Overview

- Designed for parallel querying
- Shared-nothing architecture
- Appears as a single database to the application
- Utilizes PostgreSQL
- Data Loader for parallel loading
- Not just “Read-Only”, can execute UPDATE, DELETE, transactions
- Standard connectivity via PostgreSQL compatible connectors (supports PostgreSQL protocol): JDBC, ODBC, ADO.NET
GridSQL

Diagram:
- Network
  - GridSQL Coordinator
    - Node 1
      - NodeAgent
        - Node 2
        - Node Data
      - Node Data
  - Metadata
  - NodeAgent
    - Node 3
    - Node Data
  - NodeAgent
    - Node 4
    - Node Data
The Metadata Database

• Contains schema information including table partitioning and replication
• DDL issued to the GridSQL is recorded in the metadata database
• SQL requests made to the GridSQL interrogate the metadata database for partitioning and replication information to parallelize query plan

xsystables xsystabspaces xsysindexkeys
xsyscolumns xsysindexes xsysconstraints
xsysviews
Central Coordinator

- Multi-threaded process running on designated node that manages and coordinates work between the nodes
- Makes use of metadata information
- Performs traditional DBMS functions and manages interactions with the node agents
  - Parsing and optimizing
  - Scheduling and execution
Tables are designated as being either
- Partitioned by column
- Round robin
- Replicated
- Single node

CREATE TABLE region
    (r_regionkey  INTEGER NOT NULL,
     r_name        CHAR(25) NOT NULL,
     r_comment  VARCHAR(152))
REPLICATED;
CREATE TABLE orders (  
o_orderkey INTEGER NOT NULL,
o_custkey INTEGER NOT NULL,
o_orderstatus CHAR(1) NOT NULL,
o_totalprice DECIMAL(15,2) NOT NULL,
o_orderdate DATE NOT NULL,
o_orderpriority CHAR(15) NOT NULL,
o_clerk CHAR(15) NOT NULL,
o_shippriority INTEGER NOT NULL,
o_comment VARCHAR(79) NOT NULL)  
PARTITIONING KEY o_orderkey on all;
Data Distribution

- Inserted Data Distributed for Partitioned Tables
Inserting Data

- INSERT INTO region VALUES (1, ‘North America’, ‘comment’);
- gs-loader
  - Uses COPY API
  - -b: basic checking like number of delimiters performed
  - -k: number of rows per “chunk” to try to load, percent reduction, smallest chunk size
  - Example:
    - gs-loader.sh -d DEV -u admin -i /load/lineitem.tbl -t lineitem -b /load/bad/lineitem.bad -r # -k 100000,10,1 -y /load/bad
Query Example - Processing

- Query Parsed
- Query Optimized
- Query Planned, Including Transformations
- Query Executed In Steps
  - Intelligently executes in parallel
  - First set of aggregates done in parallel at the nodes
  - Like groups of intermediate results shipped to same target node
  - Second aggregation done in parallel
  - Coordinator streams in node results, combining on the fly and sending to client result set, performing a merge sort if ORDER BY present
Query Example #1

• SELECT COUNT(*) FROM ORDERS;
Query Example #1

Step: 0
-------
Target: CREATE TABLE TMPTT1_1 (XCOL1 INT) WITHOUT OIDS
Select: SELECT COUNT(*) AS XCOL1 FROM orders

Step: 1
-------
Target:
Select: SELECT SUM(XCOL1) AS EXPRESSION1 FROM TMPTT1_1
Drop:
TMPTT1_1
Query Example #1, step 1

ExecutionStep
-------------
producerCount = 2
consumerCount = 0
isExtraStep = false
isFinalStep = false

aStepDetail
-----------
StepNo = 1
isProducer = true
isConsumer = false
queryString = SELECT COUNT(*) AS XCOL1
FROM orders
targetTable = TMPTT1_1
targetSchema =
DropList =
destType = DEST_TYPE_COORD
combineOnCoordFirst = false
consumerNodeList =

coordStepDetail
---------------
StepNo = 1
isProducer = false
isConsumer = true
queryString = null
targetTable = TMPTT1_1
targetSchema = CREATE TABLE TMPTT1_1
(XCOL1 INT) WITHOUT OIDS
DropList =
destType = (none set) -1
combineOnCoordFirst = false
consumerNodeList =

nodeUsageTable
--------------
nodeld = 2  isProducer = true  isConsumer = false
nodeld = 1  isProducer = true  isConsumer = false
Query Example #1, step 2

ExecutionStep
-------------
producerCount = 0
consumerCount = 0
isExtraStep = false
isFinalStep = true

DropList = TMPTT1_1

coordStepDetail
-------------
StepNo = 2
isProducer = true
isConsumer = false
queryString = SELECT SUM(XCOL1) as EXPRESSION1 FROM TMPTT1_1

targetTable =
targetSchema =
DropList =
destType = DEST_TYPE_COORD_FINAL
consumerNodeList =
Query Example #2

SELECT n_name, SUM(l_extendedprice)
FROM customer INNER JOIN orders on c_custkey = o_custkey
    INNER JOIN lineitem ON o_orderkey = l_orderkey
    INNER JOIN nation ON c_nationkey = n_nationkey
    INNER JOIN region ON n_regionkey = r_regionkey
WHERE r_name = 'ASIA'
    AND c_mktsegment = 'BUILDING'
GROUP BY n_name

Replicated: nation, region
Partitioning columns: customer.c_custkey, orders.o_orderkey, lineitem.l_orderkey
Step: 0
Target: CREATE TABLE TMPTT3_1 ( n_name CHAR (25), c_custkey INT) WITHOUT OIDS
Select: SELECT nation.n_name AS n_name,customer.c_custkey AS c_custkey FROM nation INNER JOIN region ON (nation.n_regionkey = region.r_regionkey) INNER JOIN customer ON (customer.c_nationkey = nation.n_nationkey) WHERE (customer.c_mktsegment = 'BUILDING') AND (region.r_name = 'ASIA')

Step: 1
Target: CREATE TABLE TMPTT3_2 ( XCOL1 CHAR (25), XCOL2 FLOAT (32)) WITHOUT OIDS
Select: SELECT TMPTT3_1.n_name AS XCOL1,sum( lineitem.l_extendedprice) AS XCOL2 FROM TMPTT3_1 INNER JOIN orders ON (TMPTT3_1.c_custkey = orders.o_custkey) INNER JOIN lineitem ON (orders.o_orderkey = lineitem.l_orderkey) group by TMPTT3_1.n_name
Drop: TMPTT3_1

Step: 2
Target: CREATE TABLE TMPTT3_3 ( n_name CHAR (25), EXPRESSION1 FLOAT (32)) WITHOUT OIDS
Select: SELECT XCOL1 AS n_name,SUM(XCOL2) AS EXPRESSION1 FROM TMPTT3_2 group by XCOL1
Drop: TMPTT3_2
Query Example #2, step 2

ExecutionStep
--------------
producerCount = 2
consumerCount = 2
isExtraStep = false
isFinalStep = false
destNodeList = 1 2

nodeUsageTable
--------------
nodeId = 2
isProducer = true
isConsumer = true

nodeId = 1
isProducer = true
isConsumer = true

aStepDetail
-----------
StepNo = 2
isProducer = true
isConsumer = true

queryString = SELECT TMPTT3_1.n_name AS XCOL1,sum(lineitem.l_extendedprice) AS XCOL2 FROM TMPTT3_1 INNER JOIN orders ON (TMPTT3_1.c_custkey = orders.o_custkey) INNER JOIN lineitem ON (orders.o_orderkey = lineitem.l_orderkey) group by TMPTT3_1.n_name

targetTable = TMPTT3_2
targetSchema = CREATE TABLE TMPTT3_2 (XCOL1 CHAR(25), XCOL2 FLOAT (32)) WITHOUT OIDS

DropList = TMPTT3_1
destType = DEST_TYPE_HASH
hashColumn = null
hashColumnPosition = 1
combineOnCoordFirst = false
consumerNodeList = 1 2
Thank you!