HDB/HAWQ Integration with Hadoop

Lili Ma  lma@pivotal.io
Agenda

- HDB/HAWQ Overview
- Storage Integration
- Resource Management Integration
- User Authorization Integration
- Future Work
HDB/HAWQ Overview
Components Interactive with Hadoop

● **Storage**
  – HDFS Catalog Cache vs. Libhdfs3
  – Parquet
  – PXF
  – InputFormat/OutputFormat
  – Hawq extract/register

● **Resource Management**
  – Standalone Resource Management vs Yarn Managed RM
  – LibYarn

● **User Authorization**
  – Ranger
Storage
Data Access Layer

HDFS Catalog -- Cache

- HDB Metadata:
  - Catalog Table -> schema & hdfs file name
  - HDFS NameNode -> Block information for each hdfs file
- HAWQ master connects to HDFS Namenode to fetch block information of HDFS files
- The block information may be huge for large table -> performance downgrade if fetching every time
- HDFS Catalog Cache → Store previous block information & LRU Replacement Policy

HDFS Data -- Libhdfs3

- How to access data in HAWQ(C) from HDFS (Java)?
- Libhdfs
  - JNI based C language library
  - Users must deploy HDFS jars on every machine to use it
- Libhdfs3
  - native Hadoop RPC protocol and HDFS data transfer protocol
  - lightweight, small memory footprints
  - Easy to use and deploy
Parquet Storage

- **HAWQ Design for Parquet**
  - Do not change anything in open source Parquet format
  - Append to a file and add a new footer to file at the end of load/insert
  - Design point for Parquet is for large writes

- **DDL**
  - `create table a(a int, b int) with(appendonly=true, orientation=parquet, compressstype=snappy);`
PXF Framework

External Tables

REST API
HTTP, port: 51200

Apache Tomcat

PXF Webapp

Native Tables

libhdfs3 (written in C) segments

Java API

Java API

Java API

Java/Thrift
Architecture - Read Data Flow

0. `select * from ext_table`

8. query result

1. `getFragments()` API

3. Assign Fragments to Segments

2. Fragments (JSON)

4. Query dispatched to Segment 1, 2, 3... (Interconnect)

5. Read() REST

6. records

7. Records (stream)

HAWQ

Master Node

Fragmenter

DN1

HAWQ seg1

Accessor

Resolver

pxf://<location>:<port>/<path>
HAWQInputFormat/HAWQParquetOutputFormat

- **Purpose**
  - HAWQ can work align with other products in Hadoop eco-system

- **HAWQInputFormat**
  - Easy for others to read data generated in HAWQ
  - Get key as Void type, value as HAWQRecord type -> get each hawq record in HAWQRecord struct -> record.getInt(index)
  - Supports both AO table and Parquet table

- **HAWQParquetOutputFormat**
  - Extension to ParquetOutputFormat, specifying type to be HAWQRecord
  - Provides an interface setHAWQSschema for others to specify HAWQ schema
  - Other system can generate HAWQRecord, and thus can write the data
HAWQ Extract/HAWQ Register

- **HAWQ Extract**
  - Extract out metadata & HDFS file location for the table to yaml configuration file
  - Yaml configuration can be used by HAWQInputFormat

- **HAWQ Register**
  - Register existing files on HDFS directly to HAWQ internal table
  - Scenario
    - Register other systems generated data
    - HAWQ cluster migration
  - Usage
    - `hawq register [-h <hostname>] [-p <port>] [-U <username>] -d <databasename> -f <hdfspath> <tablename>`
    - `hawq register [-h <hostname>] [-p <port>] [-U <username>] -d <databasename> -c <configFile> <tablename>`
Resource Management
HAWQ Resource Manager Highlights

- Hierarchical resource queues (DDL)
- Automatic Resource Allocation
- Resource Allocation policy at queue and statement level
- Global optimized resource allocation: HAWQ makes global optimized resource allocations across the cluster
- Pluggable global resource manager (two modes: None/YARN)
- Dynamic resource expansion/shrink and segment profiling
- High volume concurrent query execution & low resource allocation latency
Interaction with Yarn

LibYarn

HAWQ Resource Manager (Application Master)

YARN Resource Manager

Yarn Node Manager

- register/unregister
- allocate/release resource
- get cluster/container reports
- get queue information

resource track and schedule

active
User Authorization
Background

- **Ranger: A Global User Authorization Tool in Hadoop eco-system**
  - Can support multiple systems such as HDFS, Hive, HBase, Knox, etc.
  - Provides a central UI for user to defining policies for different systems
  - Provide a base Java Plugin thus feasible for other products to define its own plugin to be controlled by Ranger

- **HAWQ Current ACL**
  - Implement through Grant/Revoke SQL Command
  - Current ACL is controlled by catalog table, which is stored in HAWQ master

- **HAWQ needs to keep align with hadoop eco-systems, so we need integrate with Ranger ACL**
  - Provide a GUC specifying whether enable ranger as ACL check
  - Once ranger is configured, move all the ACL check to Ranger side
  - Define all the policies in Ranger
HAWQ Ranger Integration

 LDAP Server

Create user in LDAP Server

User synced to HAWQ

User synced to Ranger

HAWQ

Send ACL Check request

Ranger REST Service

HAWQ Ranger Plugin

Ranger Admin Server

Policy sync

Ranger Policy DB
Workflow

1. create user
2. sync user information
3. sync user info
4. define policy
5. store policy
6. fresh policy
7. send query
8. send ACL check
9. return result
10. return
Components

● HAWQ Ranger Plugin
  – An extension to Ranger plugin, providing functions including
    – Register itself into Ranger Server
    – Sync Ranger defined policies to plugin itself
    – Lookup Service from Ranger Server to HAWQ

● Ranger Plugin Service
  – A RESTful Service which includes HAWQ Ranger Plugin
  – Provide API of checkPrivilege for HAWQ ACL

● HAWQ ACL
  – Encapsulate ACL check to a JSON Request, and send to RPS
    – Merge the ACL check inside one query as a single JSON Request
    – Request includes three parts information: requestor; resource; privileges
Future Work
TDE (Transparent Data Encryption) Support

- **TDE**: HDFS implements transparent, end-to-end encryption
  - Data is transparently encrypted and decrypted without requiring changes to user application code
  - Data can only be encrypted and decrypted by the client
  - HDFS never stores or has access to unencrypted data or unencrypted data encryption keys

- **HAWQ Enhancement**
  - Modify libhdfs3 to add support for TDE
Parquet 2.0 Support

- **Parquet 2.0 Enhancement**
  - Add more Converted Type: Enum, Decimal, Date, Timestamp
  - Add more statistics in DataPageHeader: including max/min/null count, distinct count
  - Add Dictionary Page
  - Add sorting column information in Rowgroup meta
  - ...

- **HAWQ Upgrade to Parquet 2.0 support**
  - Bring performance improvement by leveraging statistics information
  - Become more compatible with other systems which have supported Parquet 2.0
Thanks
Pivotal
Transforming How The World Builds Software