Embracing Big Data Workload in Cloud Native Environment with Data Locality

Sammi Chen & Xiaoyu Yao
About us

• Sammi Chen – Senior software engineer from Tencent Cloud, Apache Hadoop Committer and PMC, working on Apache Hadoop HDFS/Ozone.

• Xiaoyu Yao – Principal software engineer from Cloudera, Apache Hadoop Committer and PMC, working on Apache Hadoop HDFS/Ozone.
• Big data evolution in cloud native environment
• Data Locality and why it matters
• Data locality in big data storage
  • Apache Hadoop HDFS
  • Apache Hadoop Ozone
• Apache Ozone in in Kubernetes
• Q&A
Big Data Evolution in Cloud Environment

• Co-located Compute and Storage

  • Pros
    • Fast storage access with locality
    • Less network traffic
    • Cost-effective for I/O intensive OLAP workloads (MapReduce/Hive/Impala)

  • Cons
    • Limited Elasticity: Requirements for Storage nodes and compute node are different.
    • Elastically scale storage node with compute node is difficult and may not cost-effective.
Big Data Evolution in Cloud Environment

- Separation of Compute and Storage
  - Pros
    - Elastically scale compute independent of storage
    - Cost-effective for compute-intensive workload (ML)
  - Cons
    - Lose storage locality
    - More network traffic for storage access
    - More CPU cycle (e.g., Erasure Coding)

YARN  
Network  
HDFS
• Hybrid-Cloud and Multi-Cloud
  • Pros
    • Support locality for I/O intensive workload
    • Allow agile access, e.g., ML on multi-cloud
  • Cons
    • Cost
    • Compatibility
Challenges of Cloud Native Env for Big Data

Scheduler
- Optimize resource utilization (CPU/GPU/Memory/etc.)

Networking
- Optimize bandwidth usage.

Storage
- Optimize external storage access?
Locality in Big Data

• What is Locality?

  **Local Node**: Data local to the compute node

  **Local Rack**: Data in the same rack with the compute node

  **Local DC**: Data in different rack/zone but closer to the compute node
Locality in Big Data - Benefit

• Higher throughput
• Less network traffic
• Fast job execution
• Better cluster utilization
Locality in Big Data - Scheduling

- Apache YARN
  - Support locality aware task scheduling via InputSplits

- Apache SPARK
  - Spark support locality aware scheduling of RDDs via spark.locality levels
    - PROCESS_LOCAL
    - NODE_LOCAL
    - RACK_LOCAL
    - ANY
  - Spark on K8s elastically schedule driver/executor pods with node selector
Locality in Big Data Storage (HDFS)

Apache Hadoop HDFS

- Scalable Distributed File System
  - Fast file system metadata access (200K ops/s)
  - Hundreds PBs in capacity
  - Thousands of nodes per cluster
  - Scale horizontally
  - Strong consistency
  - Resilient to failures
  - ...
Hadoop HDFS Locality
Rack aware access

/dw/hive/061209.dat
Block1
(DN1, DN2, DNN)

HDFS Client 1

Datanode 1

Datanode 2

Namenode

/dw/hive/061209.dat
Block1
(DNN, DN2, DN1)

HDFS Client 2

Datanode...

Datanode N

Rack1

Rack2
Hadoop HDFS Locality – Short Circuit Read

1. /dw/hive/061209.dat
   Block1 (DN1, DN2, DNN)

2. RequestShortCircuitRead

3. Open

4. Read with File Descriptor
Apache HDFS in Kubernetes

• Challenges
  • Monolithic namenode
    • Small Files problem
      • 300 million+ files need special GC tuning
      • Take long time to upgrade/restart
  • Expose datanode locality

• Opportunities
  • Cloud native storage orchestra
    • Existing big data workload: Analytic/IoT/Streaming, etc.
    • Upgradable from existing HDFS clusters with hundreds or thousands of nodes.
• Scalable, redundant, and distributed object store
• Scaling to billions of objects of varying sizes.
• Decoupled micro-services that support Kubernetes deployment.
• Support topology aware data placement and access.
• Support S3 access
• Support in-place upgrade from HDFS
• ...
Apache Ozone Topology

- Root
  - Data Center 1
    - Rack 1
      - Node Group 1
        - Datanode 1
        - Datanode...
        - Datanode...
  - Data Center X
    - Rack Y
      - Node Group M
        - Datanode...
        - Datanode...
        - Datanode N
Apache Ozone Locality

• Highly customizable topology schema
  • /DataCenter/NodeGroup/Rack/Datanode.

• Topology aware access policy
  • Ozone manager returns topology ordered datanodes list for client access
  • Client access block/chunks of the objects from the closest datanodes

• Topology aware placement policy
  • Trade-off between reliability and performance.
```yaml
# Customize Ozone Topology

```
Apache Ozone Topology aware Read

1. GetKey
2. Key/Block Locations sorted in distance (DN3, DN4, DN5)
3. Read Block/Chunks

Rack1
- DN1
- DN2

Rack2
- DN3
- DN4

Rack3
- DN5
- DN6
Apache Ozone Topology aware Write

1. PutKey
2. Allocate Block/Container
3. Key/Block Locations
4. Write Block/Chunks

Client

Rack1
DN1
DN2

Rack2
DN3
DN4
Raft Ring

Rack3
DN5
DN6

Ozone Manager (Namespace)

Storage Container Manager (Node, Block Management)
Apache Ozone in Kubernetes

OM

SCM

S3 Gateway

Datanodes
Apache Ozone S3 Gateway

- S3 Client
- HA Proxy
- Ozone Manager (Namespace)
- Storage Container Manager
- Rack1: S3 Gateway
  - DN1
  - DN2
- Rack2: S3 Gateway
  - DN3
  - DN4
- Rack3: S3 Gateway
  - DN5
  - DN6

Raft Ring
Apache Ozone in Kubernetes RoadMap

• Hadoop Compatible File System (Ozone-0.2.1)
  • Natively support Spark/Yarn/Hive

• Ozone S3 Gateway (Ozone-0.3.0)
  • Access Ozone via S3 API
  • Horizontally scalable with multiple stateless S3 gateways

• Ozone Deployment in Kubernetes (Ozone-0.4.0)
  • Customizable resource definition for ozone services
  • Support Spark + Ozone on Kubernetes
Apache Ozone in Kubernetes RoadMap

- Ozone CSI driver (Ozone-0.4.1)
  - Mount Ozone S3 bucket as CSI volume
  - Consumed as raw Kubernetes Storage

- Ozone Operator
  - Rook integration as storage provider in Rook
  - [https://github.com/rook/rook/issues/3235](https://github.com/rook/rook/issues/3235)
Apache Hadoop Ozone

https://hadoop.apache.org/ozone

Xiaoyu Yao
xyao@apache.org

Sammi Chen
sammichen@tencent.com