Application Performance Benchmarking for NVMe-oF Persistent Volume (On Kubernetes Cluster)

Pooja Pandey
Pranjali Malode
MSys Technologies, LLC
About MSys Technologies

Our WW Strength

800
And growing

Product Engineering Services
- Storage and Networking Engineering
- VMware Ecosystem Integration
- UX/UI, Enterprise Mobility
- Rapid Application Development
- AI, ML and Cognitive Services
- Fintech and Loyalty
- Contingent Hiring

Technology CoEs
- Storage CoE
- DevOps CoE
- QA Automation CoE
- Big data and Predictive Analytics CoE
- Digital Testing CoE
- Cloud CoE
- Open Source CoE

Outsourcing Partners to

Key Alliances

Hewlett Packard Enterprise
Chef
Rubrik
Google
IBM
Cloud Native Computing Foundation
Amazon Web Services
Docker
Chef
IBM Watson
Outline

• Scale-out Kubernetes cluster deployment
• Containers with Dynamic Persistent Storage Solution
• Application Pod Autoscaling
• Analysing Application performance benchmarking
Automated Kubernetes Cluster Deployment with GlusterFS and NVMe-oF Target
Scale Out Kubernetes Cluster Deployment

Precursor

- Separation between application data and application lifecycle management
  - VM running application with data on Virtual disks (FTP)
  - Back and restore of VM in case of failure (VMDK)

- Separate Storage and Compute in cloud environment
- Distinction between Container OS and app data using volume mounts
  - /var/www/http
  - /var/database
Scale Out Kubernetes Cluster Deployment

Precursor (Cont’d)

• Running applications in Containers as microservices (web app / db)

• To summarize
  • The bright side
  • High Velocity - updating app features without down time
Scale Out Kubernetes Cluster Deployment

Insights on Kubernetes Storage

• Kubernetes - An open-source system for automating deployment, scaling, and management of containerized applications
  • Immutability
  • Declarative configuration and
  • Self healing systems

• A Persistent Volume represents provisioned storage in the cluster
  • Need of persistent volumes
Scale Out Kubernetes Cluster Deployment

Insights on Kubernetes Storage (Cont’d)

Persistent volume claim (How do applications use Storage?)

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim
annotations:
    volume.beta.kubernetes.io/storage-class: "slow"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 15Gi
```
Scale Out Kubernetes Cluster Deployment

Insights on Kubernetes Storage (Cont’d)

• GlusterFS to back persistent volumes
• Advantages of persistent volumes
  • destruction of pod doesn’t affect data written
  • Limitation
  • Doesn’t help in prime time
Wow. That's a Lot of Self-managed Infrastructure for an Orchestration System

What we have to manage manually
Scale Out Kubernetes Cluster Deployment

Insights on Kubernetes Storage (Cont’d)

- Dynamic Persistent volume creation
- Heketi’s RESTful based volume management for GlusterFS
- How dynamic storage system works with Heketi
Scale Out Kubernetes Cluster Deployment

Insights on Kubernetes Storage (Cont’d)

Heketi's execution flow

- Heketi does the dynamic provisioning by first SSHing into the GlusterFS pod and creating a new logical volume.

- The logical volume is then loaded into GlusterFS as a brick.

- Heketi does this for us across all 3 nodes, creating 3 logical volumes and 3 bricks.

- These 3 bricks are then loaded into GlusterFS as a volume and that volume is then exposed to Kubernetes for use as a PV.
Scale Out Kubernetes Cluster Deployment

Deployment Flowchart

- Kubernetes to start a pod that runs application
- Kubernetes communicates with Heketi to satisfy the PVC request
  - Bare disk - foundation of Storage pyramid
  - GlusterFS - distributed, networked block storage system
Heketi fronted GlusterFS distributed storage
Scale Out Kubernetes Cluster Deployment

Automated Script for Deployment

1. Creates Kubernetes cluster with at least three nodes
2. Add disks to each worker node. (Manual)
3. Setup GlusterFS with Heketi deployment
4. Create storage class for GlusterFS volumes
5. Setup good to go
6. Deploy AUT to get performance benchmarking matrix
Application Scalability

• Scaling an application with increase or decrease in the number of replicas.
• Horizontal Auto Pod Scaler: Autoscaling based on CPU
  • Example: `[kube@k8s-master ~]$ kubectl autoscale deployment mongo - min=2 - max=7 - cpu-percent=60`
  • `horizontalpodautoscaler.autoscaling/mongo autoscaled`
Application Performance Benchmarking Reports on NVMe-oF Persistent Volume Attached to Kubernetes Cluster
# Performance Benchmarking

## Tool Used - YCSB (Yahoo Cloud Service Benchmarking)

### Core work load for YCSB

<table>
<thead>
<tr>
<th>WORKLOAD A</th>
<th>Update heavy workload: 50/50% Mix of Reads/Writes</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKLOAD B</td>
<td>Read mostly workload: 95/5% Mix of Reads/Writes</td>
</tr>
<tr>
<td>WORKLOAD C</td>
<td>Read-only: 100% reads</td>
</tr>
<tr>
<td>WORKLOAD D</td>
<td>Read the latest workload: More traffic on recent inserts</td>
</tr>
<tr>
<td>WORKLOAD E</td>
<td>Short ranges: Short range based queries</td>
</tr>
<tr>
<td>WORKLOAD F</td>
<td>Read-modify-write: Read, modify and update existing records</td>
</tr>
</tbody>
</table>
Performance Benchmarking

Tool Used - YCSB (Yahoo Cloud Service Benchmarking) (Cont’d)

- YCSB benchmark run on Mongo DB container
- Test workload - Workload A with approximately 3 million operations.
- Sample output from the test setup

**WORKLOAD A**
(Update heavy workload: 50/50% Mix of Reads/Writes)
Run the load phase. Choose a record count (Property ‘recordcount’) to insert into the database that is close to the number of operations you intend to run on it.
  - ‘load’ flag indicates that this is a load run.
  - ‘s’ flag prints status at 10 sec intervals
  - ‘recordcount’ is set to 3 million.
  - ‘threads’ sets the number of client threads.
  - ‘mongodb.auth’ is the property that we wrote to enable MongoDB authentication

**Command Used**
  - `./bin/ycsb load mongodb -s -P workloads/workloada -p recordcount=3000000 -threads 32 -p mongodb.url="mongodb://192.168.104.112:30001/dbname" -p mongodb.auth="true"`
MongoDB Benchmark Results

WORKLOAD A

Throughput  Read Latency  Write Latency

YCSB Throughput (ops/sec)

Number of Threads

Latency (usecs)
Performance Benchmarking

Tool used - FIO

FIO I/O performance benchmarking on Ubuntu container
REFERENCES


https://scalegrid.io/blog/how-to-benchmark-mongodb-with-ycsb/
About the Speakers

Pooja Pandey

Pooja holds over a decade of a distinguished career in Storage Quality Assurance and troubleshooting for world-class premium storage products. In her previous stints, she worked on EMC Storage, Oracle Pillar data systems, Sandisk, Cloud Velox, and Barclays products.

Pranjali Malode

Pranjali holds more than a decade of experience in Software Testing and QA implementation for Enterprise level, High-Performance Computing (HPC), Software-Defined Storage (SDS), Hyper-converged Infrastructure (HCI) storage solutions.
Thank You