OpenStack + AWS, HPC (aaS) and GPUs - A Pragmatic Guide

Martijn de Vries
Chief Technology Officer
About Bright Computing

• Headquarters in Amsterdam, NL & San Jose, CA

• Bright Cluster Manager:
  • Streamlines cluster deployments
  • Manages and healthchecks cluster after deployment
  • Integrates with OpenStack, Hadoop, Spark, Kubernetes, Mesos, Ceph

• Used on thousands of clusters all over the world

• Features to make GPU computing as easy as possible:
  • CUDA & NVIDIA driver packages
  • Pre-packaged versions of machine learning software
  • GPU configuration, monitoring and health checking
Renting versus buying

Problem description:

• Users wants to be able to run some GPU workload
• Only limited amount of hardware with GPUs available on-premise
• More GPU hardware needs to be made available to satisfy user demand
• Costs need to be minimized
• Users will need to share resources on single multi-tenant infrastructure

• Options:
  • Buy more hardware
  • Migrate workload to public cloud
Running workload off-premise
Why offload HPC workload to public cloud?

- Immediate access to hardware
- Easy to scale up/down
- Pay per use
- Lower costs compared to buying when resource demand varies greatly over time
Why keep HPC workload on-premise?

• More control over hardware (e.g. CPU, GPU, interconnect) configuration
  • (Latest) Models, configuration, firmware versions
• Substantial input/output data volume
• Cheaper at scale and high utilization
• Better control over performance (i.e. no hidden bottlenecks)
• Security
• Need access to on-site infrastructure (e.g. tape library)
• Sentimental reasons
Cloud native versus traditional workload

• Traditional HPC workload
  • Expects:
    • POSIX-like shared filesystem (e.g. NFS, Lustre, GPFS, BeeGFS)
    • MPI runtime
    • Low latency interconnect (e.g. IB, OmniPath)
  • Scheduled by HPC workload management system (e.g. Slurm, PBS Pro)

• Cloud native applications:
  • Designed to take advantage of elastic cloud-like environment
  • Composed of micro-services running in containers
  • Designed for dynamically scaling up/down
  • Mostly for software as a service, increasingly also for batch jobs
  • Scheduled by e.g. Kubernetes or Mesos+Marathon
Challenges

• Not all workload may be offloadable to cloud
• How much hardware on premise?
• How much hardware to spin up in cloud?
  • Instance flavors
  • Usage commitments
• How to make cloud offloading transparent to end-user?
• How to run traditional workload in cloud?
• How to run cloud native workload on-premise?
Hybrid approach

• On-premise cluster extended with resources from public cloud
• Makes possible to do gradual transition to cloud
• Multi-cloud possible (e.g. some jobs to AWS, some to Azure)
• Uniformity: cloud nodes look & feel same as on-premise nodes
  • Single workload management system
  • Same user authentication
  • Same software images used for provisioning
  • Same shared software environment (e.g. NFS applications tree, environment modules)
• Applications will run in cloud as if they run on on-premise cluster
Achieving Uniformity

• Provisioning
  • Node-installer loaded as AMI (instead of loading through PXE)
  • Cloud director serves as provisioning node for all nodes in particular cloud region
  • Cloud director receives copy of all software images (kept up-to-date automatically)
  • Same kernel version

• Authentication
  • Head node runs LDAP server
  • Cloud director runs LDAP replica server
  • AD/external LDAP also possible

• Workload management
  • Typical set-up: one job queue per cloud region
  • User decides whether to run job on-premise or in cloud by submitting to queue
  • Single queue containing all nodes also possible
Scaling node count up/down

- Adding/removing cloud nodes can be done:
  - Manually by administrator
  - Automatically using cm-scale tool based on workload in queue

- cm-scale can perform following operations on nodes:
  - Power on/off
  - Create new node (in cloud) / terminate
  - Move to new node category (i.e. re-purpose node)
  - Subscribe to new configuration overlay (i.e. re-purpose node)

- Custom policies possible as Python module
Moving data in/out of cloud

• Jobs depend on input data and produce output data
• cmsub allows user to specify data dependencies for jobs
• Job input data will be moved into cloud before job resources are allocated
• Data staged on temporary storage node (dynamically spun up)
• Job output data will be moved back to on-premise cluster
• Data movement is transparent to user
GPUs in AWS & Azure

• AWS

<table>
<thead>
<tr>
<th>Model</th>
<th>GPUs</th>
<th>vCPU</th>
<th>Mem (GiB)</th>
<th>GPU Memory (GiB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>p2.xlarge</td>
<td>1</td>
<td>4</td>
<td>61</td>
<td>12</td>
</tr>
<tr>
<td>p2.8xlarge</td>
<td>8</td>
<td>32</td>
<td>488</td>
<td>96</td>
</tr>
<tr>
<td>p2.16xlarge</td>
<td>16</td>
<td>64</td>
<td>732</td>
<td>192</td>
</tr>
</tbody>
</table>

• Azure

<table>
<thead>
<tr>
<th>INSTANCE</th>
<th>CORES</th>
<th>RAM</th>
<th>GPU</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC6</td>
<td>6</td>
<td>56.00 GiB</td>
<td>1X K80</td>
<td>$0.90/hr</td>
</tr>
<tr>
<td>NC12</td>
<td>12</td>
<td>112.00 GiB</td>
<td>2X K80</td>
<td>$1.80/hr</td>
</tr>
<tr>
<td>NC24</td>
<td>24</td>
<td>224.00 GiB</td>
<td>4X K80</td>
<td>$3.60/hr</td>
</tr>
<tr>
<td>NC24r</td>
<td>24</td>
<td>224.00 GiB</td>
<td>4X K80</td>
<td>$3.96/hr</td>
</tr>
<tr>
<td>NV6</td>
<td>6</td>
<td>56.00 GiB</td>
<td>1X M60</td>
<td>$1.24/hr</td>
</tr>
<tr>
<td>NV12</td>
<td>12</td>
<td>112.00 GiB</td>
<td>2X M60</td>
<td>$2.48/hr</td>
</tr>
<tr>
<td>NV24</td>
<td>24</td>
<td>224.00 GiB</td>
<td>4X M60</td>
<td>$4.97/hr</td>
</tr>
</tbody>
</table>
Running workload on-premise
GPUs in multi-tenant environment

• Simple solution:
  • Build single multi-user cluster
  • Workload management system to let users request GPU resources

• More flexible solution:
  • Allow GPUs to be consumed through OpenStack instances
  • Users can run any OS they like
  • Cluster-on-Demand (COD) for users that want a cluster for themselves
Cluster on Demand (HPCaaS)

- COD spins up fully functional Bright clusters inside of:
  - Azure
  - AWS
  - OpenStack
- Deployment time 2-3m
- Fully functional clusters become disposable resources
- Great for:
  - Development teams
  - Power users that need/want full control of environment
  - HIPAA / PCI compliance
  - Cluster partitioning for different departments
OpenStack & GPUs

- Use special GPU instance flavor to request GPUs
- Uses PCI passthrough
- vGPUs not possible yet due to lack of support in KVM

```
[martijn@krusty ~]$ nova flavor-show cod.g1.gpu
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-FLV-DISABLED:disabled</td>
<td>False</td>
</tr>
<tr>
<td>OS-FLV-EXT-DATA:ephemeral</td>
<td>0</td>
</tr>
<tr>
<td>disk</td>
<td>0</td>
</tr>
</tbody>
</table>
| extra_specs                                   | {
| id                                            | "gpu": "true", "pci_passthrough:alias": "gpu:1"
| name                                          | 6910987c-209b-4210-b056-a630618fbf7d        |
| os-flavor-access:is_public                    | False                                      |
| ram                                           | 4096                                       |
| rxtx_factor                                   | 1.0                                        |
| swap                                          | 2                                          |
| vcpus                                         |                                            |
Bright & DCGM

• GPU related functionality in Bright:
  • GPU management (e.g. settings)
  • GPU monitoring
  • GPU healthchecking

• Used to be implemented using NVML API
• As of Bright 8.0 uses NVIDIA DCGM (Data Center GPU Manager)
• DCGM packaged and set up automatically on all nodes
• CUDA and NVIDIA driver also packaged
Bright & Deep Learning

• Allow users to get deep learning workload up with minimal effort

• Bright packages:
  - Caffe : 1.0
  - Theano : 0.9.0
  - MXnet : 0.9.3
  - Tensorflow : 1.1.0
  - Tensorflow-legacy : 0.12
  - bazel : 0.4.5
  - keras : 2.0.3
  - CNTK : 2.0rc2
  - CUDNN: 5.1 and 6.0
  - DIGITS : 5.0 (Updated Feb 2017)
  - NCCL : 1.3.4
  - Caffe2: 0.7.0
  - Caffe-MPI : 6c2c347
  - OpenCV3 : 3.1.0
  - Protobuf : 3.1.0
  - Chainer : 1.23.0
  - cuPy : 1.0.0b1
  - CUB : 1.6.4
  - MLPython : 0.1
  - TensorRT : 1.0
Demo

• Spin up small virtualized cluster in Bright Engineering’s internal Krusty cloud
• 1 virtual head node, 1 virtual GPU node (Tesla K40)
• Extend virtual cluster into Azure with 2 GPU nodes (Tesla K80)
Conclusions

• Bright GPU clusters running can easily be extended to AWS and Azure for extra temporary capacity
• OpenStack can be used to offer GPUs to users in on-premise infrastructure
• Bright’s Cluster-on-Demand can be used to create disposable Bright clusters on the fly
• Bright Cluster Manager provides GPU management & monitoring interface backed by DCGM
• Bright Cluster Manager provides rich collection of Machine Learning frameworks, tools & libraries