Apache HAWQ Resource Management and Its Integration with YARN

Yi Jin  Pivotal Inc.
yjin@pivotal.io

Apache Hadoop Native SQL. **Advanced**, MPP, elastic query engine and **analytic database** for **enterprises**. Now **incubating** with Apache.
• Apache HAWQ Brief Introduction

• Apache HAWQ Resource Manager High Level Design

• Key Implementation of Apache HAWQ Resource Manager

• Apache HAWQ Integration with YARN

• Query Performance Considerations
History of Apache HAWQ

2011
Prototype

2012
HAWQ 1.0
Alpha

2013
HAWQ 1.0
GA

2013 - 2014
HAWQ 1.1
HAWQ 1.2
HAWQ 1.3

2015 -
HAWQ 2.0
Apache Incubation
http://hawq.incubator.apache.org

Apache Hadoop Native SQL. Advanced, MPP, elastic query engine and analytic database for enterprises. Now incubating with Apache.
Apache HAWQ High-Level Architecture

- **YARN Resource Manager**
- **Catalog service**
- **HAWQ Master**
- **HDFS NameNode**
- **HDFS DataNode**

- **YARN Node Manager**
  - **Container**
  - **QE**
  - **Segment**

- **YARN Node Manager**
  - **Container**
  - **QE**
  - **Segment**

- **YARN Node Manager**
  - **Container**
  - **QE**
  - **Segment**

- **YARN Node Manager**
  - **Container**
  - **QE**
  - **Segment**
High-Level Flow of Query Execution

In case running in YARN mode

- YARN
- containers

Resource Manager

query resource

Parser

Parse tree

Planner

plan

Dispatcher

slice and resource quota

Executor

result

QD

QE

© Copyright 2013 Pivotal. All rights reserved.
• Apache HAWQ Brief Introduction

• Apache HAWQ Resource Manager High Level Design

• Key Implementation of Apache HAWQ Resource Manager

• Apache HAWQ Integration with YARN

• Query Performance Considerations
Apache HAWQ Resource Manager Technical Highlights

- Easy to define hierarchical resource queues through DDL.
- Customizable resource allocation policies: Resource allocation is completely automatic and the resource allocation policy can be customized at queue and statement level.
- Support high volume concurrent query execution with low resource allocation latency.
- Fine grained resource management. HAWQ can manage resources at operator level.
- Global optimized resource allocation. HAWQ makes global optimized resource allocations across the cluster.
- Pluggable global resource manager. Currently, standalone and YARN modes are supported, it is easy to support new global resource manager (Mesos for example).
Apache HAWQ Resource Concepts

Small granularity, High frequency resource negotiation between QD and Apache HAWQ RM.

Large granularity, Low frequency resource negotiation between Apache HAWQ and GRM server.

One Virtual Segments (VSEG) are allocated containing query resource for a group of QEs. Resource granularity is usually small, 256MB for example.

Global resource manager (GRM) containers are assigned to one HAWQ segment. Resource granularity is usually as large as GBs per container. YARN is considered as one GRM.

HAWQ Segment manages the resource consumable for QEs.

One host has at most one segment deployed for one HAWQ cluster.
Apache HAWQ Resource Manager Components

- Segment Resource Quota Manager
- Segment Request Handler Server
- QD Client
- Request Handler Server
- Resource Context Manager
- Resource Queue Manager
- Dynamic Resource Pool
- Fault Tolerance Service
- Standalone mode Resource Broker
- YARN mode Resource Broker
- Sync/Async RPC Framework
- libYARN
- TBA Resource Broker

- Resource manager Segment side
- Each QD
- Resource manager Master side
Resource Inter-Process Communication

**QD**
- QD acquires and returns query resource as a list of virtual segment.
- QD dispatches plan with query resource quota to QEs.
- QEs pass back query result.
- QD Client
- Resource manager acquires YARN cluster report, container report, and activated YARN container list.
- Resource broker registers as un-managed application and negotiates resource with YARN.
- Resource broker activates YARN containers and gets container status.
- Through RPC
- Through interconnect
- Through libYARN

**QE**
- Enforce resource usage according to the quota.

**SEG Resource Manager**
- Segment side resource manager reports segment status.

**Resource Manager**
- Resource manager increases or decreases resource quota in segment.

**YARN Resource Manager**
- Resource broker registers as un-managed application and negotiates resource with YARN.

**YARN Resource Broker**
- Resource broker activates YARN containers and gets container status.

**YARN Node Manager**

In case running in YARN mode
• Apache HAWQ Brief Introduction
• Apache HAWQ Resource Manager
  High Level Design
• Key Implementation of Apache HAWQ
  Resource Manager
• Apache HAWQ Integration with YARN
• Query Performance Considerations
Apache HAWQ Resource Queues

### DDL for manipulating resource queues

- **CREATE RESOURCE QUEUE** `queue_name` WITH (queue_attr_list);
- **ALTER RESOURCE QUEUE** `queue_name` WITH (queue_attr_list);
- **DROP RESOURCE QUEUE** `queue_name`;

`queue_attr_list ::= queue_attr=value | queue_attr=value,queue_attr_list`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PARENT</strong></td>
<td>The parent queue's name.</td>
</tr>
<tr>
<td><strong>ACTIVE_STATEMENTS</strong></td>
<td>The limit of number of parallel active statements. Resource queue's concurrency control.</td>
</tr>
<tr>
<td><strong>MEMORY_LIMIT_CLUSTER</strong></td>
<td>The percentage of memory resource consumable from its parent queue's capacity.</td>
</tr>
<tr>
<td><strong>CORE_LIMIT_CLUSTER</strong></td>
<td>The percentage of virtual core resource consumable from its parent queue's capacity. <em>Currently, this value must be identical with MEMORY_LIMIT_CLUSTER.</em></td>
</tr>
<tr>
<td><strong>ALLOCATION_POLICY</strong></td>
<td>The resource allocation policy, currently, only 'even' policy is supported.</td>
</tr>
<tr>
<td><strong>VSEG_RESOURCE_QUOTA</strong></td>
<td>The resource quota of one virtual segment (VSEG).</td>
</tr>
<tr>
<td><strong>RESOURCE_OVERCOMMIT_FACTOR</strong></td>
<td>The factor of overusing resource from the other resource queues.</td>
</tr>
<tr>
<td><strong>NVSEG_UPPER_LIMIT</strong></td>
<td>The absolute maximum number of VSEGs for one statement. Default value 0.</td>
</tr>
<tr>
<td><strong>NVSEG_LOWER_LIMIT</strong></td>
<td>The absolute minimum number of VSEGs for one statement. Default value 0.</td>
</tr>
<tr>
<td><strong>NVSEG_UPPER_LIMIT_PERSEG</strong></td>
<td>NVSEG_UPPER_LIMIT_PERSEG times available segment count derives the absolute maximum number of VSEGs for one statement. Default value 0.0.</td>
</tr>
<tr>
<td><strong>NVSEG_LOWER_LIMIT_PERSEG</strong></td>
<td>NVSEG_LOWER_LIMIT_PERSEG times available segment count derives the absolute minimum number of VSEGs for one statement. Default value 0.0.</td>
</tr>
</tbody>
</table>

*The attributes must be specified when creating a resource queue.*
Tree-structured Resource Queues

Resource queues’ capacities are automatically estimated based on dynamic cluster resource capacity and resource queue DDL manipulations. Statements consume resource from leaf queues.

```
CREATE RESOURCE QUEUE pg_depl WITH(
    PARENT='pg_root',
    ACTIVE_STATEMENTS=100,
    VSEG_RESOURCE_QUOTA='mem:512mb',
    OVERCOMMIT_FACTOR=1
);

ALTER RESOURCE QUEUE pg_default WITH(
    MEMORY_LIMIT_CLUSTER=20%,
    CORE_LIMIT_CLUSTER=20%
);

CREATE ROLE demorole RESOURCE QUEUE pg_depl;

SET ROLE demorole;

SELECT COUNT(*) FROM foo;

ALTER RESOURCE QUEUE pg_depl WITH(
    MEMORY_LIMIT_CLUSTER=80%,
    CORE_LIMIT_CLUSTER=80%,
    OVERCOMMIT_FACTOR=2
);
```
Assign Resource Among Resource Queues

Apache HAWQ assign globally allocated resource among resource queues according to queue capacities. Branch resource queues are not considered. Idle leaf resource queues are not considered.

<table>
<thead>
<tr>
<th>Case</th>
<th>pg_default weight</th>
<th>ptest weight</th>
<th>depart1adhoc weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>All busy</td>
<td>50%</td>
<td>20%</td>
<td>30%</td>
</tr>
<tr>
<td>Only ptest idle</td>
<td>62.5%</td>
<td>0%</td>
<td>37.5%</td>
</tr>
<tr>
<td>Only pg_default busy</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Only ptest busy</td>
<td>0%</td>
<td>20%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Estimate Resource Size Following Resource Queue Definition

In one resource queue, Apache HAWQ resource manager automatically:

- calculates consumable resource for one statement,
- applies global resource consumption restrictions,
- applies resource queue level resource consumption restrictions,
- evenly dispatches resource to concurrent waiting requests,
- selects as many as possible segments to contain VSEGs,
- evenly consumes segment resource when allocating VSEGs among them,
- applies statement level resource consumption setting if applicable.

**Diagram Description:**

1. **Resource request from QD**
   - Expected range of number of VSEGs
   - Data locality
   - Statement level VSEG quota
   - Session level GUCs for number of VSEGs

2. **Apply resource queue definition**

3. **Apply resource queue allocation policy**

4. **Allocate VSEGs from segments in resource pool**

5. **Resource to be sent to QD**
   - VSEG quota
   - List of VSEGs

6. **Check resource distribution of allocated VSEGs**

7. **Push into target resource queue’s waiting list and control concurrency**

8. **Not accepted resource is returned and pushed back to wait again.**
Dynamic Extensible Cluster and Segment Profiling

Apache HAWQ cluster can be easily extended by starting up more segments
- Apache HAWQ segment can be independently installed, configured and started to join a running HAWQ.
- Apache HAWQ segments register themselves by actively sending their profile out.
- Resource Pool and FTS in resource manager automatically maintain a complete profile of whole cluster.
- Resource Manager automatically adjusts resource queue capacity according to the change of cluster capacity.

One Segment’s profile describes
- Host configuration: name, ip addresses.
- Segment configuration: temp directories.
- Segment availability
- Segment global resource containers
- Segment resource allocation and usage.

• Apache HAWQ Brief Introduction

• Apache HAWQ Resource Manager High Level Design

• Key Implementation of Apache HAWQ Resource Manager

• Apache HAWQ Integration with YARN

• Query Performance Considerations
Configure to Integrate Apache HAWQ with YARN

- YARN configuration required by Apache HAWQ
  - Capacity Scheduler is required
  - Assign one queue exclusively to Apache HAWQ

- Configure `hawq-site.xml` to enable yarn mode and let Apache HAWQ connect to YARN

<table>
<thead>
<tr>
<th>Property</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hawq_global_rm_type</code></td>
<td>Apache HAWQ global resource manager type that indicates Apache HAWQ runs in corresponding modes. ‘yarn’ means enabling yarn mode.</td>
</tr>
<tr>
<td><code>hawq_rm_yarn_address</code></td>
<td>YARN resource manager address including.</td>
</tr>
<tr>
<td><code>hawq_rm_yarn_scheduler_address</code></td>
<td>YARN scheduler address.</td>
</tr>
<tr>
<td><code>hawq_rm_yarn_queue_name</code></td>
<td>YARN queue name that used by Apache HAWQ resource manager.</td>
</tr>
<tr>
<td><code>hawq_rm_yarn_app_name</code></td>
<td>Apache HAWQ YARN application name.</td>
</tr>
</tbody>
</table>
A Brief Introduction to libYARN

- libYARN is a c/c++ library that assists application to register to and unregister from Hadoop YARN, to allocate from and release resource to YARN.
- libYARN provides a synchronous way of requesting resource from YARN.
- libYARN communicates with YARN via protobuf RPC.
- libYARN supports Kerberos for connection security.
- libYARN supports YARN resource manager HA. When libYARN fails to connect YARN active resource manager, it automatically tries to connect YARN standby resource manager.
Apache HAWQ in YARN Mode

- Apache HAWQ forks resource broker process to register itself as a YARN application to YARN Resource Manager.
- Resource broker process acts as an **unmanaged application master**. An unmanaged application master is managed outside of YARN resource manager. There isn’t a container launched for application master.
  - A thread in resource broker process is created sending heartbeat to YARN resource manager on behalf of Apache HAWQ to keep registered application active and alive.
  - QE’s are running on segments with Node Manager running. If a HAWQ segment has no Node Manager running, it will be marked as DOWN by HAWQ and will not be used by resource pool.
  - Apache HAWQ activates YARN containers by dummy processes once getting the allocated container list.
  - Apache HAWQ enforces resource usage in all segments by itself.
A step-by-step from start HAWQ and execute a SQL statement without resource returning

1. Register HAWQ as an application exclusively consuming a YARN queue
2. Periodically fetch YARN cluster report, container report and queue report to recognize YARN cluster
3. Acquire YARN containers with host preference information
4. Return YARN containers
5. Unregister HAWQ in YARN

HAWQ resource manager

- Drive resource broker to acquire global resource manager containers. The quota of a global resource manager can be (1GB,1core), (2GB, 1core), etc.

HAWQ query dispatcher

- Acquire/Return query resource

HAWQ resource pool

- Indexed resource quota table
- Accepted YARN containers' quota
- To be returned YARN containers' quota

Global resource manager container lifecycle manager

- Increase HAWQ segment resource quota when have new global resource manager's containers allocated;
- Decrease HAWQ segment resource quota when some global resource manager's containers are decided to be kicked.

Allocation of virtual segments with fixed resource quota assigned and dispatch workload to segments. The resource quota can be as small as 128MB, 256MB and as large as GBs.
Friendly Apache HAWQ YARN Resource Return

Apache HAWQ resource manager friendly uses YARN allocated resource, which makes it flexibly co-exist with another YARN applications

- Only acquire resource on-demand
- Configurable minimum water-level of holding resource
- Pause allocating query resource and return YARN containers when resource manager overuses YARN queue and it finds the YARN queue is busy
- Return resource when resource in-use water mark becomes lower
• Apache HAWQ Brief Introduction
• Apache HAWQ Resource Manager
  High Level Design
• Key Implementation of Apache HAWQ
  Resource Manager
• Apache HAWQ Integration with YARN
• Query Performance Considerations
On-demand Quantity of VSEGS

- Apache HAWQ expects only on-demand resource to run query which is calculated through query plan
  - Avoid high network communication cost in large scale cluster
  - Improve resource efficiency according to query workload
  - Improve concurrency capacity

- Apache HAWQ resource manager allocates only on-demand resource
  - Follow global GUC variables to control resource consumption
    - `hawq_rm_nvseg_perquery_limit`
    - `hawq_rm_nvseg_perquery_perseg_limit`
  - Follow session-level resource consumption setting to allocate resource
    - `set hawq_rm_stmt_nvseg=10;`
    - `seg hawq_rm_stmt_vseg_memory=1gb;`
    - `select * from t;`
    - `set hawq_rm_stmt_nvseg=0;`
  - Follow resource queue definition including concurrency setting, VSEG resource quota, limits of number of VSEGS
    - `ACTIVE_STATEMENTS`
    - `VSEG_RESOURCE_QUOTA`
    - `NVSEG_UPPER_LIMIT/NVSEG_LOWER_LIMIT`
    - `NVSEG_UPPER_LIMIT_PERSEG/NVSEG_LOWER_LIMIT_PERSEG`
  - Follow resource expectation from query plan
Allocating VSEGs According to HDFS Data Locality

- Apache HAWQ holds target table’s data locality information that is passed to resource manager when acquiring query resource.

- Apache HAWQ resource manager allocates VSEGs among segments if it is appropriate to follow data locality information.
  - It works for small scale queries requiring a few VSEGs only
  - If a lot of VSEGs are required for a large scale query execution, this strategy is not adopted
  - If the VSEG allocation intensifies the balance of resource usage among segments too much, this strategy is not adopted
Even Resource Consumption among Segments

- Apache HAWQ resource manager tries its best to negotiate with global resource manager (YARN for example) to get evenly allocated resource
  - Preferred hosts are passed to global resource manager to acquire containers
  - When Apache HAWQ resource manager decides to return containers, the segments having most allocated containers are considered preferentially

- Apache HAWQ resource manager tries its best to allocate VSEGS among segments to make all available segments even utilized
  - Combined workload ordered index is dynamically maintained in resource pool so that the segments having lower workload and more available resource are considered in priority
  - Round-robin strategy is adopted generally to ensure that for one query execution, all possible available segments are utilized, and the segments have almost the same number of VSEGS
  - Data-locality based VSEG allocation strategy is restricted in some cases
Apache Hadoop Native SQL. Advanced, MPP, elastic query engine and analytic database for enterprises. Now incubating with Apache.