Wicked fast PaaS: Performance tuning of OpenShift 3.5 and Docker 1.12

Red Hat OpenShift Engineering
Jeremy Eder and Mike Fiedler, 2017-05-03
Awesome titles such as ...
Choose Your Own Adventure®

kubernetes

etcd

HA Proxy

Red Hat® Enterprise Linux®

OpenShift

Gluster

Ansible
if [ "$containers" = "linux" ];
then
echo "fundamentals don't change"
fi
Subsystem Food Groups

- Fruits
- Grains
- Vegetables
- Protein
- Dairy

CPU
Memory
Storage
Network
Terminology Overview

Control Plane
- LB
- master and etcd
- master and etcd
- master and etcd

Infrastructure
- registry and router
- registry and router

Compute Nodes and Storage Tier

#redhat #rhsummit
Let’s just get this out of the way

● We’re talking about OCP 3.5 or later
● Slides:  https://www.slideshare.net/jeremyeder/
● Code:  https://github.com/openshift/svt
● There’s no video recording of this.
Deploying 2048 OpenShift nodes on the CNCF Cluster

OpenShift 3.5: Installation

[defaults]
forks = 20
gathering = smart
fact_caching = jsonfile
fact_caching_timeout = 600
callback_whitelist = profile_tasks

[ssh_connection]
ssh_args = -o ControlMaster=auto -o
ControlPersist=600s
control_path = %(directory)s/%%h-%%r
pipelining = True
timeout = 10

https://docs.openshift.org/latest/scaling_performance/install_practices.html
Docker Graph Driver: devicemapper vs overlay2

- RHEL 7.4
- SELinux
- Overlay2
Container Native Storage

- Dynamic
- Hyperconverged
- Scalable
- Performant

CNS Session Thu 11:30a, Rm 157A
Container Native Storage: StorageClasses

apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name: cnsclass
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://172.25.87.92:8080"
  restuser: "admin"

apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name: ec2class
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  zone: us-west-2b
Container Native Storage

- Dynamic
- Hyperconverged
- Scalable
- Performant

![Graph showing the number of bound PVs over time.](image)
Container Native Storage

AWS+EBS+io1 vs AWS+EBS+io1+CNS

- randread-4KiB_cns
- randread-4KiB_io1
- randread-64KiB_cns
- randread-64KiB_io1
distributed key value store that provides a reliable way to store data across a cluster of machines
OpenShift 3.5: etcd-3.1.x

- etcd-2.x limited node scalability
- etcd-3.x gets us to 2000+ nodes comfortably
- Image metadata moved from etcd to registry in 3.4.z and 3.5.

OpenShift 3.5: Image Metadata moved to Registry
OpenShift 3.5: etcd-3.1, storage mode v2, 5K projects

Quorum Reads added in OCP 3.4 for data safety
OpenShift 3.5: etcd-3.1.x, 1k proj/4k pods
OpenShift 3.5: Metrics

- Bump scalability limits 12,000 → 25,000 pods
- METRICS_DURATION=7, METRICS_RESOLUTION=30
- **Capacity Planning and Scalability docs**

![Diagram showing Heapster, Hawkular, and Cassandra]

- Heapster (collection)
- Hawkular (metrics)
- Cassandra (storage)
OpenShift 3.5: Logging (EFK)

- Logging Sizing Guidelines

systemd/Fluentd (collection) → Elasticsearch (storage) → Kibana (visualization)
OpenShift 3.5: Routing/Network Ingress Tier

- HAProxy-based ingress tier (haproxy runs as a pod)

```
projects:
  - num: 1
    basename: centos-stress
    ifexists: delete
    tuning: default
    templates:
      - num: 1
        file: ./content/quickstarts/stress/stress-pod.json
    parameters:
      - RUN: "wrk" # which app to execute inside WLG pod
      - RUN_TIME: "120" # benchmark run-time in seconds
      - PLACEMENT: "test" # Placement of the WLG pods based on a node's label
      - WRK_DELAY: "100" # maximum delay between client requests in ms
      - WRK_TARGETS: "^cakephp-" # extended RE (egrep) to filter target routes
      - WRK_CONNS_PER_THREAD: "1" # how many connections per worker thread/route
      - WRK_KEEPALIVE: "y" # use HTTP keepalive [yn]
      - WRK_TLS_SESSION_REUSE: "y" # use TLS session reuse [yn]
      - URL_PATH: "/" # target path for HTTP(S) requests
```
OpenShift 3.5: Routing/Network Ingress Tier

HTTP keepalive, TLS session reuse, 1 connection per thread

Requests per second

1kB

1p-mix-cpu*
1p-mix-cpu0
1p-mix-cpu1
1p-mix-cpu2
1p-mix-cpu3
1p-mix-mc10x-cpu*
2p-mix-cpu*
4p-mix-cpu*
2p-mix-cpu02

#redhat #rhsummit
OpenShift 3.5: Alpha Support for GPUs

- Works fine
- Mostly manual for now
- GA gated on finalizing resource management

https://blog.openshift.com/use-gpus-openshift-kubernetes/
Tooling
“I want an environment with thousands of deployments, pods (with persistent storage), build configurations, routes, services, secrets and more…”

http://sandeen.net/wordpress/uncategorized/coming-clean-on-o_ponies/
“I want an environment with thousands of deployments, pods (with persistent storage), build configurations, routes, services, secrets and more…”

http://sandeen.net/wordpress/uncategorized/coming-clean-on-o_ponies/
OpenShift Scalability Testing

- Cluster horizontal scale
  - # of nodes, # of running pods across all nodes
  - application traffic
- Node vertical scale
  - # of pods running on a single node
  - work that 1 node can support (applications, builds, storage)
- Application scalability
  - Scale # of application replicas up/down
OpenShift Performance Tests

- Resource usage/response times for scenarios
  - Application workload and access performance
  - Builds (OpenShift)
  - Metrics and Log collection

- OpenShift infrastructure performance
  - Resource usage of processes under load
  - Network (SDN) throughput
  - Routing
  - Storage (EBS, Ceph, Gluster, Cinder, etc)
Tools

- [https://github.com/openshift/svt](https://github.com/openshift/svt)
  - cluster load-up
  - traffic generation
  - concurrent builds, deployments, pod start/stop
  - reliability testing
  - network performance
  - logging and metrics tests
Cluster loader

- **cluster-loader** - python tool to quickly load clusters according to a **YAML** test specification.

- Can be used with Kubernetes or OpenShift

```yaml
projects:
  - num: 1000
    basename: nginx-explorer
    tuning: default
    templates:
      - num: 10
        file: cluster-loader/nginx.yaml
      - num: 20
        file: cluster-loader/explorer-pod.yaml
```
Demo
THANK YOU

plus.google.com/+RedHat

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHatNews