Help Build the most Advanced SQL Database on Hadoop: **Apache HAWQ**

Lei Chang
lchang@pivotal.io

Apachecon 2015, Budapest, Europe

Pivotal

- Introduction
- Architecture
- Query processing
- Catalog service
- Virtual segments
- Interconnect
- Transaction management
- Resource management
- Storage
- How to contribute

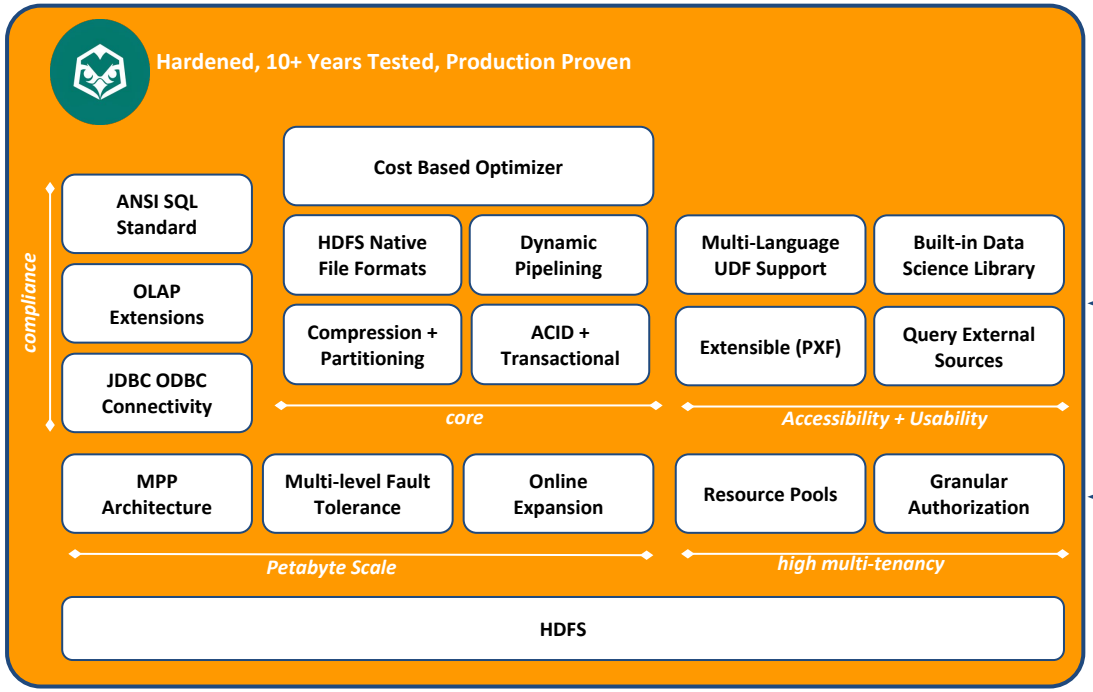
Pivotal

HAWQ: A Hadoop Native Parallel SQL Engine

Apache HAWQ

- Leverage Existing Skills & Tools
- Easily Integrate with Other Tools

- Well Integrated with Hadoop Ecosystem



- Discover New Relationships
- Enable Data Science
- Analyze External Sources
- Query All Data Types!

- Manage Multiple Workloads
- Petabyte Scale Analytics
- Sub-second Performance

Pivotal

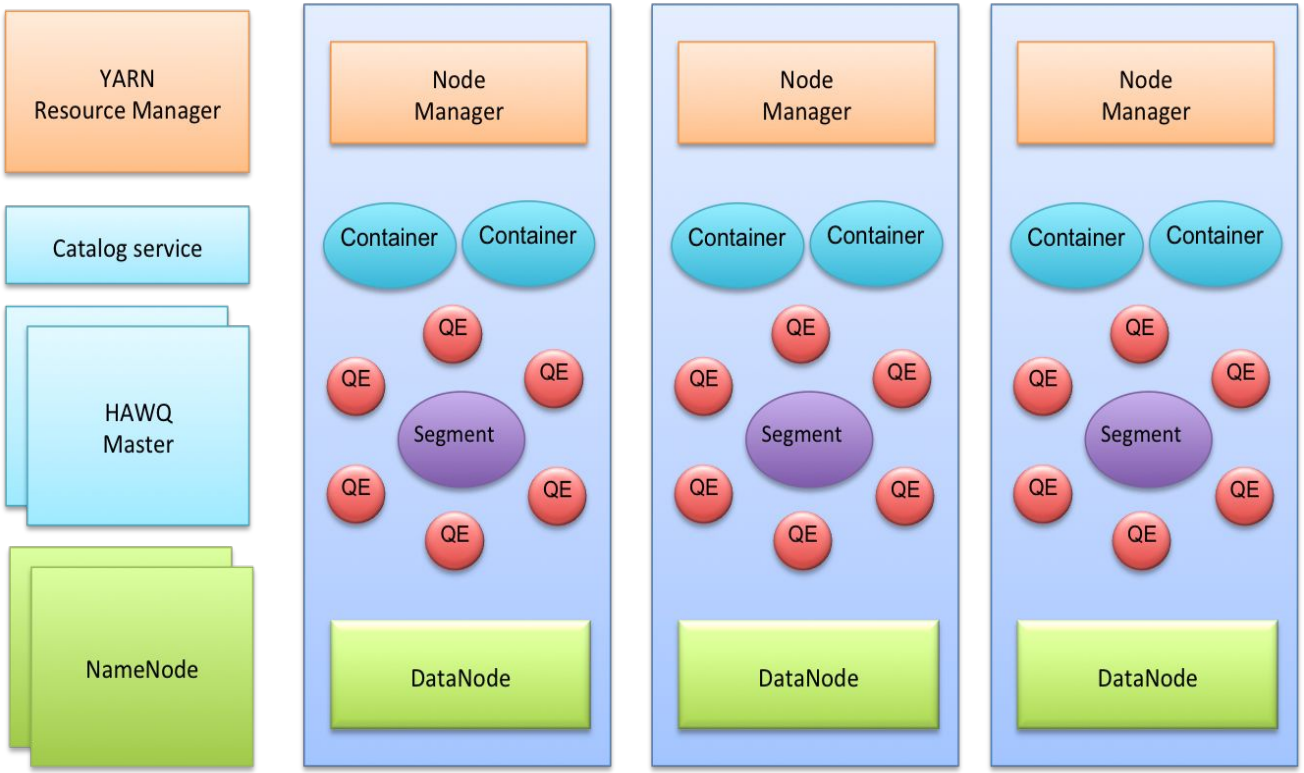
- Interactive queries
- Scalability
- Consistency
- Extensibility
- Standard compliance
- Productivity

Pivotal

History

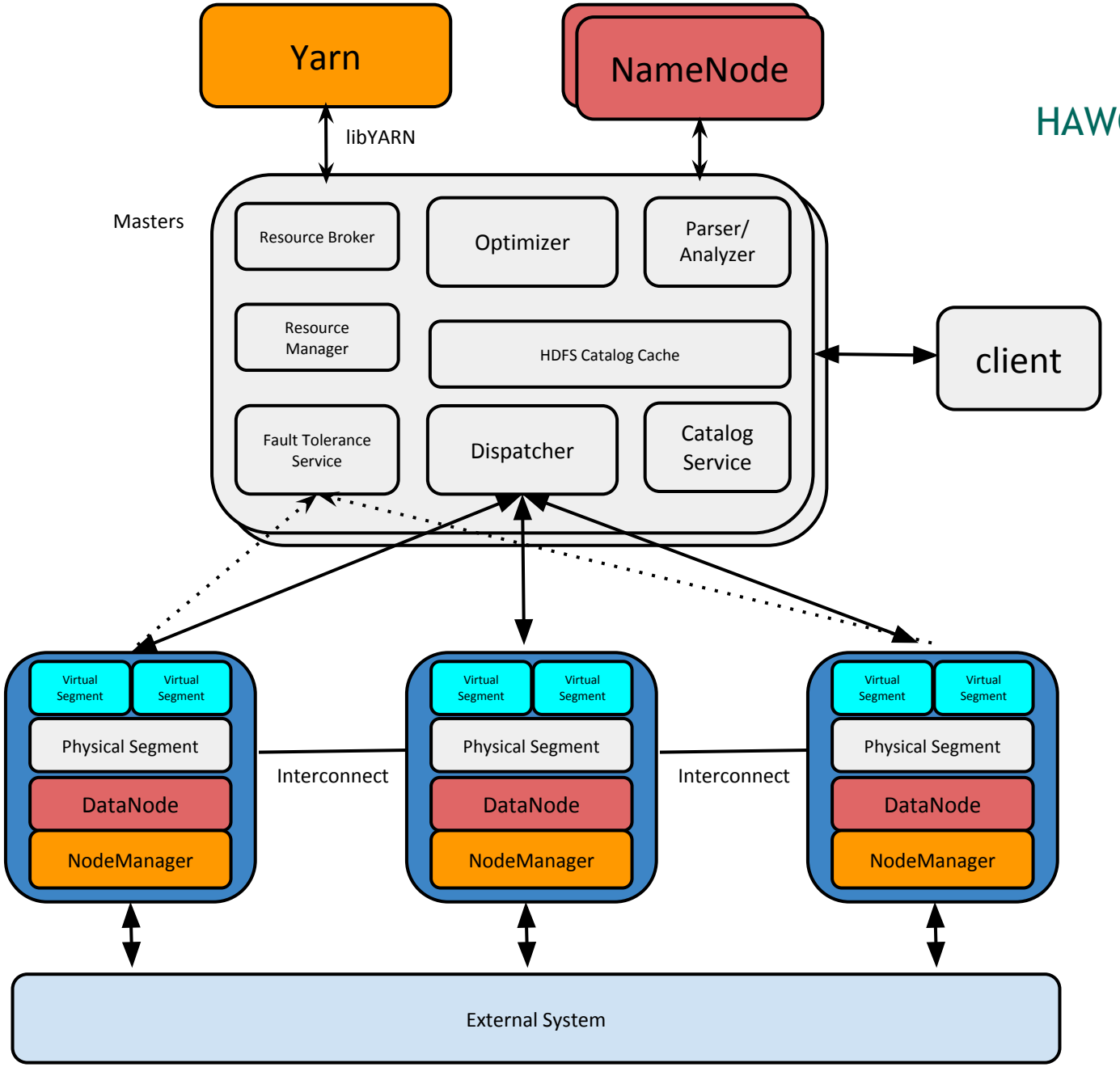
- 2011: Prototype
  - GoH (Greenplum Database on HDFS)
- 2012: HAWQ Alpha
- March 2013: HAWQ 1.0
  - Architecture changes for a Hadoopy system
- 2013~2014: HAWQ 1.x
  - HAWQ 1.1, HAWQ 1.2, HAWQ 1.3…
- 2015: HAWQ 2.0 Beta & Apache incubating
  - http://hawq.incubator.apache.org

HAWQ components

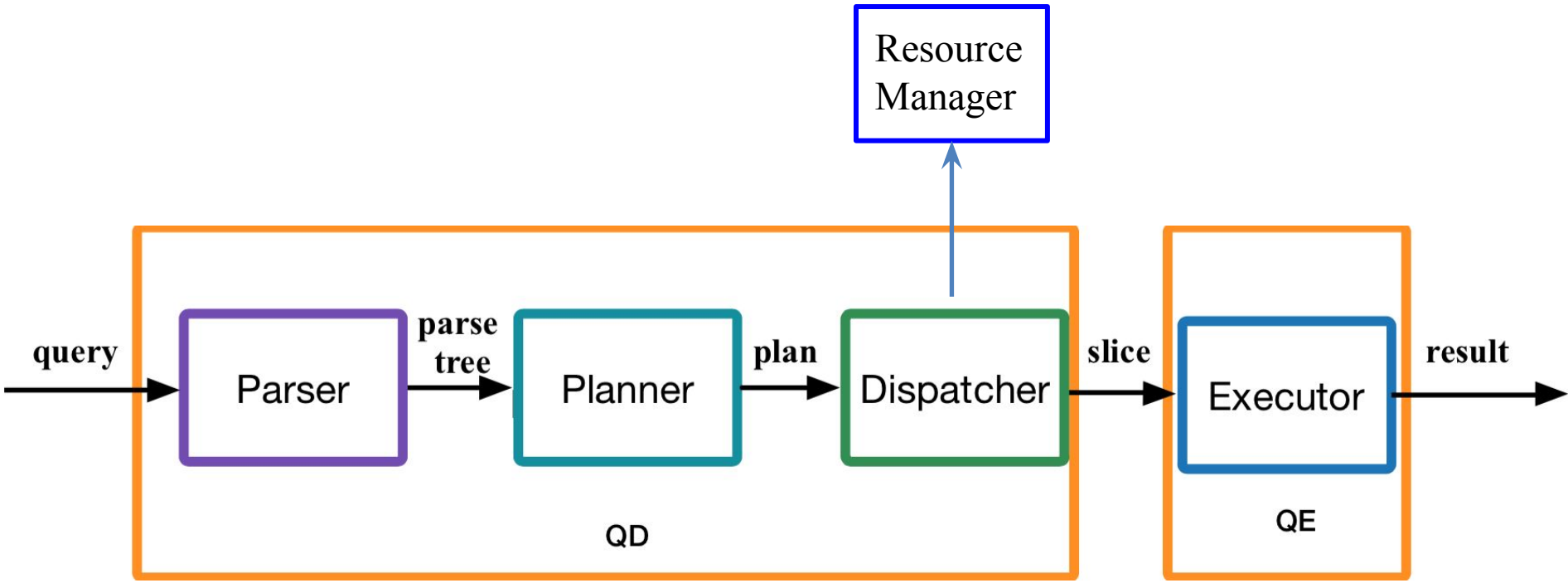


Pivotal

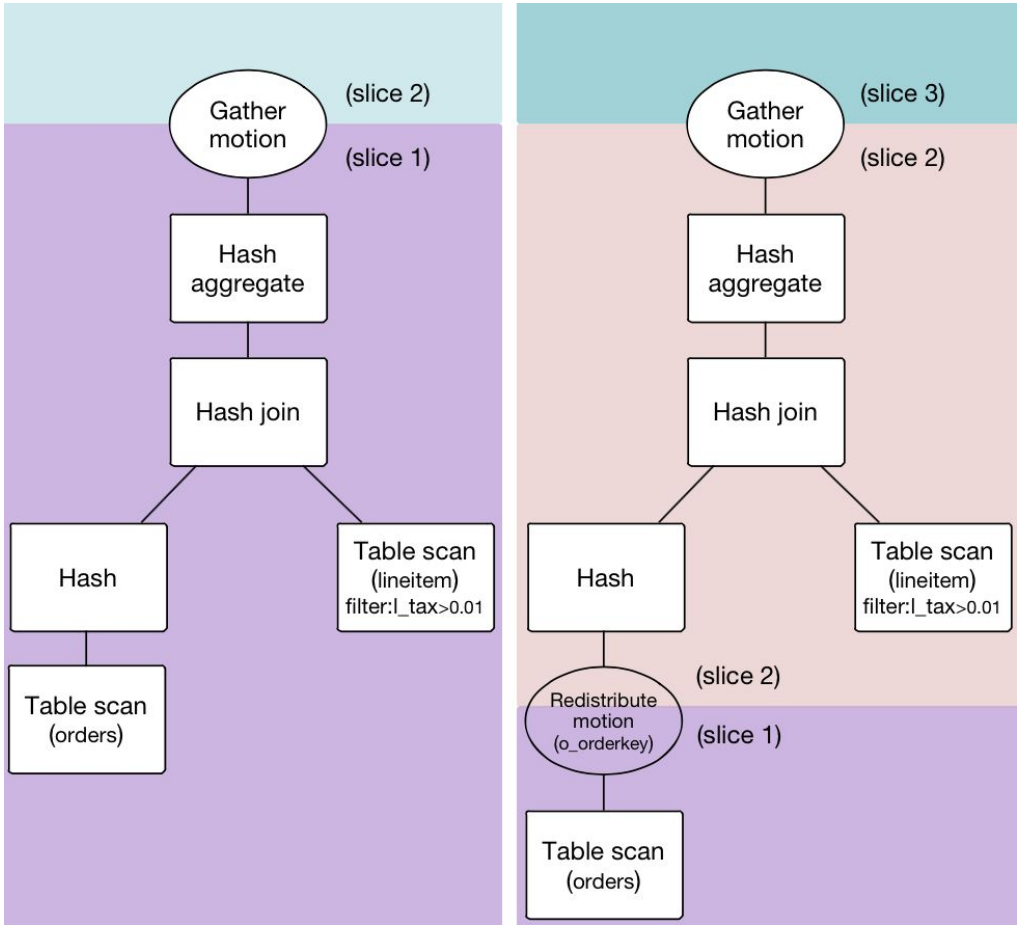
HAWQ Architecture



Basic query execution flow



Plan



Motion:

- Redistribution
- Broadcast
- Gather

```
SELECT l_orderkey, count(l_quantity)
FROM lineitem, orders
WHERE l_orderkey=o_orderkey AND l_tax>0.
GROUP BY l_orderkey;
```

Pivotal

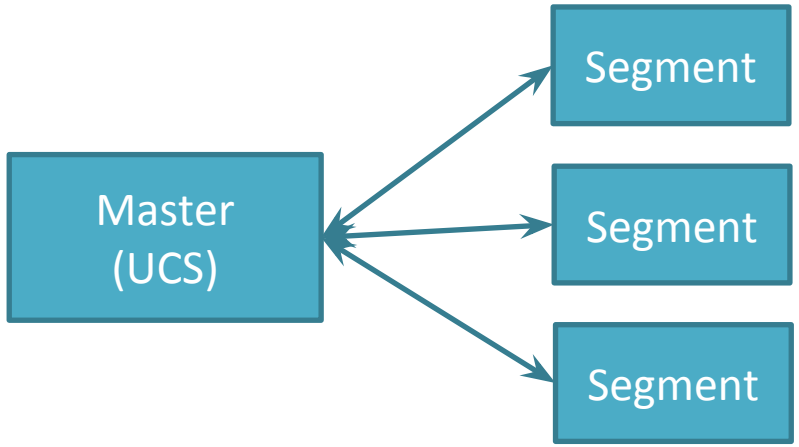
Catalog service

- Currently on master and external in future
- Query language: CaQL
- Basic single-table SELECT, COUNT(), multi-row DELETE, and single row INSERT/UPDATE

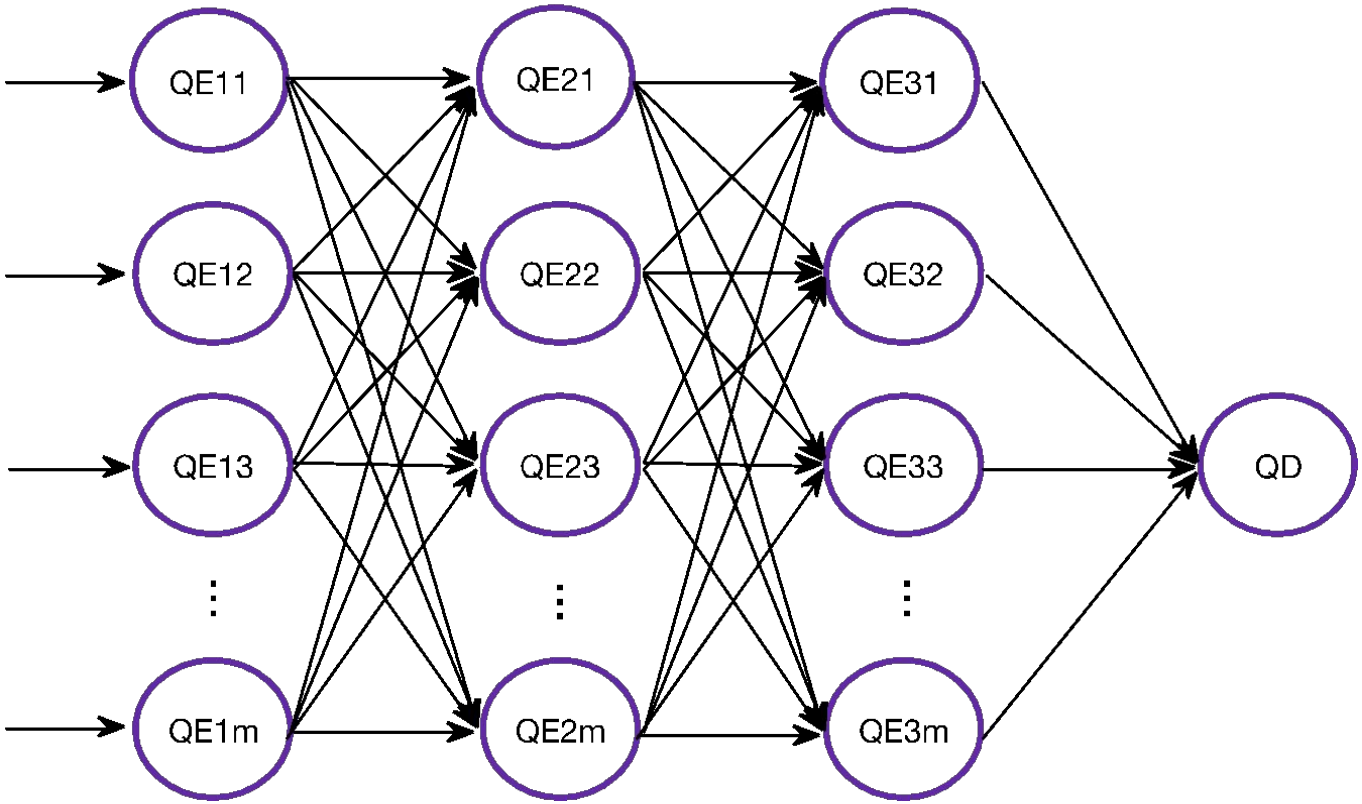
Pivotal

Virtual segments & elasticity

- Segment is stateless
- Metadata in Catalog Service
- Self-described plan
- Compression
- Virtual Segment
  - Only one physical segment on each node
    - Ease install and management
    - Ease performance tuning
  - Number of virtual segments
    - Resource available at the query running time
    - The cost of the query
- Node expansion & shrinking



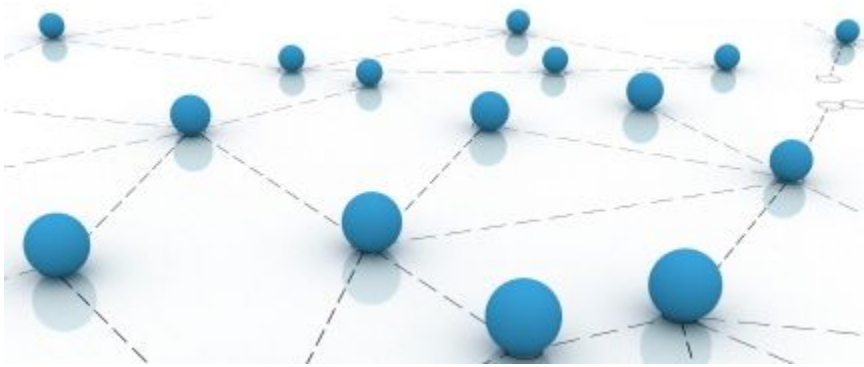
Interconnect



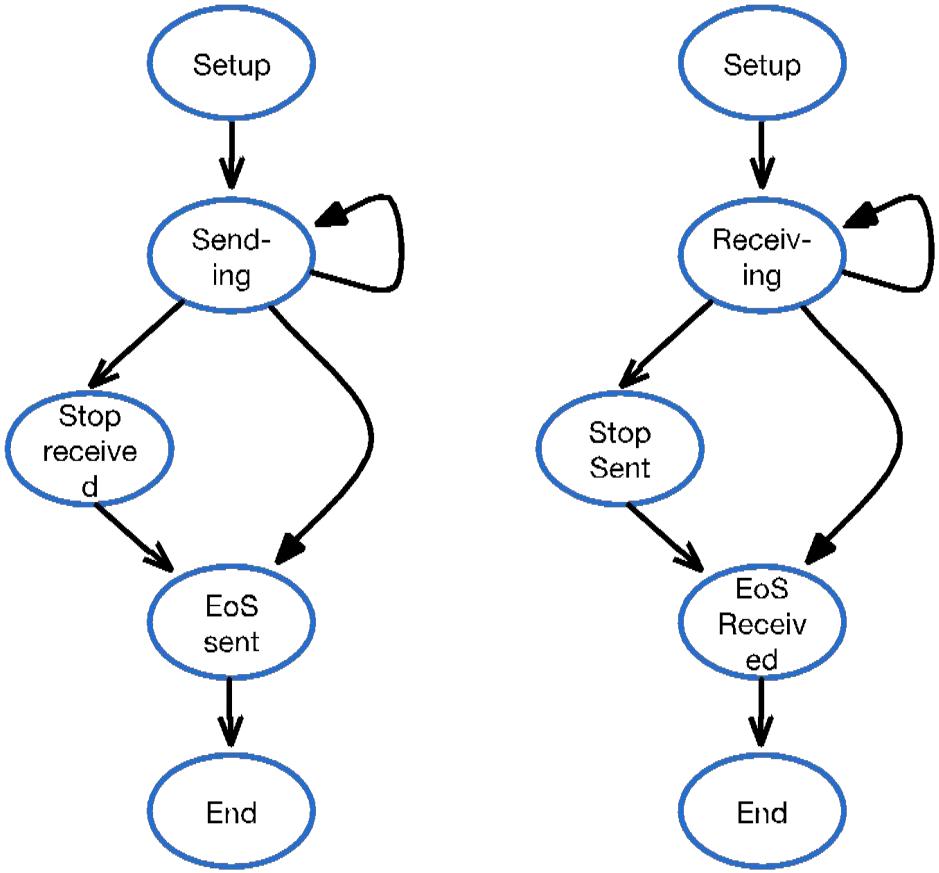
Pivotal

Interconnect

- Two transports: TCP & UDP
- TCP interconnect limitations
  - Limited scale
  - High Latency for connection setup
- Goals of UDP interconnect
  - Reliability
  - Ordering
  - Flow control
  - Performance and scalability
  - Portability



Interconnect state machine



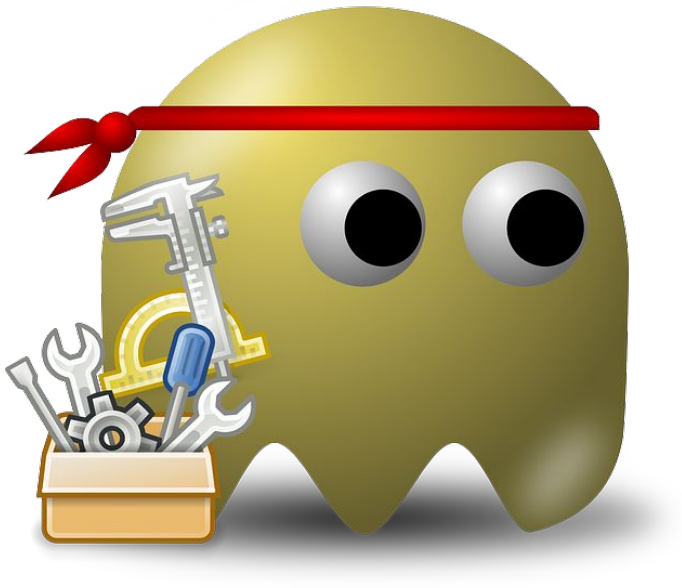
(a) Sender

(b) Receiver

Pivotal

The Design

- Multiple virtual connections share a common socket
- Multi-threaded
- Flow control: flow control window based
- Packet losses and out-of-order packets
- Deadlock elimination



Pivotal

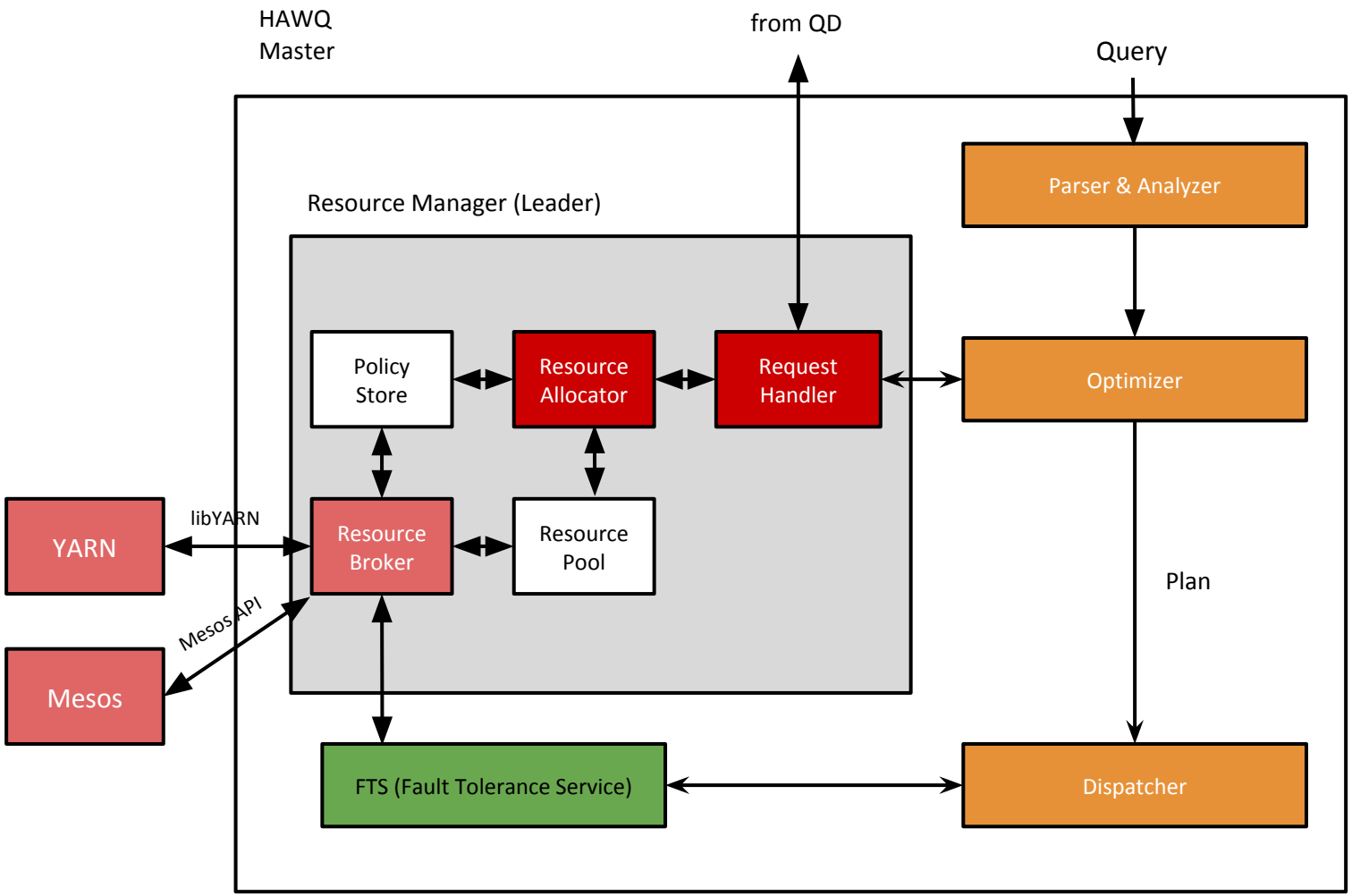
Transaction Management

- Snapshot isolation
- Catalog data: MVCC
- User data: Logical EoF & swimming lane
- Truncate to help remove garbage data

Resource manager

- Responsibility
  - Responsible for acquiring from YARN and return resources to YARN
  - Responsible for resource allocation among HAWQ users, queries and operators
- Two level resource management
  - First level: integration with external resource manager (such as YARN…)
  - Second level: HAWQ internal resource manager (user/query level resource management)
  - Third level: operator level
- Hierarchical resource queues
- CPU and memory management (allocation & enforcement)

Resource manager components



Storage

- Row oriented: AO
  - Quicklz, zlib
- PAX like: Parquet
  - Snappy, gzip
- Other format: via PXF

Pivotal

Contributing to HAWQ

- Documentation
- Wiki
- Bug reports
- Bug fixes
- Features

- Website: http://hawq.incubator.apache.org/
- Wiki: https://cwiki.apache.org/confluence/display/HAWQ
- Repo: https://github.com/apache/incubator-hawq.git
- JIRA: https://issues.apache.org/jira/browse/HAWQ
- Mailing lists: dev/user@hawq.incubator.apache.org

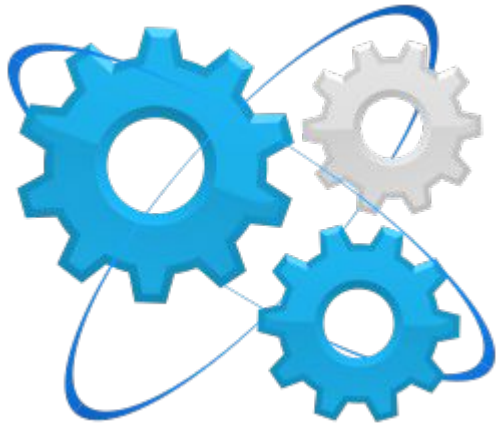
Code contribution process

- Start a JIRA
- Fork a github repo: https://github.com/apache/incubator-hawq.git
- Clone your repo to local
- Add the github repo as “upstream”
- Create a feature branch and commit your code
- Start a pull request for code review

Details:https://cwiki.apache.org/confluence/display/HAWQ/Contributing+to+HAWQ

Build & Setup

- Build
  - Option 1: Use pre-built docker image: https://hub.docker.com/r/mayjojo/hawq-dev/
  - Option 2: Build dependencies by yourself
    - https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=61320026
- Setup & Run
  - HDFS (required)
  - YARN (Optional)
  - HAWQ Init/start/stop cluster
  - psql -d postgres



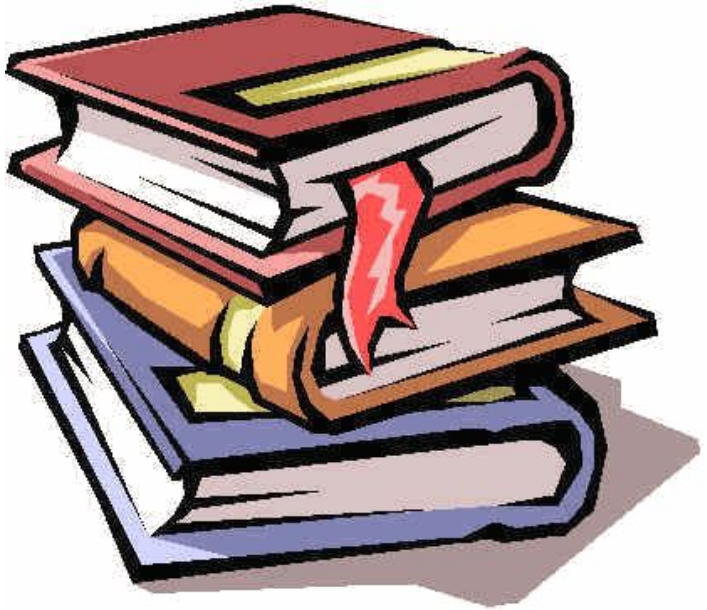
Pivotal

References

- HAWQ website:
  - http://hawq.incubator.apache.org
  - http://pivotal.io/big-data/pivotal-hawq
- HAWQ research papers
  - Lei Chang et al: HAWQ: a massively parallel processing SQL engine in hadoop. SIGMOD Conference 2014: 1223-1234
  - Mohamed A. Soliman et al: Orca: a modular query optimizer architecture for big data. SIGMOD Conference 2014: 337-348
  - Lyublena Antova et al: Optimizing queries over partitioned tables in MPP systems. SIGMOD Conference 2014: 373-384
  - Amr El-Helw et al: Optimization of Common Table Expressions in MPP Database Systems. PVLDB 8(12): 1704-1715 (2015)

Summary

- HAWQ: A Hadoop Native Parallel SQL Engine
- How to contribute



Pivotal