CloudXPRT Web Microservices Workload

Introduction

In the web-tier microservices workload, a simulated user logs in to a web application that does three things: provides a selection of stock options, performs Monte-Carlo simulations with those stocks, and presents the user with options that may be of interest. The workload reports performance in transactions per second, which testers can use to directly compare IaaS stacks and to evaluate whether any given stack is capable of meeting service-level agreement (SLA) thresholds.

CloudXPRT installation

Configure the benchmark

The installation scripts within the installation directory 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,
- configure 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 highly recommend running this benchmark on high end servers. While running, the benchmark will scale to utilize all the cores available. However, for functional testing, your physical node or VM must have at least:

- 16 logical or virtual CPUs
- 8GB Ram
- 10GB Disk Space

Installation steps

Configure the environment

1. 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_web-microservices/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 may 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", and configure the 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
```

Note:

- This process may take anywhere from 6 up to 20 minutes.
- 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 information 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 all the tests on this machine.

```
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.
```

Run the benchmark on-premises

Running CloudXPRT Web Microservices

```
cd CloudXPRT_vX.XX_web-microservices/cnbrun
```

Configure benchmark parameters

Edit config.json file to set the parameters for CloudXPRT.

- **runoption**: Run only a specific microservice or all of them individually (not in multitenancy)
- **iterations**: Number of iterations to run for each microservice
- **hpamode**: If true, use Kubernetes HPA to scale pods as load increases. Otherwise, create max pods from the beginning
- **postprocess**: If true, run the postprocess binary at the end of the run
- **ppoutputfile**: Sef if want post process results to be saved in a file
- **autoloader.initialclients**: The initial number of clients the load generator will create
- **autoloader.clientstep**: The number of clients to increase for each load generator iteration
- **autoloader.lastclient**: Stops the load generator when reaching this number of clients. If set to -1, it will continue to run until cluster is saturated (i.e. CPU ~100%)
- **autoloader.SLA**: Service Level Agreement for 95 percentile latency constraint. If set to -1, there is no latency constraint
- **autoloader.timeinterval**: The amount of time to spend within each load generator iteration with the specified amount of clients
- **workload.version**: Docker image version to use

- **workload.cpurequests**: Amount of CPU cores requested to assign to each pod (integer only)

Note: cnbrun, gobench, autoloader, and all shell scripts need to have executable permissions

Start the Benchmark Run

Running the load generator within the System Under Test (SUT)

Once parameters in config.json are configured, run the cnbrun executable.

```
./cnbrun
```

If benchmark run is interrupted in the middle, to clean up the resources generated

```
./cleanups.sh
```

To collect system information for the cluster, run the following script, on the Control Plane node only.

```
./system_info.sh
```

Running the load generator outside of the SUT

Requirement: The machine running the load generator must be on the same network as the Kubernetes cluster.

Copy the following directories from the Control Plane node of the cluster to the machine you want to run the load generator on:

1. cnbrun directory
2. $HOME/.kube directory

On the load generator machine:

1. Move the .kube directory to the user's home directory $HOME/
2. Install kubectl

   ```
   sudo apt-get install kubectl
   ```

3. Rename mc.remote.sh to mc.sh

   ```
   mv mc.remote.sh mc.sh
   ```

4. Ensure that the autoloader, cnbrun, cnbweb, gobench, and mc.sh within the cnbrun directory have executable permissions.

   ```
   chmod +x autoloader cnbrun cnbweb gobench mc.sh
   ```

5. Once parameters in config.json are configured, run the cnbrun executable.

   ```
   ./cnbrun
   ```

Benchmark Results

After a run, results will be written to the same node that ran the load generator. If you used a multi-node cluster, the other nodes within the cluster will not have any result files. The results will be written to the 'cnbrun/output' directory. These files do not get deleted or overwritten. They will accumulate in the output directory after each run.

For each run, you will have 4 files:

1. A log file with the results in a formatted table
2. A csv file with the results
3. A log file with all the stdout output during the run
4. A copy of the config file used for that run

Metrics

The results can be summarized using the following metrics:

1. Max successful requests per minute under a specified SLA
2. Total Max successful requests per minute (regardless of SLA)

Currently deriving the metric from the results is a manual process from the log file with the formatted table.

By default, the max Service Level Agreement (SLA) for 95 percentile latency is 3 seconds (specified in config.json's 'autoloader.SLA' parameter). The load generator will stop when either the system under test can no longer meet the specified SLA or when the SUT cannot beat the current maximum number of successful request within 5 retries.

Below are the condensed results from a run. You can choose different SLA's of interest from the logfile. For example, choosing SLA's as 1000ms, 2000ms, and 3000ms, we can compare the request rate that the SUT was able to consistently respond to within that time.

From the results, you can see that the system was able to handle: - 604 requests per minute under 1000ms, - 889 requests per minute under 2000ms, and - 900 requests per minute under 3000ms

The max throughput within this run is 900 successful requests per minute.

CONCURRENCY	SUCC_REQS	...	SUCC_REQS_RATE(REQ/S)	...	MC_RESP_TIME(95%ile)(MS)
1	50	...	1	...	607
2	98	...	3	...	667
...	...	...	...	...	...
17	587	...	19	...	942

18	604	..	20	..	987
19	624	..	20	..	1007
20	639	..	21	..	1093
...	...	..	...	..	...
49	884	..	29	..	1965
50	889	..	29	..	1994
51	889	..	29	..	2041
52	890	..	29	..	2189
...	...	..	...	..	...
59	897	..	29	..	2819
60	907	..	30	..	2706
61	900	..	30	..	2800
62	888	..	29	..	3032

Configure and run on AWS

Install Kubernetes cluster with KOPS on AWS and run CloudXPRT

References:

- https://medium.com/containermind/how-to-create-a-kubernetes-cluster-on-aws-in-few-minutes-89dda10354f4
- https://github.com/kubernetes/kops/blob/master/docs/getting_started/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

Preparation:

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 azureuser. Otherwise, you can directly change over to the new user using the following command.

```
su - awsuser
```

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 = AKIXXXXXXXXXXXXX
aws_secret_access_key = sIrkzNOXxXXXXXXXXXXXxxxXXXX
```

Create SSH key pairs

```
ssh-keygen
```

Create Virtual Machines (VMs)

```
tar -xzf CloudXPRT_vXXXX_web-microservices.tar.gz
cd CloudXPRT_vXXXX_web-microservices/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_vXXXX_web-microservices.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_vXXXX_web-microservices.tar.gz
cd CloudXPRT_vXXXX_web-microservices/installation
ifconfig
```

```
modify cluster_config.json file with IP address shown with ifconfig
sudo ./prepare-cluster-CSP.sh
sudo ./create-cluster.sh
```

Run CloudXPRT

```
cd CloudXPRT_vXXXX_web-microservices/cnbrun
```

Modify config.json file according to README in cnbrun directory

Run the benchmark:

```
./cnbrun
```

Note: Results will be written in the 'output' directory.

Save results locally

In your local machine, copy the results

```
scp ubuntu@public_IP_of_VM:~/CloudXPRT_vXXXX_web-
microservices/cnbrun/output/* .
```

Clean up Cluster

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

```
terraform destroy
```

Configure and run on Azure

Preparation:

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_vXXXX_web-microservices.tar.gz
    cd CloudXPRT_vXXXX_web-microservices/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_vXXXX_web-microservices.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_vXXXX_web-microservices.tar.gz
cd CloudXPRT_vXXXX_web-microservices/installation
ifconfig
modify cluster_config.json file with IP address shown with ifconfig
sudo ./prepare-cluster-CSP.sh
sudo ./create-cluster.sh
```

Run CloudXPRT

```
cd CloudXPRT_vXXXX_web-microservices/cnbrun
```

Modify config.json file according to README in cnbrun directory

Run the benchmark:

```
./cnbrun
```

Note: Results will be written in the 'output' directory.

Save results locally

In your local machine, copy the results

```
scp azureuser@public_IP_of_VM:~/CloudXPRT_vXXXX_web-
microservices/cnbrun/output/* .
```

Clean up Cluster

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

```
terraform destroy
```

Configure and run on GCP

Install Kubernetes cluster with KOPS on Google Cloud and run CloudXPRT

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/

Preparation:

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
```

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_vXXXX_web-microservices.tar.gz
cp your-service-account-key.json CloudXPRT_vXXXX_web-
microservices/terraform/gcp
cd CloudXPRT_vXXXX_web-microservices/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_vXXXX_web-microservices.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_vXXXX_web-microservices.tar.gz
cd CloudXPRT_vXXXX_web-microservices/installation
ifconfig
modify cluster_config.json file with IP address shown with ifconfig
sudo ./prepare-cluster-CSP.sh
sudo ./create-cluster.sh
```

Run CloudXPRT

```
cd CloudXPRT_vXXXX_web-microservices/cnbrun
```

Modify config.json file according to README in cnbrun directory

Run the benchmark:

```
./cnbrun
```

Note: Results will be written in the 'output' directory.

Save results locally

In your local machine, copy the results

```
scp gcpuser@public_IP_of_VM:~/CloudXPRT_vXXXX_web-
microservices/cnbrun/output/* .
```

Clean up Cluster

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

```
terraform destroy
```

Demo with UI

Instructions for running the benchmark in demo mode with UI

These scripts make it easy to bring up all of the services that are used during a normal CloudXPRT run. The main difference is that only one replica of each service is deployed and the services remain deployed until you want to remove them. It gives users time to interact with the web pages that the web server is serving.

Deploy all services

```
./services.sh up
```

Viewing Web Server Pages

Once all of the services are up, the script will print out possible address you can visit to interact with the front end. Example output from the script:

```
   You may access the web server UI by visiting one of the following
addresses in your web browser:
       http://10.233.47.78:8070 on any machine within the cluster, or
       http://192.168.0.11:30800 externally on any machine within the same
network
```

The second address printed is the ip address of the Control Plane node. If you have a multi-node cluster, you can access the web service by visiting the ip address of either node along with the same port number listed from the script.

To get the web-service ClusterIP address and ports exposed:

```
kubectl get service web-service
    NAME          TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)
AGE
    web-service   NodePort   10.233.47.78   <none>        8070:30800/TCP
14m
```

The ClusterIP address and port 8070 is only accessible from any nodes within Kubernetes cluster.

Externally, use the actual node's ip address along with the port listed within the 30000-32767 range.

Remove all services

```
./services.sh down
```

Build the benchmark from source

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

```
cd CloudXPRT-src/web-microservices
sudo apt install pkg-config libssl-dev -y
```

- Create the release archive in directory "build" as file CloudXPRT_vX.XX_web-microservices.tar.gz

```
make build
```