Reproducible and Scalable Experiments with SkyhookDM Ceph

Jayjeet Chakraborty
Carlos Maltzahn, Ivo Jimenez, Jeff LeFevre
UC Santa Cruz
Generic Systems Experimentation Workflow

**Build, Deploy and Run experiments**

Boot VMs or bare metal Nodes

Prepare plots and notebooks

Doing manually is time consuming and error-prone!
High-Level Workflow for SkyhookDM Experiments

- Spawn nodes and deploy k8s. Get the config.
- Benchmark Network and Disk I/O.
- Deploy Ceph and run Rados benchmarks.
- Benchmark throughput by running queries on hundreds of GBs of tabular data.
- Study the Jupyter notebooks, Plots and Grafana snaps.

Should be platform independent and automated!
If done manually,

```bash
# manually build ceph everytime and update all osd, mon, etc.
$ git clone git://github.com/ceph/ceph
$ ./install-deps.sh
$ ./do_cmake.sh
$ cd build
$ make
$ sudo make install

# benchmarking the blockdevices in every node. take care of fio version and the parameters.
$ fio --filename=/dev/sdb --rw=randwrite --direct=1 --ioengine=libaio --bs=64k --numjobs=8 --runtime=120
$ fio --filename=/dev/sdb --rw=randwrite --direct=1 --ioengine=libaio --bs=128k --numjobs=8 --runtime=120

# running rados bench multiple times with different params.
$ ceph osd pool create testpool 32 32 replicated
$ rados bench --no-hints -t 1 -p testpool 120 seq
$ rados bench --no-hints -t 8 -p testpool 120 seq
$ rados bench --no-hints -t 32 -p testpool 120 seq

# plotting results
$ python plot_fio.py
$ python plot_rados.py
```
Overview of Containers

- Less resource usage than VMs
- Platform independent and portable software
- Consistent operation across environments
- Greater efficiency
$ docker run -e BLOCKDEVICE=sdb
    -e IODEPTH=32
    -v $PWD:/workspace
    --rm
    --entrypoint /bin/bash
    -w /workspace
    bitnami/kubectl:1.17.4
    ./run_benchmarks.sh

Solves platform dependency.
But still lacks automation!
Introducing Popper

- Popper
- Containers
- Operating System
- Hardware
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftripplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "/.sweep/script.j2",
    "--space", "/.sweep/space.yml",
    "--output", "/.sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
“Popperizing” the SkyhookDM Experimentation Workflow

Development and testing environment

```
Development and testing only requires a Linux environment.
Linux machine (ideally running a 64-bit, 32-bit compatible, recent Linux kernel, such as Red Hat, with at least 32 GB of disk space and at least 16 GB of RAM.)
```

Linux desktop instructions

```
5. To use your own Linux machine, please go directly to the
   Docker instructions (Virtual Box or VirtualBox).
```

Docker instructions

```
1. Install Docker on your host machine. A Linux VPS host is not
   required for installing Docker on a Raspberry Pi.
2. On your host machine, create a directory for your script.
   In this example of 80 GB of storage space is left
3. The docker container will need access to all of your data
   in your host machine or your data must be mirrored
4. If your data needs to be shared
5. Use Docker to create a container with the following:
   Name the container, docker/cm docker container
6. Start the container and make it run
```

Build

```
To build the container with the following steps:
   1. Install Docker with the following:
     docker run -it --rm --name docker cm
   2. Build the container using Dockerfile
```

## setup kubernetes
### popper run -f kubernetes.yml

## baseline cluster
### popper run -f iperf/fio.yml

## deploy ceph/skyhookdm-ceph
### popper run -f rook.yml

## setup monitoring
### popper run -f prometheus.yml

## run rados benchmarks
## popper run -f radosbench.yml

## run experiment benchmarks
## popper run -f run_query.yml
Implementation Highlights

- Highly configurable and scalable workflows with parameter sweeps.
- Monitoring infrastructure with Prometheus + Grafana.
- Everything in Kubernetes. Rook.io, kube-prometheus, kubespray, kubestone, etc.
Workflows for Every High-level Step

```python
options:
env:
  KUBECONFIG: /kubeconfig/config
  BLOCKDEVICES: /block/devices
  PV_SIZE: 4GB
  IP:event’:32' 128K in 4KB
  IO_SIZE: 32
  IO_DEPTH: 32
  DISK_SIZE: 128
  ID: liboso
  NUM_HOSTS: 8
  Mode: read write roundread roundwrite
  TESTNAME: use_testname

steps:
  - id: bootstrap-config
    uses: docker://k8s/gcp-py3.10-rpm
    runs: [python]
    args: [./kubernetes_fio/scripts/bootstrap.py]
  - id: start
    uses: docker://bitkiy/kubectl:1.17.4
    runs: [bash, -eu]
    args:
    - |
      kubectl apply -f ./kubernetes_fio/pv.yml
      kubectl apply -f ./kubernetes_fio/pvc.yml
      kubectl apply -f ./kubernetes_fio/job.yml
  - id: run-benchmarks
    uses: docker://bitkiy/kubectl:1.17.4
    runs: [./kubernetes_fio/scripts/run_benchmarks.sh]
  - id: download-results
    uses: docker://bitkiy/kubectl:1.17.4
    runs: [./kubernetes_fio/scripts/download_results.sh]
  - id: plot-results
    uses: docker://jupyter/datascience-notebook:python-3.8.5
    runs: [jupyter]
    args: ['--ioconvert', '--execute', '--tornotebook', '/kubernetes_fio/notebook/plot.ipynb']
    options:
      ports: 8088/tcp:8888
  - id: teardown
    uses: docker://bitkiy/kubectl:1.17.4
    runs: [bash, -eu]
    args:
    - |
      kubectl delete -f ./kubernetes_fio/job.yml
      kubectl delete -f ./kubernetes_fio/pvc.yml
      kubectl delete -f ./kubernetes_fio/pv.yml

options:
env:
  KUBECONFIG: /kubeconfig/config
  NAMESPACE: kubernetes
  WRITEURATION: '128'
  READURATION: '128'
  THREADS: '1 0 32'
  DECENT_S": 128
  PG_SIZE: 128
  POOL_NAME: 'textbench'
  POOL_TYPE: 'replicated'
  REPLICATION_DISABLED: '1'
  CLIENT: $CLIENT

steps:
  - id: bootstrap-config
    uses: docker://k8s/gcp-py3.10-rpm
    runs: [python]
    args: [./radosbench/pyscripts/bootstrap.py]
  - id: start
    uses: docker://bitkiy/kubectl:1.17.4
    runs: [bash, -eu]
    args:
    - |
      kubectl apply -f "$NAMESPACE" -f ./radosbench/deployment.yml
  - id: copy-config
    uses: docker://bitkiy/kubectl:1.17.4
    runs: [./radosbench/pyscripts/copy_config.sh]
  - id: run-benchmarks
    uses: docker://bitkiy/kubectl:1.17.4
    runs: [./radosbench/pyscripts/run_benchmarks.sh]
  - id: plot-results
    uses: docker://jupyter/datascience-notebook:python-3.8.5
    runs: [jupyter]
    args: ['--ioconvert', '--execute', '--tornotebook', '/radosbench/notebook/plot.ipynb']
    options:
      ports: 8088/tcp:8888
  - id: teardown
    uses: docker://bitkiy/kubectl:1.17.4
    runs: [bash, -eu]
    args:
    - |
      kubectl delete -f "$NAMESPACE" -f ./radosbench/deployment.yml
```

[Diagram showing workflows for every high-level step]
Monitoring with Prometheus and Grafana
Jupyter Notebooks for further Exploration

```python
import os
import json
import matplotlib.pyplot as plt

results_dir = os.path.join('results', 'jupyter_notebook_results')

# iterate over each result file and plot the results
for file in files:
    if not file.endswith('.json'): continue
    with open(os.path.join(results_dir, file)) as f:
        results = json.load(f)
        seconds = []
        bandwidth = []
        for run in results['intervals']:
            seconds.append(float(run['sum'][:-1]))
            bandwidth.append(float(run['sum'][:-1]) / 10)
        plt.plot(seconds, bandwidth, marksize=10, linewidth=0.1)

plt.xlabel('Time (in seconds)')
plt.ylabel('Bandwidth (Gb/s)')
plt.title('IPERF3 Benchmarks')
plt.legend()
plt.savefig(os.path.join(results_dir, 'iperf3_benchmarks.png'))
plt.show()
```

Bandwidth Plots
Case Study: Benchmarking SkyhookDM on River SSL

- Baseline the River SSL Kubernetes cluster and performed performance benchmarks for SkyhookDM Ceph.
- Discovered bottleneck in Network I/O for 10GbE links.
- Discovered unbalanced CPU usage in OSDs due to unbalanced PGs.

New avenues for further investigation!
Future Work

● Try capturing and creating workflows for other categories of Ceph benchmarks like CephFS and Ceph RBD benchmarks.

● Create experiment/benchmark workflows for other popular systems e.g. Key-value stores like RocksDB, databases like PostgreSQL, etc.
Thank you!

Visit https://github.com/uccross/skyhookdm-workflows/

Questions?

jchakra1@ucsc.edu
https://twitter.com/heyjc25