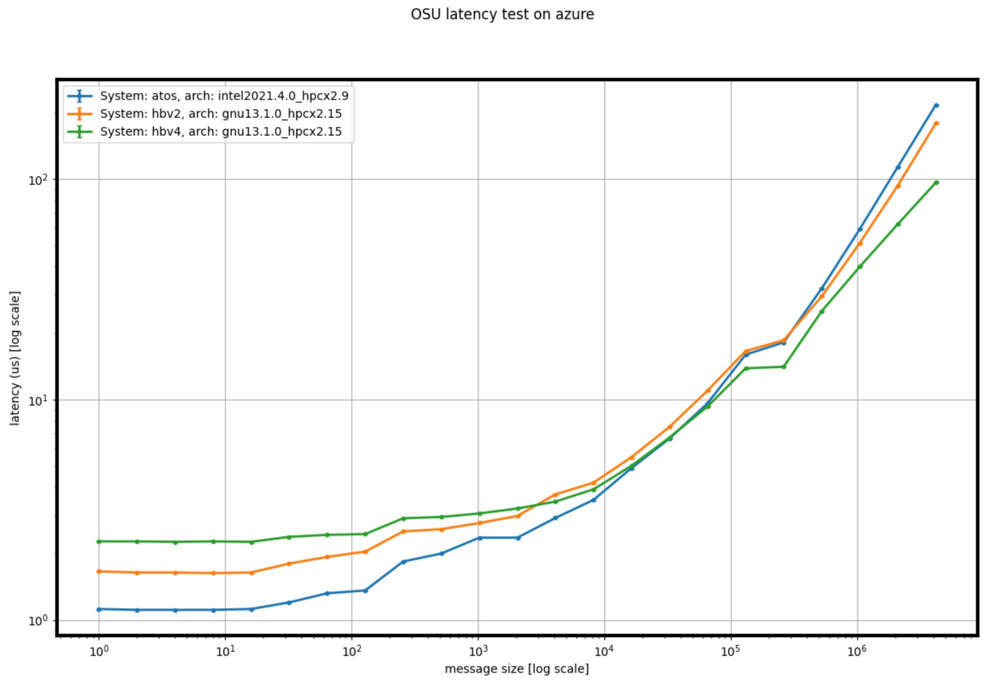
- Spectral transform
- Hybrid MPI-OpenMP, many nodes
- Availible on Github

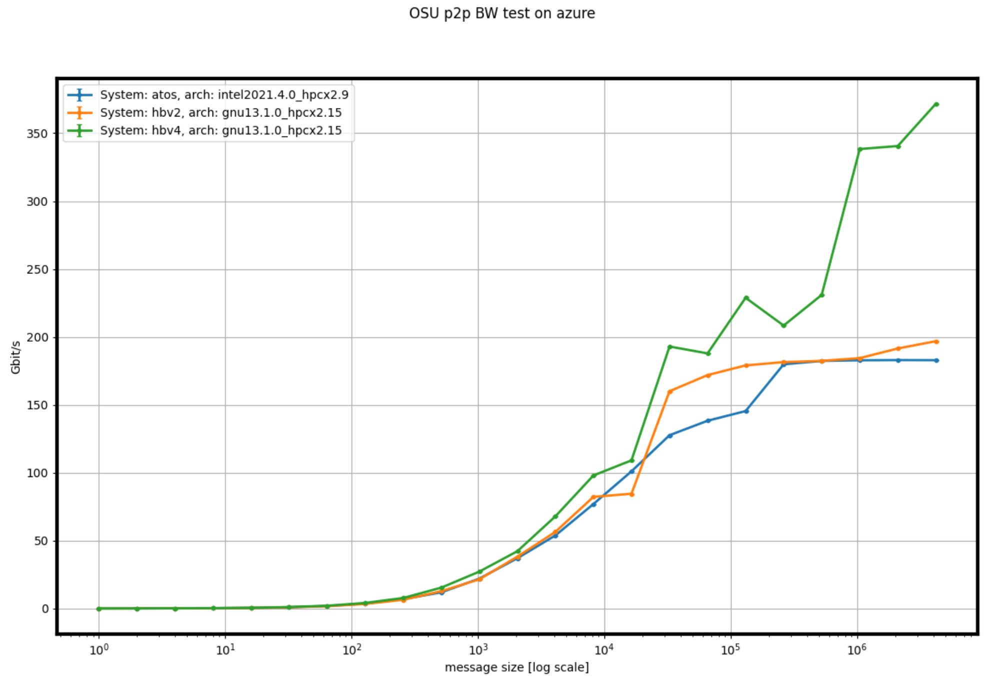
- Unfortunately when time came to benchmark we had a memory leak when running on Azure

OSU Benchmark suite

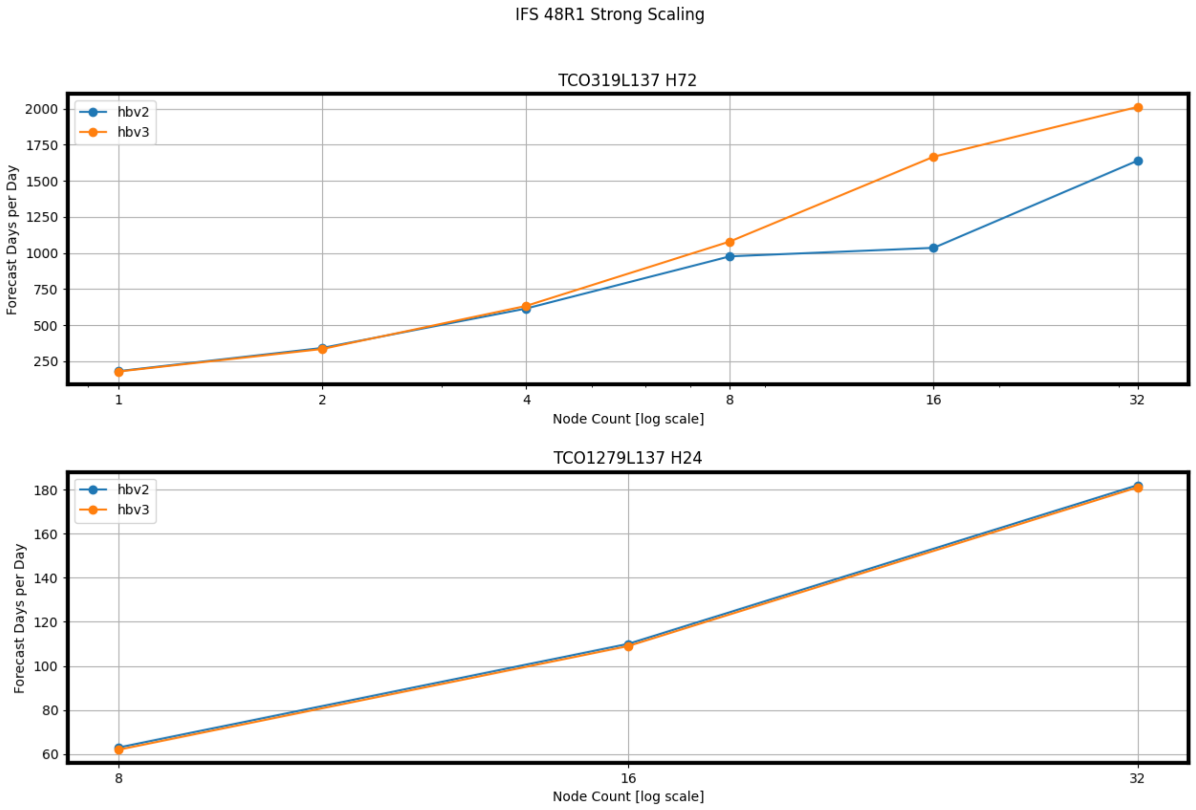
- Above, lower is better
- Below, higher is better

- High latency at small message sizes requires further investigation





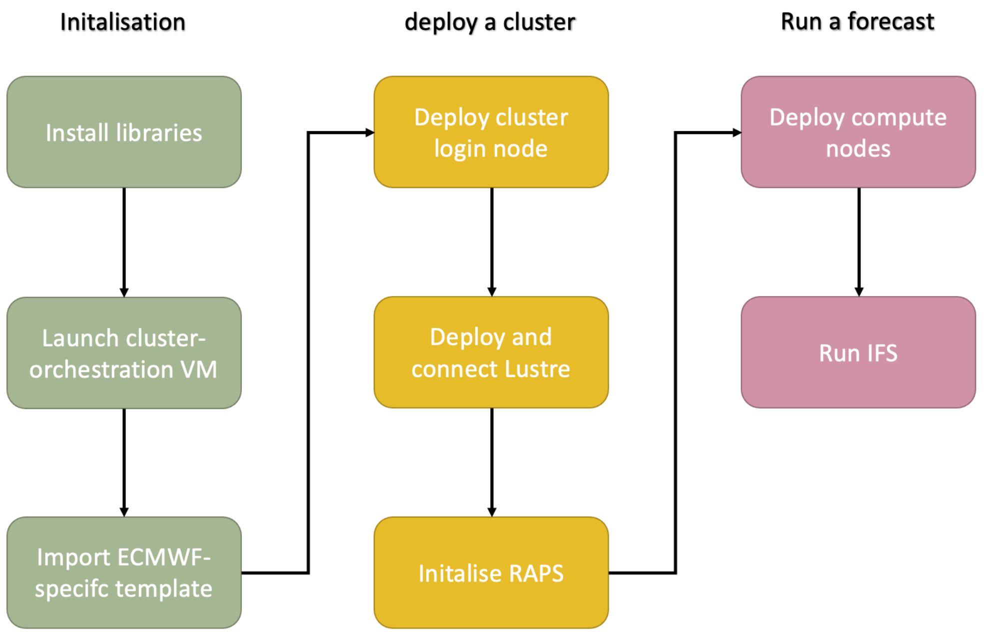
IFS – Strong Scaling



- Caveats:
  - IO turned off, no comparison to Atos
  - Lacking optimisations like Statically Allocated Huge Pages
  - IB for HBv4 was not availible

Conclusion

- Cloud is great for flexibly trying out new hardware
  - Great performance results
  - Quotas and budget limits scaling
- Significant setup work involved
- HPC on the cloud still a young field



Thanks

- Thanks to Alexandre Jean, Cedric Husianycia & everyone from Microsoft for their support

Further reading

ORNL/TM-2023/3083

Evaluating the Cloud for Capability Class Leadership Workloads

Jack Lange, Thomas Papatheodore, Todd Thomas, Chad Effler, Aaron Haun, Carlos Cunningham, Kyle Fenske, Rafael Ferreira da Silva, Ketan Maheshwari, Junqi Yin, Sajal Dash, Markus Eisenbach, Nick Hagerty, Balint Joo, John Holmen, Matthew Norman, Dan Dietz, Tom Beck, Sarp Oral, Scott Atchley, Phil Roth

September 11, 2023

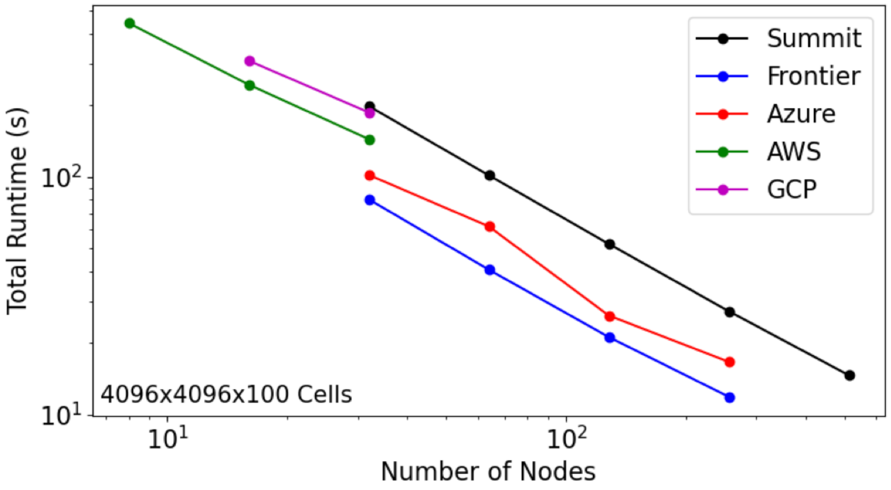


Figure 5. 3D Cloud Model strong scaling results on the cloud vendors, Summit, and Frontier.

Further reading

Computer Science > Distributed, Parallel, and Cluster Computing

[Submitted on 27 Oct 2022 (v1), last revised 1 Nov 2022 (this version, v2)]

Noise in the Clouds: Influence of Network Performance Variability on Application Scalability

Daniele De Sensi, Tiziano De Matteis, Konstantin Taranov, Salvatore Di Girolamo, Tobias Rahn, Torsten Hoefler

Cloud computing represents an appealing opportunity for cost-effective deployment of HPC workloads on the best-fitting hardware. However, although cloud and on-premise HPC systems offer similar computational resources, their network architecture and performance may differ significantly. For example, these systems use fundamentally different network transport and routing protocols, which may introduce network noise that can eventually limit the application scaling. This work analyzes network performance, scalability, and cost of running HPC workloads on cloud systems. First, we consider latency, bandwidth, and collective communication patterns in detailed small-scale measurements, and then we simulate network performance at a larger scale. We validate our approach on four popular cloud providers and three on-premise HPC systems, showing that network (and also OS) noise can significantly impact performance and cost both at small and large scale.

Comments: To appear in SIGMETRICS 2023
Subjects: **Distributed, Parallel, and Cluster Computing (cs.DC)**; Networking and Internet Architecture (cs.NI); Performance (cs.PF)
ACM classes: C.2; C.4
Cite as: arXiv:2210.15315 **[cs.DC]**
(or arXiv:2210.15315v2 **[cs.DC]** for this version)
https://doi.org/10.48550/arXiv.2210.15315