CloudXPRT Data Analytics Workload

Introduction

The CloudXPRT - Data Analytics workload measures scalable data analysis of particle physics experiment data (HIGGS dataset) using XGBoost. XGBoost is a gradient-boosting framework that data scientists often use for ML-based regression and classification problems. The purpose of the workload in the context of CloudXPRT is to evaluate how well an on-prem or cloud hardware infrastructure enables XGBoost to speed and optimize model training. The workload reports latency and throughput rates. As with the data analytics workload, testers can use this workload's metrics to compare IaaS stack performance and to evaluate whether any given stack is capable of meeting SLA thresholds.

Set up the benchmark

These scripts will install and create a Kubernetes cluster using Kubespray. They will help you:

- configure your environment to run CloudXPRT,
- get the IP addresses for all machines in your cluster,
- set up passwordless SSH,
- install Ansible/Kubespray requirements,
- create the cluster, and
- remove the cluster once you are done running CloudXPRT.

Terminology

- Node - A single machine or virtual machine
- Control Plane Node - The node running the installation, this will become the Kubernetes control plane node.
- Worker Node - Each machine that will join the Kubernetes cluster.

Supported OS

- On-prem: Ubuntu 20.04.2 or later
- Cloud: Ubuntu 18.04 or later

Minimum requirements

We recommend running this benchmark on high-end servers. While running, the benchmark will scale to use all available cores. However, for functional testing, your physical node or VM must have at least:

- 16 logical or virtual CPUs
- 8GB Ram
- 50GB Disk Space

Installation steps

Setup environment

In each machine in your cluster:

```
- Set the sudouser password if it is not set (Note: **must be the same on
each machine**)

  ```
sudo passwd sudouser
  ```

- Add the following line at the end of /etc/sudoers file

  ```
sudouser ALL=(ALL) NOPASSWD: ALL
  ```

- Ensure openssh-server is installed

  ```
sudo apt-get install openssh-server -y
  ```

- Allow password authentication

  - Edit /etc/ssh/sshd_config
  - Uncomment and modify the PasswordAuthentication line to allow SSH login
with password

      ```
  PasswordAuthentication yes
      ```
```

- Restart sshd

  ```
sudo service sshd restart
  ```

Control Plane node

Go to the installation directory.

```
cd CloudXPRT_vX.XX_data-analytics/installation
```

1. Edit cluster_config.json

   For each machine in the cluster, add its IPV4 address and optionally desired hostname. One machine per {..} section within the "nodes" list, starting with the Control Plane node.

   Note: Although optional, each hostname must be unique and can only contain **lowercase alphanumeric characters**. If hostnames are not provided, Kubespray will rename each host as node1, node2, ..., nodeN. This means that the Control Plane node's hostname will be changed to 'node1'.

   If your machines are behind a proxy, make sure to set "set_proxy" to "yes" add the correct proxy settings for "http_proxy" and "https_proxy". Those proxy settings will be applied on all the nodes to ensure that they can communicate through the Kubernetes networking plugin. Furthermore, you must reboot the nodes in order for them to take effect since /etc/environment is modified. You have the option "reboot" to allow the prepare-cluster.sh script to reboot all the nodes automatically. By default, the reboot option in cluster_config.json is set to 'yes'. If you set it to 'no', please manually reboot your machines after running prepare-cluster.sh, otherwise the cluster creation in the step 3 will fail.

   Example configuration for a three node cluster:

```
  "nodes": [
    {
        "ip_address": "192.168.0.11",
        "hostname": "controlplane"
    },
    {
        "ip_address": "192.168.0.12",
        "hostname": "worker1"
    },
    {
        "ip_address": "192.168.0.13",
        "hostname": "worker2"
    }
],
```

2. In the Control Plane node, run "prepare-cluster.sh" script as sudo user to perform preparation steps.

```
sudo ./prepare-cluster.sh
```

3. In the Control Plane node, run the "create-cluster.sh" script as sudo user

```
sudo ./create-cluster.sh
```

This process may take anywhere from 6 up to 20 minutes.

Note: If you get an error with respect to docker-ce repository '**RETRYING: ensure docker-ce repository public key is installed ...**', double check that the proxies are configured correctly! You may repeat the "prepare-cluster.sh" script to set this again, or you may manually edit them in each node of your cluster.

You should also double check that the date and time are the same on all of the nodes.

For more reference on Kubespray and possible errors, please check out their GitHub repo: https://github.com/kubernetes-sigs/kubespray

Reset Docker and Kubernetes cluster

To remove cluster and docker installation on every node, run the "remove-cluster.sh" script in the Control Plane node after you finish your runs.

```
sudo ./remove-cluster.sh
```

Answer 'y' or 'yes' to the prompt.

Note: This will not remove the proxy settings. If you want to run CloudXPRT again, you can run the "create-cluster.sh" script to re-create the Kubernetes cluster.

Setup data analytics Benchmark

This process may take anywhere from 5 to 10 minutes.

```
cd CloudXPRT_vX.XX_data-analytics/setup
# Execute the next command only if using a multi-node cluster
./cnb-analytics_OnPrem-MultiNode_setup.sh ## multi-node cluster only
# Execute the next command for all cluster types, single-node or multi-
node
sudo ./cnb-analytics_setup.sh
```

Run the benchmark

Once you complete successful installation

```
cd CloudXPRT_vX.XX_data-analytics/cnbrun/
```

Configure parameters for a test run

Edit the cnb-analytics_config.json file to set the parameters for CloudXPRT data-analytics. It is sugested you run it once with default parameters to make sure correct functionallity.

```
    nano cnb-analytics_config.json
    cpus_per_pod: Number of vCPUs per Pod. default 12
    numKAFKAmessages: Number of transactions to be delivered and executed.
default 1. Users should normally use at least 100 messages for statistical
purposes.
    loadgen_lambda: Inter-arrival time (sec) between transactions following
Poisson distribution. default 12
```

Run CloudXPRT-analytics

Once parameters are configured, run the cnbrun executable.

```
    sudo ./cnbrun
    sudo ./cnb-analytics_parse-all-results.sh
```

Note: use cnb-analytics_clear.sh to reset kubernetes in case you have an invalid run.

Deep dive analysis to determine best cluster configuration for best throughput

A script is provided to create a swept analysis in order to find the best throughput under a particular SLA.

```
    sudo ./cnb-analytics_run-automated.sh
```

Make sure you set the desired parameters. Identify the number of vCPUs in your cluster and Look through the respective case scenario in line 46. In the following example, we will be editing the parameters for a cluster with 48 vCPUs. In this case, we are willing to run the workload using three different Lambdas and three different vCPUs per Pod configurations for a total of 9 different runs.

```
    nano cnb-analytics_run-automated.sh
    48) Lambda=(31 32 33); vCPU_per_POD=(11 14 22);
    sudo ./cnb-analytics_run-automated.sh
    sudo ./cnb-analytics_parse-all-results.sh
```

In case of errors please clear the temp PODs using:

```
    sudo ./cnb-analytics_clear.sh
```

Benchmark results

A script is provided to create a csv table from output folders

```
    ./cnb-analytics_parse-all-results.sh
```

Some of the metrics listed in the output are listed below: - NumberOfPods: number of working Pods - vCPUsperPod: number of vCPUs used per Pod - DeliveredKAFKAmessages:

number of Kafka messages that were processed among Pods. - 90th_Percentile: Tail latency for the 90th percentile - Throughput_jobs/min: Throughput in transactions per minute User has the freedom to define a throughput that comply with 90th_Percentile latency. - FILE: location of output_result.txt file containing the results from a particular simulation - NumberOfPods: number of working Pods executing XGBoost - vCPUsperPod: number of vCPUs used per Pod (executing XGBoost training) - number of Jobs: Total number of jobs executed during the simulation - Lambda: Interarrival time between transactions (seconds). The lower the lambda the more traffic you send to the CUT. - jobs_duration: Includes creation of Pods and the eleapsed time between the creation of the first transaction at the Load generator until arrival of the last transaction. - min_duration: minimum transaction time - max_duration: maximum transaction time - stdev_duration: Standard deviation of transaction time - mean_duration: maximum transaction time - 90th_Percentile: 90th percentile for transaction times - 95th_Percentile: 95th percentile for transaction times - Throughput_jobs/min: Throughput in transactions per minute

Setup and run on AWS

Preparing to setup benchmark on AWS:

On local Ubuntu linux machine, create a new user then switch to this user

```
sudo adduser awsuser
sudo adduser awsuser sudo
```

If you are using GUI on the local Ubuntu machine, logout and log back in as gcpuser. Otherwise, you can directly change over to the new user using the following command:

```
su - awsuser
cd ~
```

Download Terraform binary and put it under /usr/local/bin directory

- https://www.terraform.io/downloads.html

Create AWS IAM user and change permissions for this user, refer to AWS official documentations:

- https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html
- https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_change-permissions.html

The AWS IAM user needs to be granted with the following permissions:

- AmazonEC2FullAccess
- AmazonRoute53FullAccess
- AmazonS3FullAccess
- IAMFullAccess
- AmazonVPCFullAccess

Install AWS CLI, refer to the AWS official documentations:

```
https://docs.aws.amazon.com/cli/latest/userguide/install-linux.html
```

Create access keys for IAM user, refer to the AWS official documentations

```
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html
```

Configure AWS CLI using access keys created

```
aws configure
    AWS Access Key ID [None]: AKIXXXXXXXXXXXXX
    AWS Secret Access Key [None]: sIrkzNOXxXXXXXXXXXXXxxxXXXX
    Default region name [None]: us-west-2
    Default output format [None]: json
```

Verify key and secret are stored into "~/.aws/credentials" file

```
[default]
aws_access_key_id = AKIXXXXXXXXXXXX
aws_secret_access_key = sIrkzNOXxXXXXXXXXXXxxxXXXX
```

Create SSH key pairs

```
ssh-keygen
```

Create Virtual Machines (VMs)

```
tar -xzf CloudXPRT_vX.XX_data-analytics.tar.gz
cd CloudXPRT_vX.XX_data-analytics/terraform/aws
modify variables.tf file
terraform init
terraform apply
wait some time for it to finish (around 1-2 minutes)
```

Run CloudXPRT and save results

Access cluster

```
Get the public addresses of VMs created
terraform show
```

Edit config file under .ssh directory to bypass company proxy issues, an example of config file:

```
User ubuntu
ProxyCommand nc -X 5 -x proxy.XXX.com:1080 %h %p
```

Note: Only needed if ssh/scp is blocked by company proxy, get the proxy settings from your companies' IT.

Copy ssh key file to the VM to be set up as Control Plane node

```
scp .ssh/id_rsa ubuntu@public_IP_of_VM:~/.ssh
```

Copy CloudXPRT release package to the VM to be set up as Control Plane node

```
scp CloudXPRT_vX.XX_data-analytics.tar.gz ubuntu@public_IP_of_VM:~/
```

Note: Make sure you use your own connection string!

SSH into Control Plane node

```
ssh ubuntu@public_IP_of_VM
```

Install Kubernetes cluster

```
tar -xzf CloudXPRT_vX.XX_data-analytics.tar.gz
cd CloudXPRT_vX.XX_data-analytics/installation
ifconfig
modify cluster_config.json file with IP address shown with ifconfig
sudo ./prepare-cluster-CSP.sh
sudo ./create-cluster.sh
```

setup CloudXPRT data-analytics

```
cd CloudXPRT_vX.XX_data-analytics/setup/
# Execute the next command only if using a multi-node cluster
sudo ./cnb-analytics_OnPrem-MultiNode_setup.sh ## multi-node cluster only
# Execute the next command for all cluster types, single-node or multi-
node
sudo ./cnb-analytics_setup.sh
```

Run the benchmark

A script is provided to create a swept analysis in order to find the best throughput under a particular SLA.

```
cd CloudXPRT_vX.XX_data-analytics/cnbrun/
sudo ./cnb-analytics_run-automated.sh
```

Note: Results will be written in the 'output_*' directories. **Note:** use cnb-analytics_clear.sh to reset kubernetes in case you have an invalid run.

Save results locally

Leave the SSH connection from the Control Plane node. In your local machine, copy the results

```
scp ubuntu@public_IP_of_VM:~/CloudXPRT_vX.XX_data-
analytics/cnbrun/output_* .
```

Clean up Cluster

After you are done running CloudXPRT and have saved the results:

```
terraform destroy
```

References:

- https://www.bogotobogo.com/DevOps/DevOps-Kubernetes-II-kops-on-AWS.php
- https://medium.com/containermind/how-to-create-a-kubernetes-cluster-on-aws-in-few-minutes-89dda10354f4
- https://github.com/kubernetes/kops/blob/master/docs/aws.md
- https://medium.com/@mcyasar/amazon-aws-kubernetes-kops-installation-7a205fe2d118
- https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html
- https://docs.aws.amazon.com/cli/latest/userguide/install-linux.html
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html
- https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html
- https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_change-permissions.html

Setup and Run on GCP

Preparing to setup and run benchmark on Google Cloud Platform

On local Ubuntu linux machine, create a new user then switch to this user

```
sudo adduser gcpuser
sudo adduser gcpuser sudo
```

If you are using GUI on the local Ubuntu machine, logout and log back in as gcpuser. Otherwise, you can directly change over to the new user using the following command:

```
su - gcpuser
cd ~
```

Download Terraform binary and put it under /usr/local/bin directory

- https://www.terraform.io/downloads.html

Install Google Cloud SDK and other tools by using the Google SDK installer (recommended)

- Reference: https://cloud.google.com/sdk/docs/downloads-interactive

```
curl https://sdk.cloud.google.com | bash
exec -l $SHELL
```

Install Google Cloud SDK in alternative ways, refer to Google official documentations:

```
https://cloud.google.com/sdk/install
```

Log on your Google cloud account, make sure your account works and configure default credentials. Need create a project to work on or use an existing one.

```
gcloud init
gcloud compute zones list
gcloud auth application-default login
```

Create service account and download the service account key to your machines where KOPS run, refer to Google official documentations:

```
https://cloud.google.com/docs/authentication/production
```

Create SSH key pairs

```
ssh-keygen
```

Create Virtual Machines (VMs)

```
tar -xzf CloudXPRT_vX.XX_data-analytics.tar.gz
cp your-service-account-key.json CloudXPRT_vX.XX_data-
analytics/terraform/gcp
cd CloudXPRT_vX.XX_data-analytics/terraform/gcp
modify variables.tf file
terraform init
terraform apply
wait some time for it to finish (around 1-2 minutes)
```

Run CloudXPRT and save results

Access cluster

```
Get the public addresses of VMs created
terraform show
```

Edit config file under .ssh directory to bypass company proxy issues, an example of config file:

```
User gcpuser
ProxyCommand nc -X 5 -x proxy.XXX.com:1080 %h %p
```

Note: Only needed if ssh/scp is blocked by company proxy, get the proxy settings from your companies' IT.

Copy ssh key file to the VM to be set up as Control Plane node

```
scp .ssh/id_rsa gcpuser@public_IP_of_VM:~/.ssh
```

Copy CloudXPRT release package to the VM to be set up as Control Plane node

```
scp CloudXPRT_vX.XX_data-analytics.tar.gz gcpuser@public_IP_of_VM:~/
```

Note: Make sure you use your own connection string!

SSH into Control Plane node

```
ssh gcpuser@public_IP_of_VM
```

Install Kubernetes cluster

```
tar -xzf CloudXPRT_vX.XX_data-analytics.tar.gz
cd CloudXPRT_vX.XX_data-analytics/installation
ifconfig
modify cluster_config.json file with IP address shown with ifconfig
sudo ./prepare-cluster-CSP.sh
sudo ./create-cluster.sh
```

setup CloudXPRT data-analytics

```
cd CloudXPRT_vX.XX_data-analytics/setup/
# Execute the next command only if using a multi-node cluster
sudo ./cnb-analytics_OnPrem-MultiNode_setup.sh ## multi-node cluster only
# Execute the next command for all cluster types, single-node or multi-
node
sudo ./cnb-analytics_setup.sh
```

Run the benchmark:

A script is provided to create a swept analysis in order to find the best throughput under a particular SLA.

```
cd CloudXPRT_vX.XX_data-analytics/cnbrun/
sudo ./cnb-analytics_run-automated.sh
```

Note: Results will be written in the 'output' directory. **Note:** use cnb-analytics_clear.sh to reset kubernetes in case you have an invalid run. then re-issue ./cnbrun

Save results locally

On your local system

```
scp gcpuser@public_IP_of_VM:~/CloudXPRT_vX.XX_data-
analytics/cnbrun/output_* .
```

Clean up Cluster

After you are done running CloudXPRT and have saved the results:

```
terraform destroy
```

References:

- https://github.com/kubernetes/kops/blob/master/docs/getting_started/gce.md
- https://cloud.google.com/sdk/install
- https://cloud.google.com/storage/docs/gsutil
- https://www.cloudtechnologyexperts.com/kubernetes-google-cloud-kops/

Setup and Run on Azure

On local Ubuntu linux machine, create a new user then switch to this user

```
sudo adduser azureuser
sudo adduser azureuser sudo
```

If you are using GUI on the local Ubuntu machine, logout and log back in as azureuser. Otherwise, you can directly change over to the new user using the following command.

```
su - azureuser
```

Download Terraform binary and put it under /usr/local/bin directory

- https://www.terraform.io/downloads.html

Install Azure CLI

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

Note: If this step fails, refer to Microsoft Azure official documentation to install Azure CLI in an alternative way:

```
https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest
```

Authenticate with your Azure account

```
az login
az account list-locations
```

Create SSH key pairs

```
ssh-keygen
```

Create Virtual Machines (VMs)

```
tar -xzf CloudXPRT_vX.XX_data-analytics.tar.gz
cd CloudXPRT_vX.XX_data-analytics/terraform/azure
modify variables.tf file
terraform init
terraform apply
wait some time for it to finish (around 1-2 minutes)
```

Run CloudXPRT and save results

Access cluster

```
Get the public addresses of VMs created
terraform show
```

Edit config file under .ssh directory to bypass company proxy issues, an example of config file:

```
User azureuser
ProxyCommand nc -X 5 -x proxy.XXX.com:1080 %h %p
```

Note: Only needed if ssh/scp is blocked by company proxy, get the proxy settings from your companies' IT.

Copy ssh key file to the VM to be set up as Control Plane node

```
scp .ssh/id_rsa azureuser@public_IP_of_VM:~/.ssh
```

Copy CloudXPRT release package to the VM to be set up as Control Plane node

```
scp CloudXPRT_vX.XX_data-analytics.tar.gz azureuser@public_IP_of_VM:~/
```

Note: Make sure you use your own connection string!

SSH into Control Plane node

```
ssh azureuser@public_IP_of_VM
```

Install Kubernetes cluster

```
tar -xzf CloudXPRT_vX.XX_data-analytics.tar.gz
cd CloudXPRT_vX.XX_data-analytics/installation
ifconfig
modify cluster_config.json file with IP address shown with ifconfig
sudo ./prepare-cluster-CSP.sh
sudo ./create-cluster.sh
```

setup CloudXPRT data-analytics

```
cd CloudXPRT_vX.XX_data-analytics/setup/
# Execute the next command only if using a multi-node cluster
sudo ./cnb-analytics_OnPrem-MultiNode_setup.sh ## multi-node cluster only
# Execute the next command for all cluster types, single-node or multi-
node
sudo ./cnb-analytics_setup.sh
```

Run the benchmark:

A script is provided to create a swept analysis in order to find the best throughput under a particular SLA.

```
cd CloudXPRT_vX.XX_data-analytics/cnbrun/
sudo ./cnb-analytics_run-automated.sh
```

Note: Results will be written in the 'output' directory. **Note:** use cnb-analytics_clear.sh to reset kubernetes in case you have an invalid run. then re-issue ./cnbrun

Save results locally

On your local system

```
    scp azureuser@public_IP_of_VM:~/CloudXPRT_vX.XX_data-
analytics/cnbrun/output_* .
```

Clean up Cluster

After you are done running CloudXPRT and have saved the results:

```
    terraform destroy
```

Uninstall the benchmark

Uninstall components used in Cloud Analytics

```
    cd CloudXPRT_vX.XX_data-analytics/setup
    sudo ./cnb-analytics_cleanup.sh
```

To remove cluster and docker installation on every node, run the "remove-cluster.sh" script in the Control Plane node after you finish your runs.

```
    cd CloudXPRT_vX.XX_data-analytics/setup
    sudo ./remove-cluster.sh
```

Answer 'y' or 'yes' to the prompt.

Note: This will not remove the proxy settings. If you want to run CloudXPRT again, you can run the "create-cluster.sh" script to re-create the Kubernetes cluster. Reset Docker and Kubernetes cluster

Build the benchmark

Instructions for building the benchmark from source on Ubuntu 18.04

Download and install GO

```
    wget https://dl.google.com/go/go<version>.linux-amd64.tar.gz
    sudo tar -C /usr/local/ -xzf go<version>.linux-amd64.tar.gz
    echo "export PATH=$PATH:/usr/local/go/bin" >> $HOME/.profile
    source $HOME/.profile
```

Compile GO binaries and create release packages

```
- Create the release archive in directory "build" as file
CloudXPRT_vX.XX_data-analytics.tar.gz
```

```
    make build
```

Known Issues

FAQ

Q1. The benchmark is looping for long time while setting up any of the "pod/services"
- Please open a new console and use the script to visually verify there are no errors and/or pods in underfined status.

```
sudo ./cnb-analytics_status.sh
```

Q2. How to clean the current execution and start again ?
- Please remove all running processes follow the next steps:

```
ps -aux | grep cnbrun | grep automated (identify and kill corresponsing scripts)
sudo ./cnb-analytics_clear.sh
cd ../setup/setup_kafka
sudo ./cleanup.sh
sudo ./setup.sh
cd ../../cnbrun
sudo ./cnbrun
```

```
```