Greenplum Architecture

Alexey Grishchenko

HUG Meetup
28.11.2015

Pivotal™

Agenda

- Introduction
- GPDB Architecture Overview
- Distribution and Partitioning
- Loading External Data
- Maintenance Procedures
- Performance Considerations
- Competitive

Agenda

- **Introduction**
- GPDB Architecture Overview
- Distribution and Partitioning
- Loading External Data
- Maintenance Procedures
- Performance Considerations
- Competitive

The Pivotal Greenplum Database is…

A Highly-Scalable, Shared-Nothing Database

- Leading MPP architecture, including a patented next-generation optimizer
- Optimized architecture and features for loading and queries
- Start small, scale as needed
- Polymorphic storage, compression, partitioning

A Platform for Advanced Analytics on Any (and All) Data

- Rich ecosystem (SAS, R, BI & ETL tools)
- In-DB Analytics (MADlib, Custom, languages: R, Java, Python, PERL, C, C++)
- High degree of SQL completeness so analysts can use a language they know
- Domain: Geospatial, Text processing (GPText)

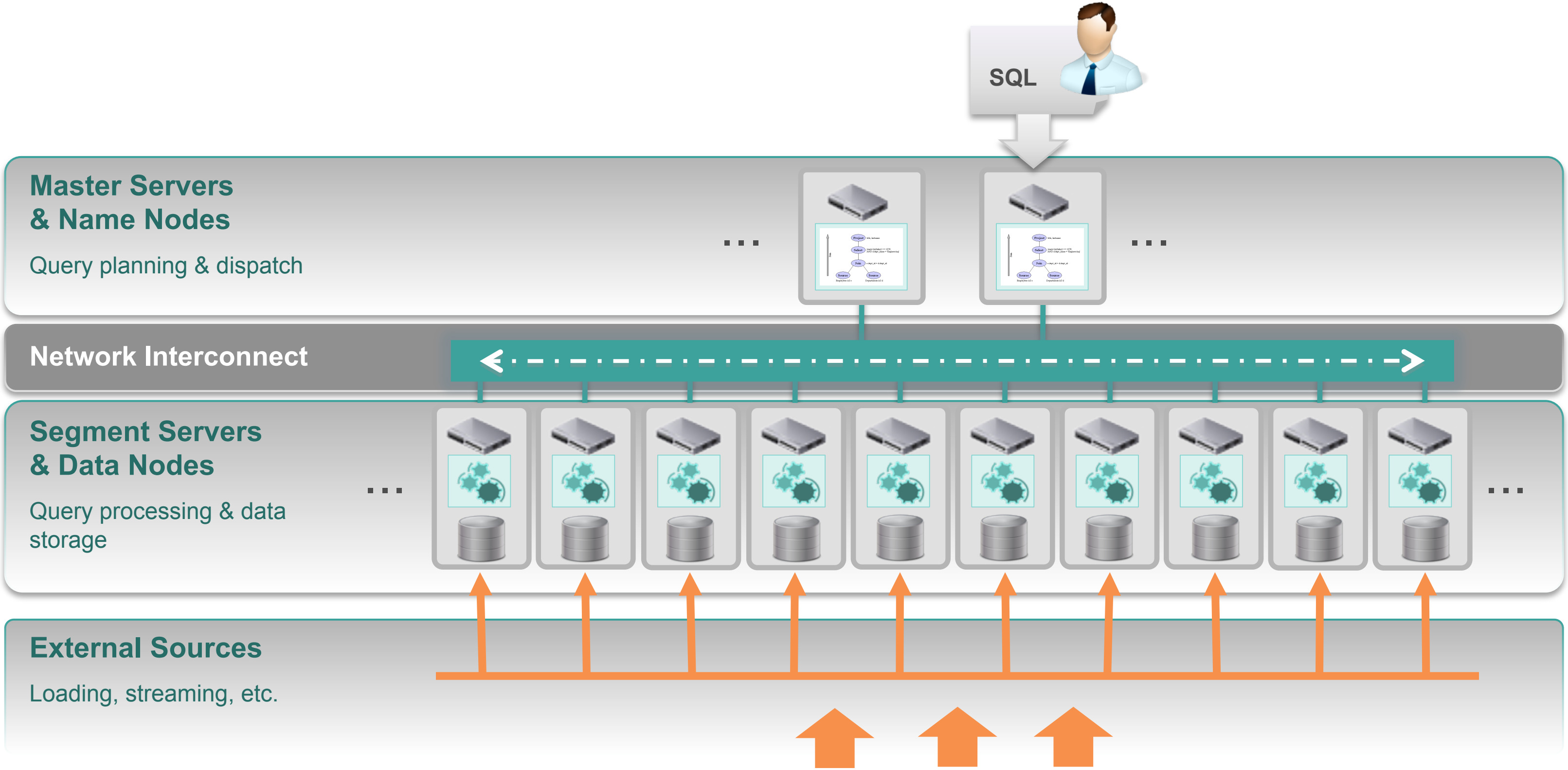
An Enterprise Ready Platform Capable of Flexing With Your Needs

- Available as needed – either as an appliance or software
- Secures data in-place, in flight, and with authentication to suit
- Capable of managing a variety of mixed workloads

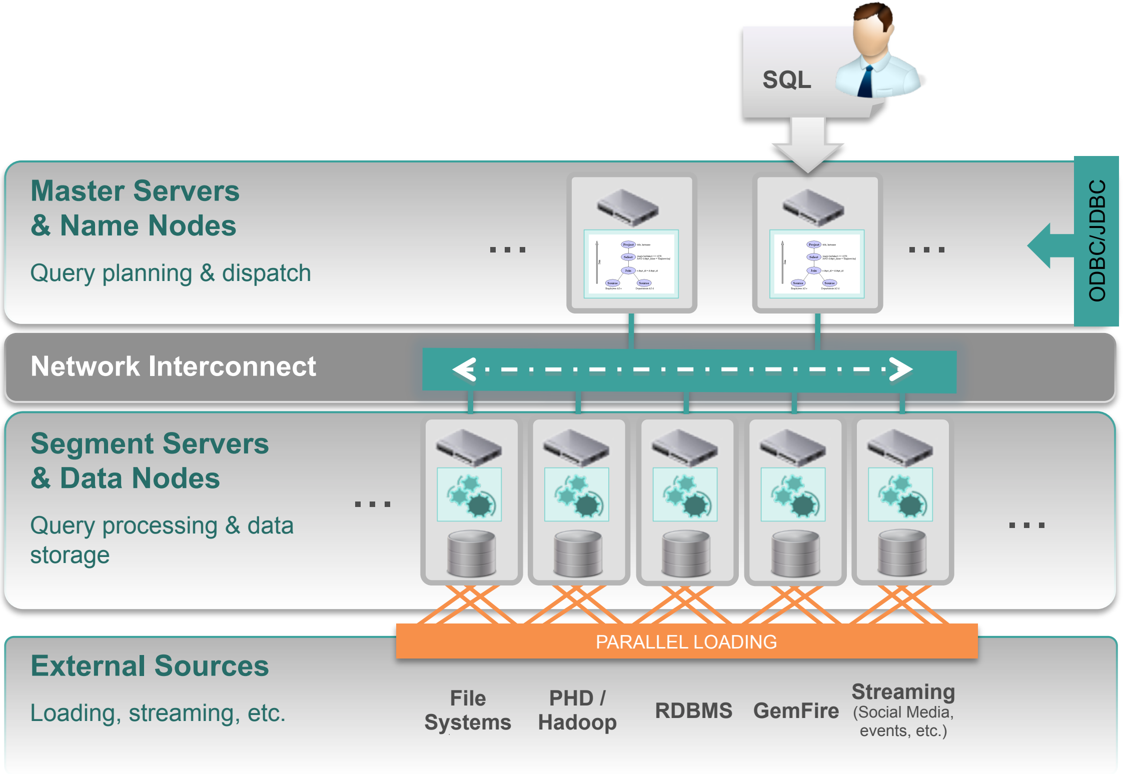
The Pivotal Greenplum Database Overview

- A highly scalable shared-nothing database
- A platform for advanced analytics on any (and all) data
- An enterprise ready platform capable of flexing with your needs

MPP 101: Performance Through Parallelism

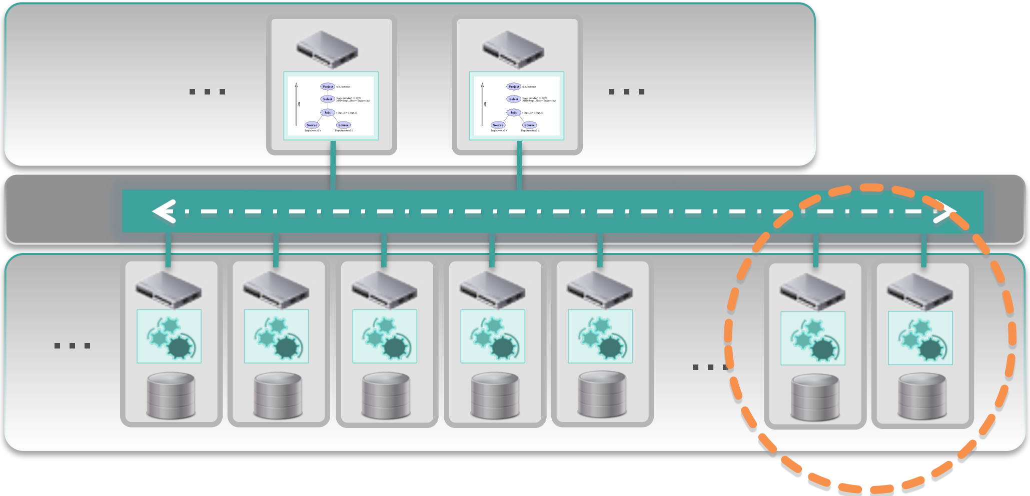


MPP 102: True High Speed Loading


SQL
Master Servers & Name Nodes
Query planning & dispatch
ODBC/JDBC
Network Interconnect
Segment Servers & Data Nodes
Query processing & data storage
PARALLEL LOADING
External Sources
Loading, streaming, etc.
File Systems
PHD / Hadoop
RDBMS
GemFire
Streaming (Social Media, events, etc.)

- Parallelizes Everything
  - All nodes can process loading requests
  - No subsequent “Data Reorganization” steps.
  - Scales at over 10+TB/hr. per rack.
  - Only constrained by the speed of the source
- Automates Parallelism
  - GPLoad utility automatically parallelizes file-based loading
  - Integrated with ETL products to parallelize ETL-based loading with minimal added effort

MPP 201: Start Small and Scale as Needed



New Segment Servers

Query planning & dispatch

- Advantages:
  - Scale In-Place
  - No Forklifting
  - Immediately Usable
  - Simple Process

Advanced MPP: Polymorphic Storage™

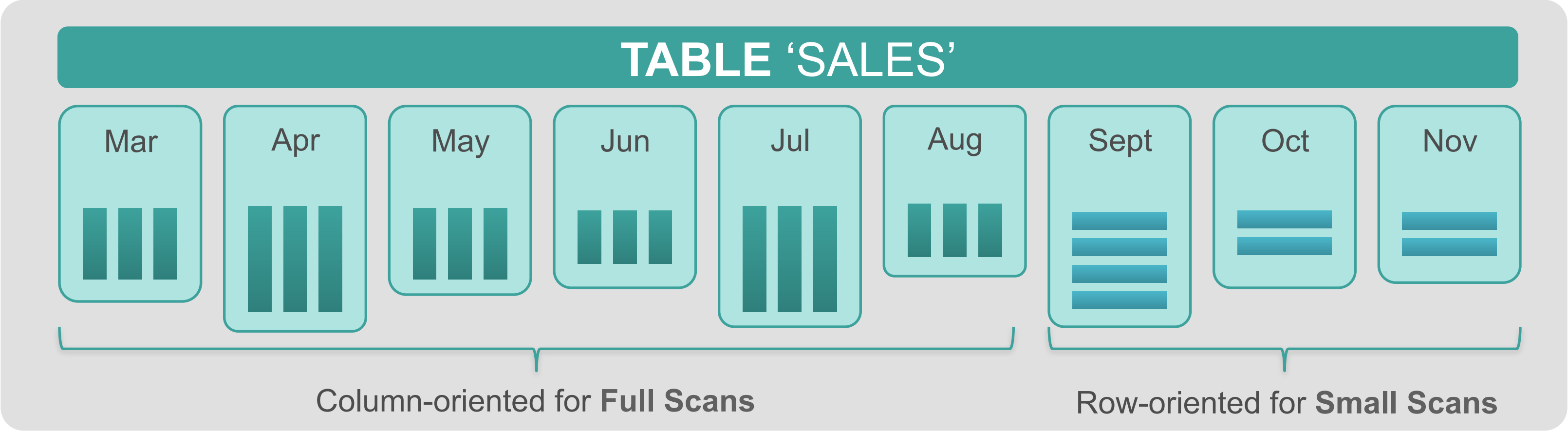


TABLE ‘SALES’
Mar
Apr
May
Jun
Jul
Aug
Sept
Oct
Nov
Column-oriented for **Full Scans**
Row-oriented for **Small Scans**

- Columnar storage is well suited to scanning a large percentage of the data
- Row storage excels at small lookups
- Most systems need to do both
- Row and column orientation can be mixed within a table or database

- Both types can be dramatically more efficient with compression
- Compression is definable column by column:
  - Blockwise: Gzip1-9 & QuickLZ
  - Streamwise: Run Length Encoding (RLE) (levels 1-4)
- Flexible indexing, partitioning enable more granular control and enable true ILM

Advanced MPP: Run Length Encoding

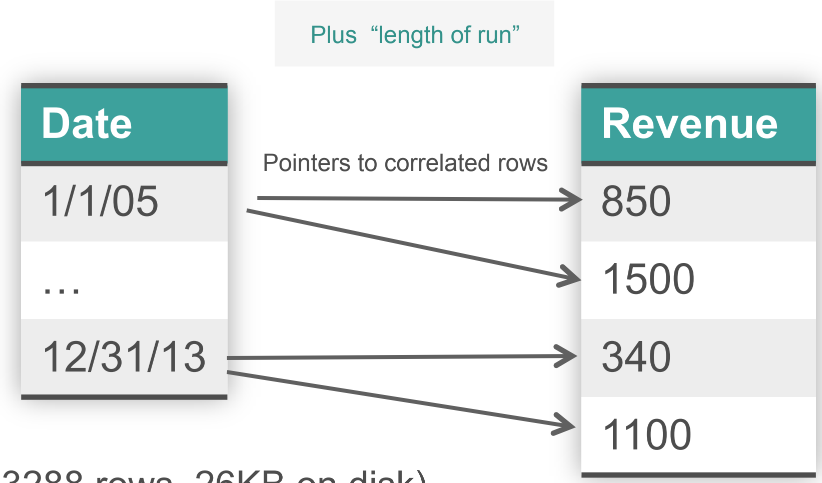
Unlocking the Potential of Column-Oriented Data

With columnar storage and RLE, this data…

Date	User_ID	Revenue
1/1/05	13111123	850
1/1/05	32343122	1500
12/31/13	45322323	340
12/31/13	39923001	1100

(10 billion rows, ~225GB on disk)

…can be stored like this…



Plus “length of run”

Date
1/1/05
…
12/31/13

Pointers to correlated rows

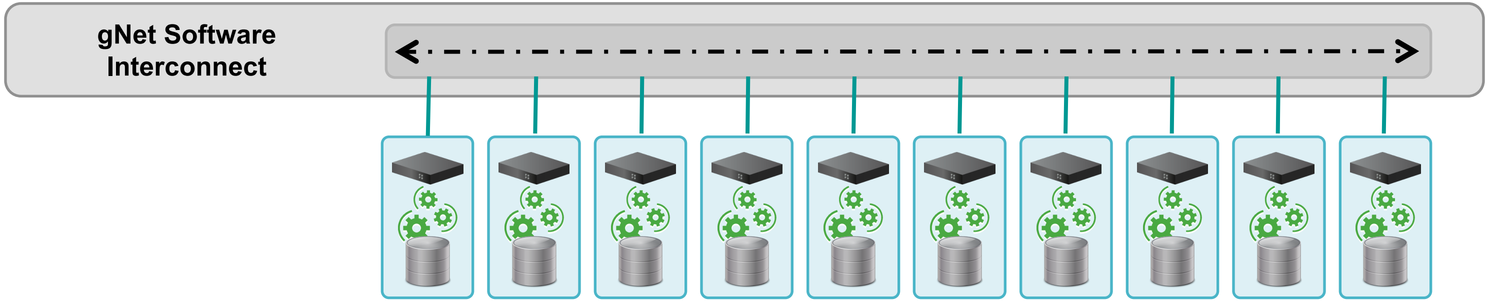
Revenue
850
1500
340
1100

(3288 rows, 26KB on disk)

(10 billion values – also compressed, ~37GB on disk)

gNet Software Interconnect

- A supercomputing-based "soft-switch" responsible for
  - Efficiently pumping streams of data between motion nodes during query-plan execution
  - Delivers messages, moves data, collects results, and coordinates work among the segments in the system

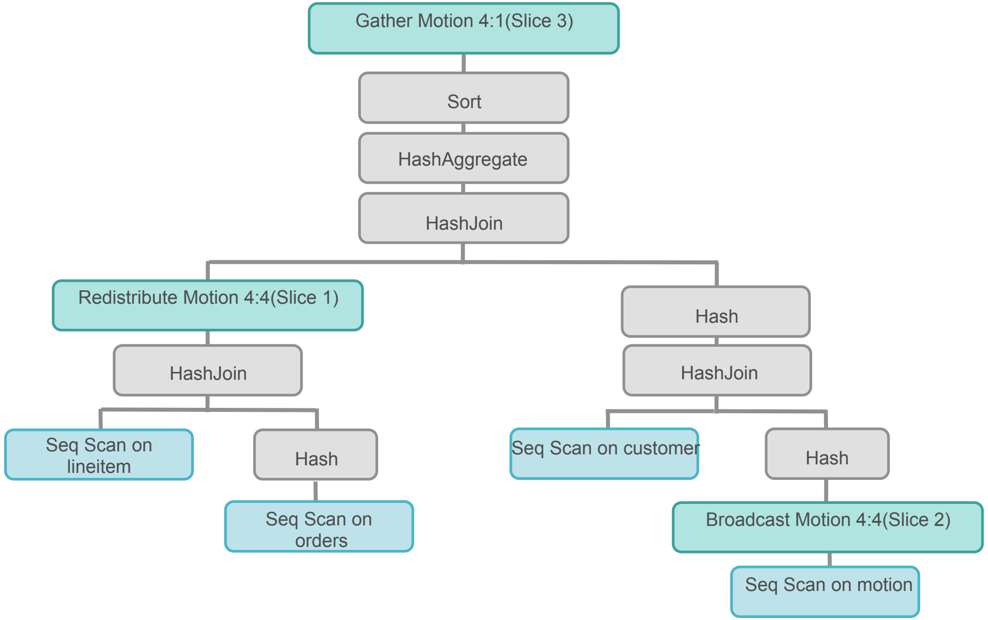


Pivotal™

Parallel Query Optimizer

- Cost-based optimization looks for the most efficient plan
- Physical plan contains scans, joins, sorts, aggregations, etc.
- Global planning avoids sub-optimal ‘SQL pushing’ to segments
- Directly inserts ‘motion’ nodes for inter-segment communication

PHYSICAL EXECUTION PLAN FROM SQL OR MAPREDUCE



Gather Motion 4:1(Slice 3)

Sort

HashAggregate

HashJoin

Redistribute Motion 4:4(Slice 1)

HashJoin

Seq Scan on lineitem

Hash

Seq Scan on orders

Hash

HashJoin

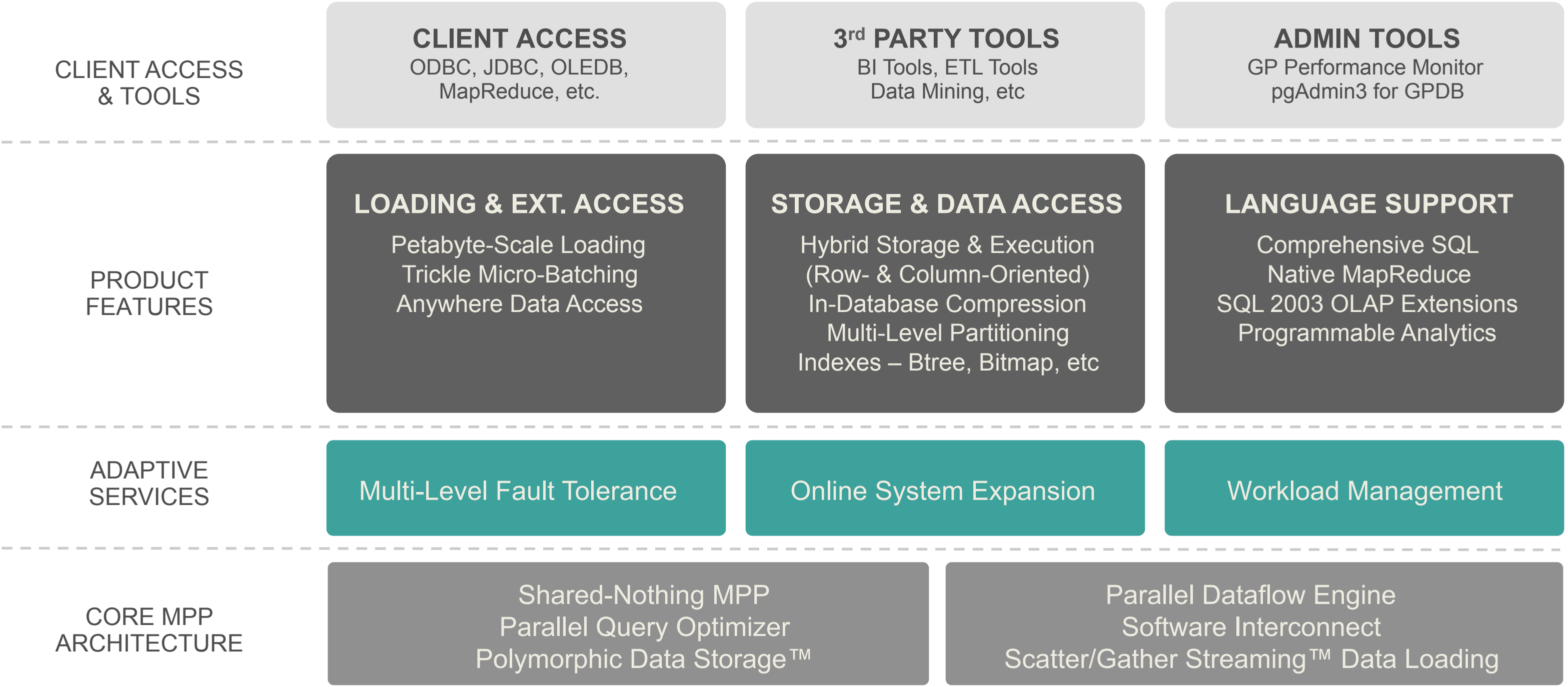
Seq Scan on customer

Hash

Broadcast Motion 4:4(Slice 2)

Seq Scan on motion

The Pivotal Greenplum Database at a Glance



CLIENT ACCESS & TOOLS

CLIENT ACCESS
ODBC, JDBC, OLEDB,
MapReduce, etc.

3rd PARTY TOOLS
BI Tools, ETL Tools
Data Mining, etc

ADMIN TOOLS
GP Performance Monitor
pgAdmin3 for GPDB

PRODUCT FEATURES

LOADING & EXT. ACCESS
Petabyte-Scale Loading
Trickle Micro-Batching
Anywhere Data Access

STORAGE & DATA ACCESS
Hybrid Storage & Execution
(Row- & Column-Oriented)
In-Database Compression
Multi-Level Partitioning
Indexes – Btree, Bitmap, etc

LANGUAGE SUPPORT
Comprehensive SQL
Native MapReduce
SQL 2003 OLAP Extensions
Programmable Analytics

ADAPTIVE SERVICES

Multi-Level Fault Tolerance

Online System Expansion

Workload Management

CORE MPP ARCHITECTURE

Shared-Nothing MPP
Parallel Query Optimizer
Polymorphic Data Storage™

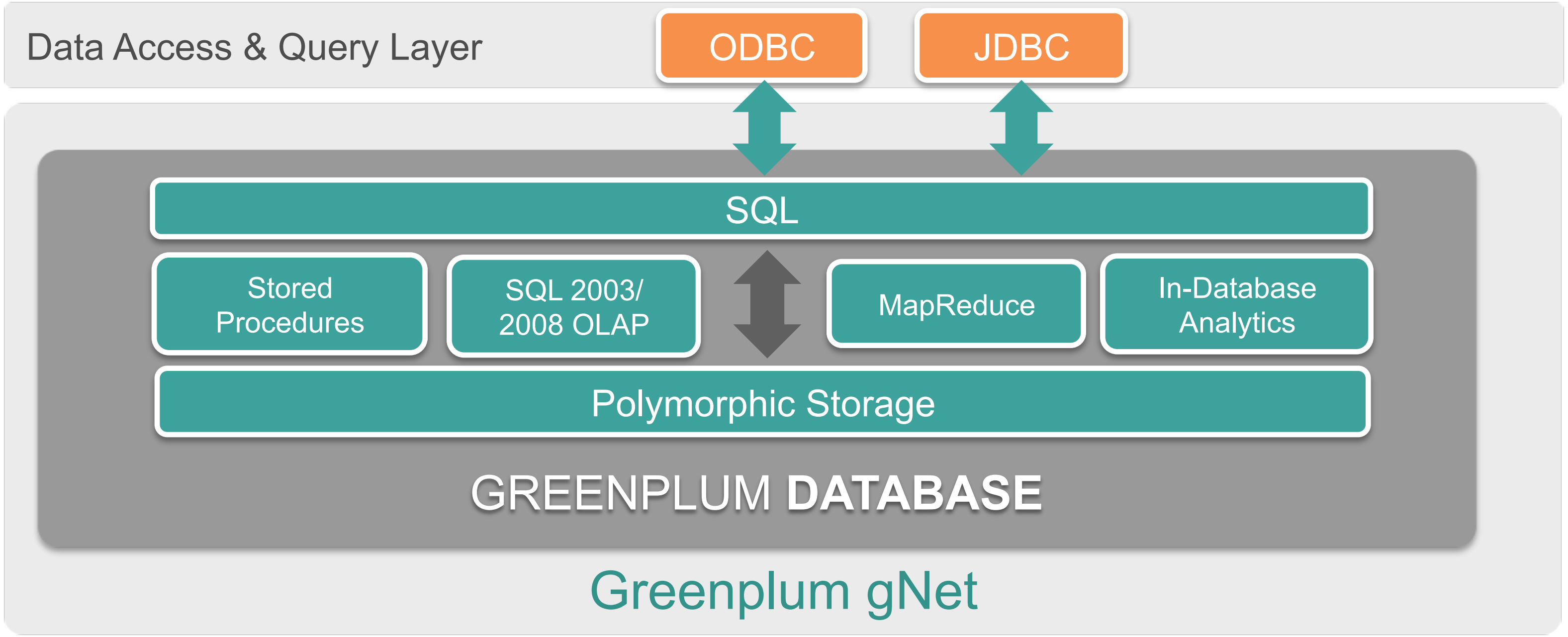
Parallel Dataflow Engine
Software Interconnect
Scatter/Gather Streaming™ Data Loading

The Pivotal Greenplum Database Overview

- A highly scalable shared-nothing database
- A platform for advanced analytics on any (and all) data
- An enterprise ready platform capable of flexing with your needs

Pivotal™

Analytical Architecture Overview



Data Access & Query Layer

ODBC

JDBC

SQL

Stored Procedures

SQL 2003/ 2008 OLAP

MapReduce

In-Database Analytics

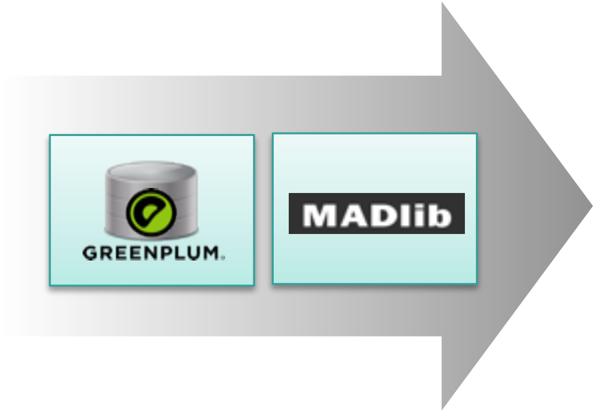
Polymorphic Storage

GREENPLUM **DATABASE**

Greenplum gNet

Easy to Use (at Scale) with SQL

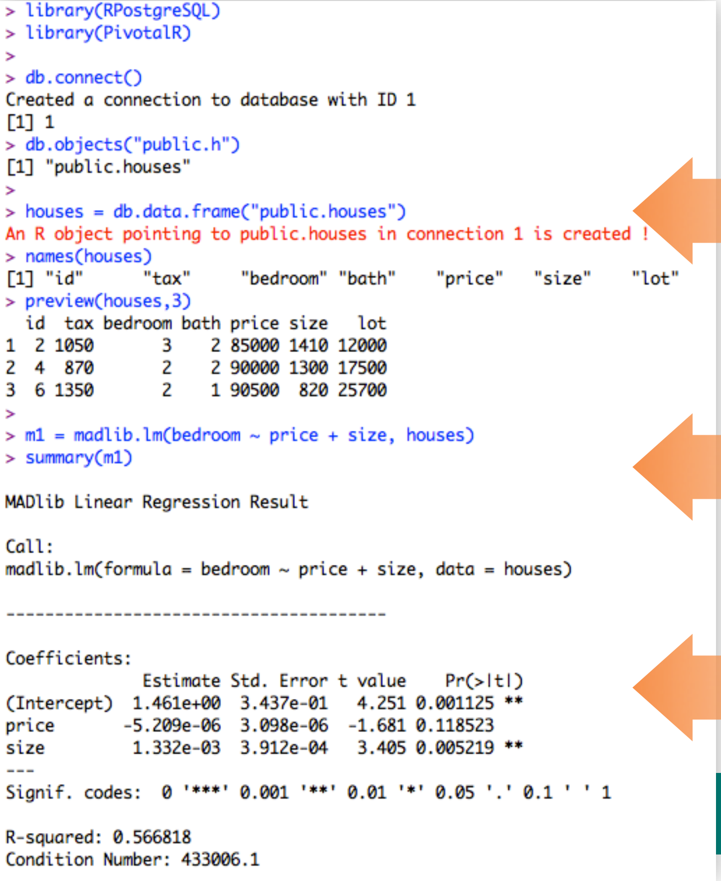
```
 SELECT
 (madlib.linregr(earns/hours, array[rooms, notcitizen,
married, ... ])).*
FROM (SELECT * FROM use_r.census) AS foo;
```



Rows	Elapsed Time
3 million	.5 s
10 million	1 s
100 million	5 s
500 million	23 s
1 billion	46 s
10 billion	453 s

Integrated with Tools/Languages, incl. R

- Load PivotalR Library
- List the columns in the table and preview the first 3 rows of data (the limit is passed through to the db)
- Examine the resulting model



```
> library(RPostgreSQL)
> library(PivotalR)
>
> db.connect()
Created a connection to database with ID 1
[1] 1
> db.objects("public.h")
[1] "public.houses"
>
> houses = db.data.frame("public.houses")
An R object pointing to public.houses in connection 1 is created !
> names(houses)
[1] "id"      "tax"     "bedroom" "bath"    "price"   "size"    "lot"
> preview(houses,3)
  id  tax bedroom bath price size   lot
1  2 1050       3    2 85000 1410 12000
2  4  870       2    2 90000 1300 17500
3  6 1350       2    1 90500  820 25700
>
> m1 = madlib.lm(bedroom ~ price + size, houses)
> summary(m1)

MADlib Linear Regression Result

Call:
madlib.lm(formula = bedroom ~ price + size, data = houses)

---------------------------------------

Coefficients:
              Estimate Std. Error t value  Pr(>|t|)
(Intercept)  1.461e+00  3.437e-01   4.251 0.001125 **
price       -5.209e-06  3.098e-06  -1.681 0.118523
size         1.332e-03  3.912e-04   3.405 0.005219 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-squared: 0.566818
Condition Number: 433006.1
```

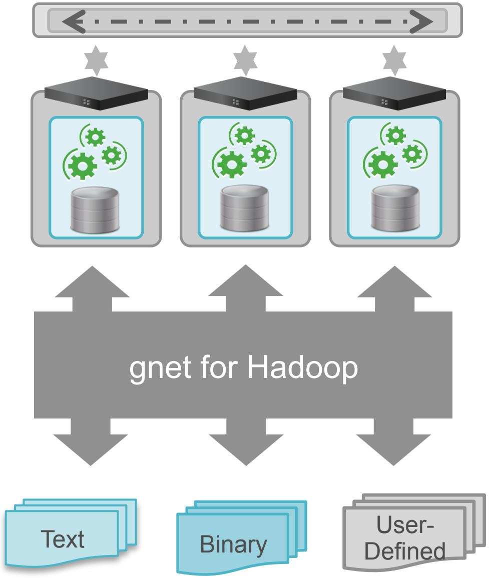
- Create the “houses” object as a proxy object in R. The data is not loaded into R
- Run a linear regression. This is executed in-database.
- The model is stored in-database, greatly simplifying the development of scoring applications

The Pivotal Greenplum Database Overview

- A highly scalable shared-nothing database
- A platform for advanced analytics on any (and all) data
- An enterprise ready platform capable of flexing with your needs

High Performance Integration with Hadoop

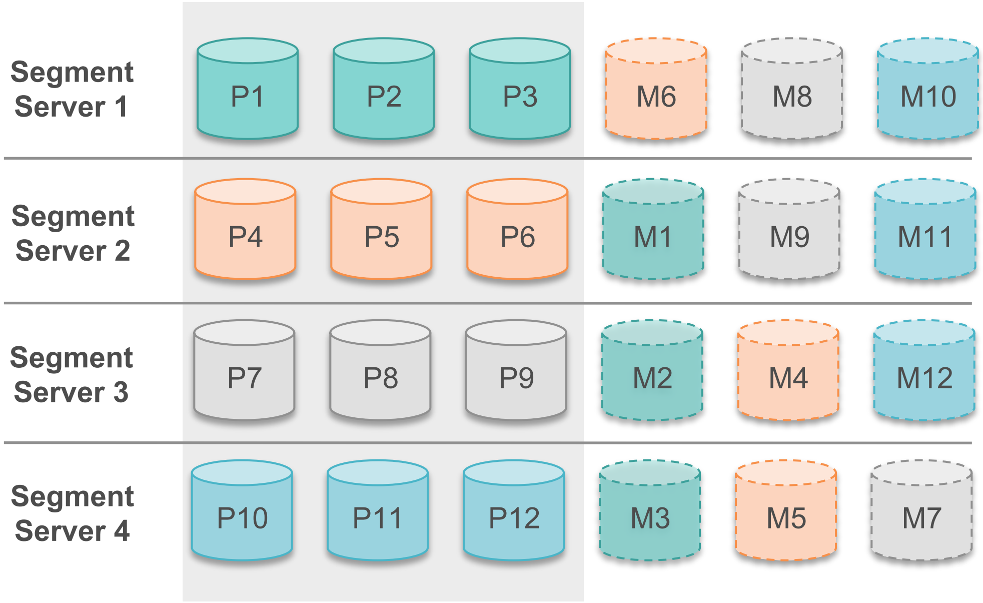
Parallel Query Access



- Connect any data set in Hadoop to GPDB SQL Engine
- Process Hadoop data in place
- Parallelize movement of data to/from Hadoop thanks to GPDB market leading data sharing performance
- Supported formats:
  - Text (compressed and uncompressed)
  - binary
  - proprietary/user-defined
- Support for Pivotal HD, MapR, Hortonworks, Cloudera

Comprehensive High Availability

- Master and Segment Mirroring with block level replication
  - Low resource consumption
  - Differential resynch capable for fast recovery
  - Minimize interdependencies!
- Segment servers support multiple database instances
  - Primary instances that actively process queries
  - Standby mirror instances



Segment Server 1	P1	P2	P3	M6	M8	M10
Segment Server 2	P4	P5	P6	M1	M9	M11
Segment Server 3	P7	P8	P9	M2	M4	M12
Segment Server 4	P10	P11	P12	M3	M5	M7

Set of Active Segment Instances

Comprehensive Backup/Restore

- Full and Incremental backup support with in-database tools
- Incremental backup
  - Only changed partitions are pulled for the backup
  - Restore to any point-in-time through support of “synthetic restores”
  - Synthetic restores automatically assemble the right backup based on the point-in-time specified: manual backup specification is not required
- Deep support for Data Domain
  - WAN replication of backup sets to remote DR sites
  - Granular delta-only backup support

Summary: The Pivotal Greenplum Database Delivers…

A Highly-Scalable, Shared-Nothing Database

- Leading MPP architecture, including a patented next-generation optimizer
- Optimized architecture and features for loading and queries
- Start small, scale as needed
- Polymorphic storage, compression, partitioning

A Platform for Advanced Analytics on Any (and All) Data

- Rich ecosystem (SAS, R, Chorus Studio, BI & ETL tools)
- In-DB Analytics (MADlib, Custom, languages: R, Java, Python, PERL, C, C++)
- High degree of SQL completeness so analysts can use a language they know
- Domain: Geospatial, Text processing (GPText)

An Enterprise Ready Platform Capable of Flexing With Your Needs

- Available as needed – either as an appliance or software
- Secures data in-place, in flight, and with authentication to suit
- Capable of managing a variety of mixed workloads

Pivotal™

Agenda

- *Introduction*
- **GPDB Architecture Overview**
- Distribution and Partitioning
- Loading External Data
- Maintenance Procedures
- Performance Considerations
- Competitive

Pivotal™

MPP Shared Nothing Architecture

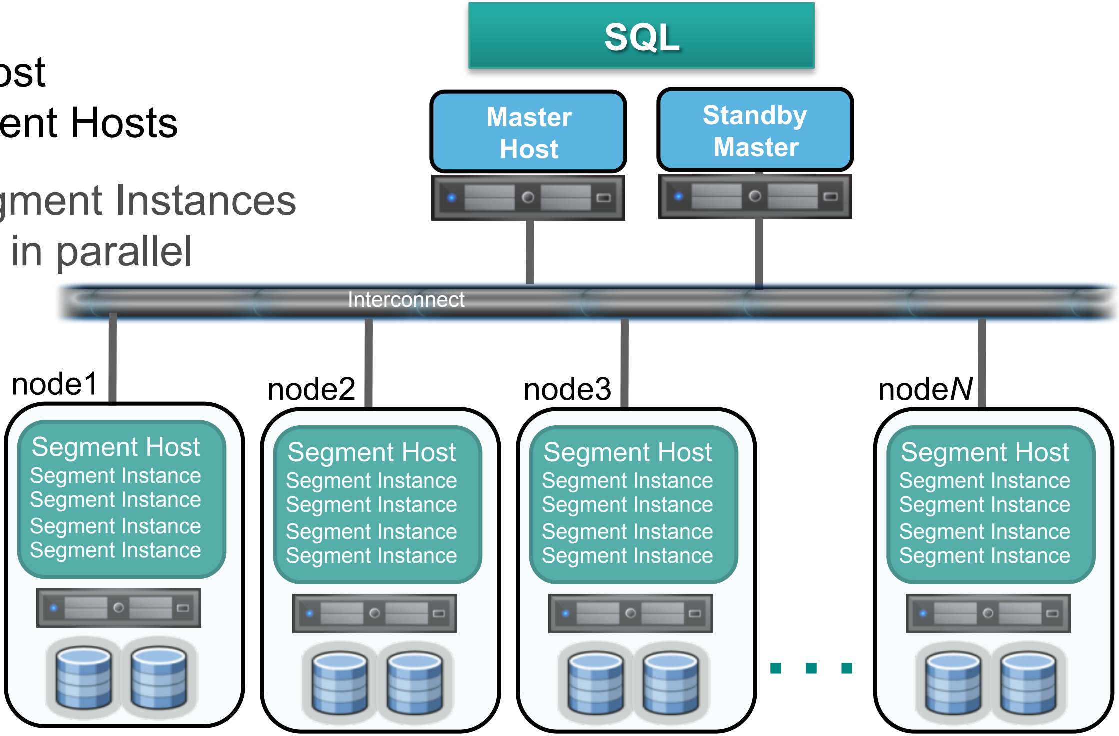
Flexible framework for processing large datasets

Master Host and Standby Master Host
Master coordinates work with Segment Hosts

Segment Host with one or more Segment Instances
Segment Instances process queries in parallel

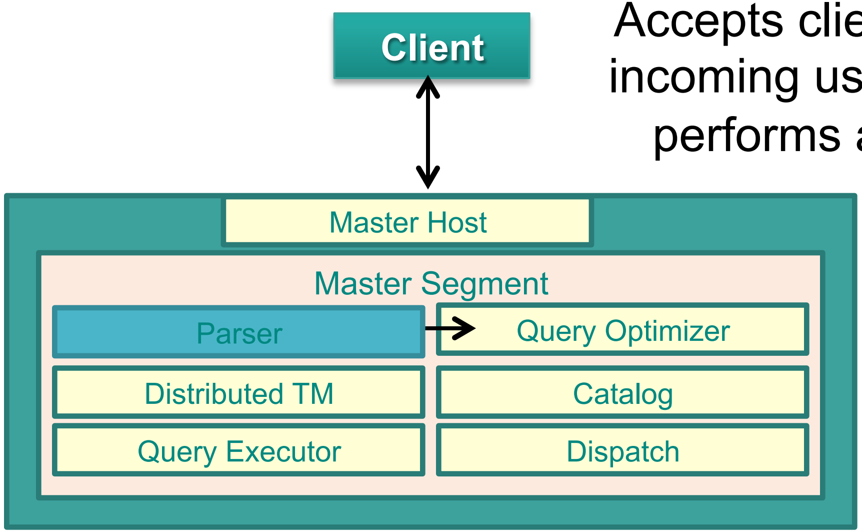
Segment Hosts have their own CPU, disk and memory (shared nothing)

High speed interconnect for continuous pipelining of data processing



Master Host

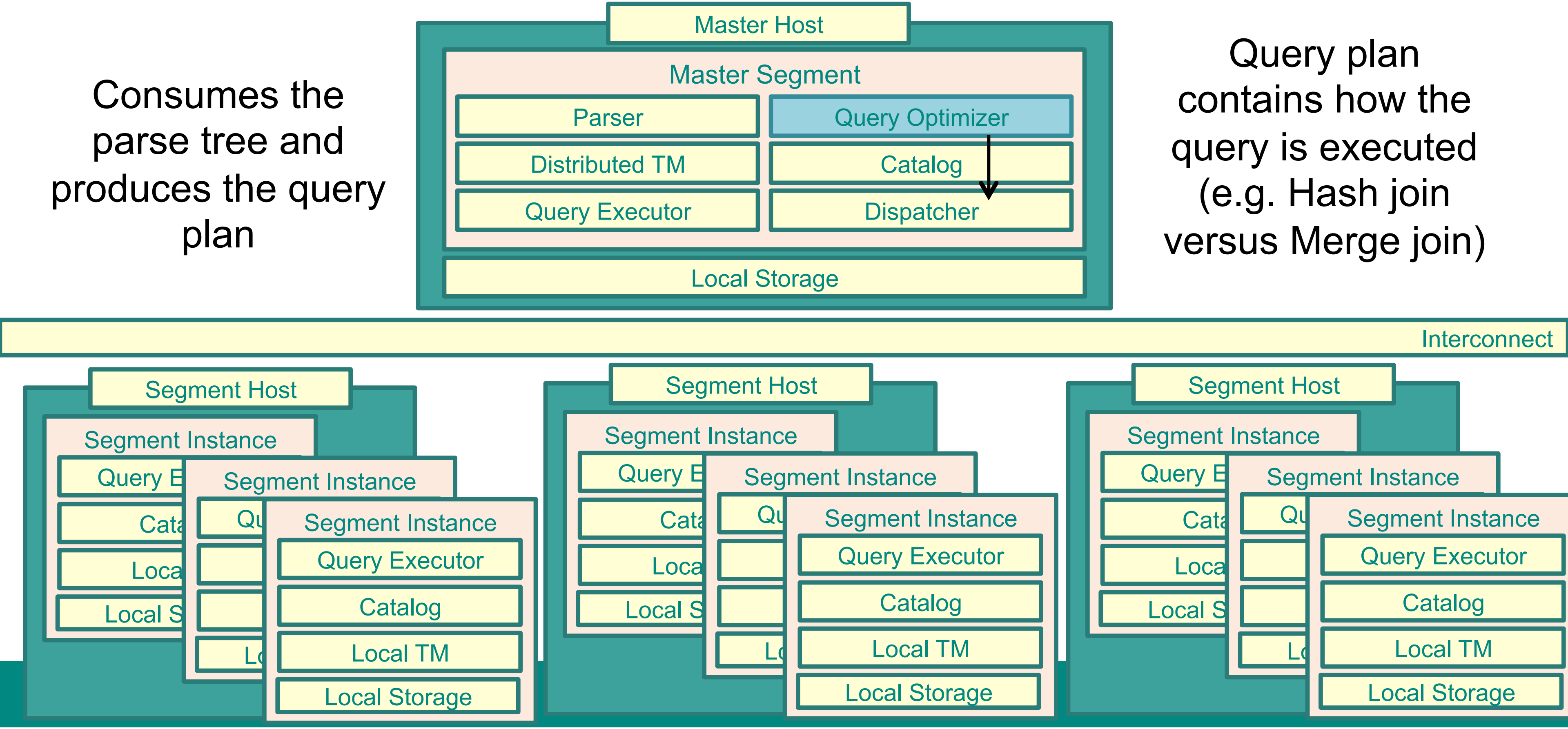
Accepts client connections, incoming user requests and performs authentication



Parser enforces syntax, semantics and produces a parse tree

Query Optimizer

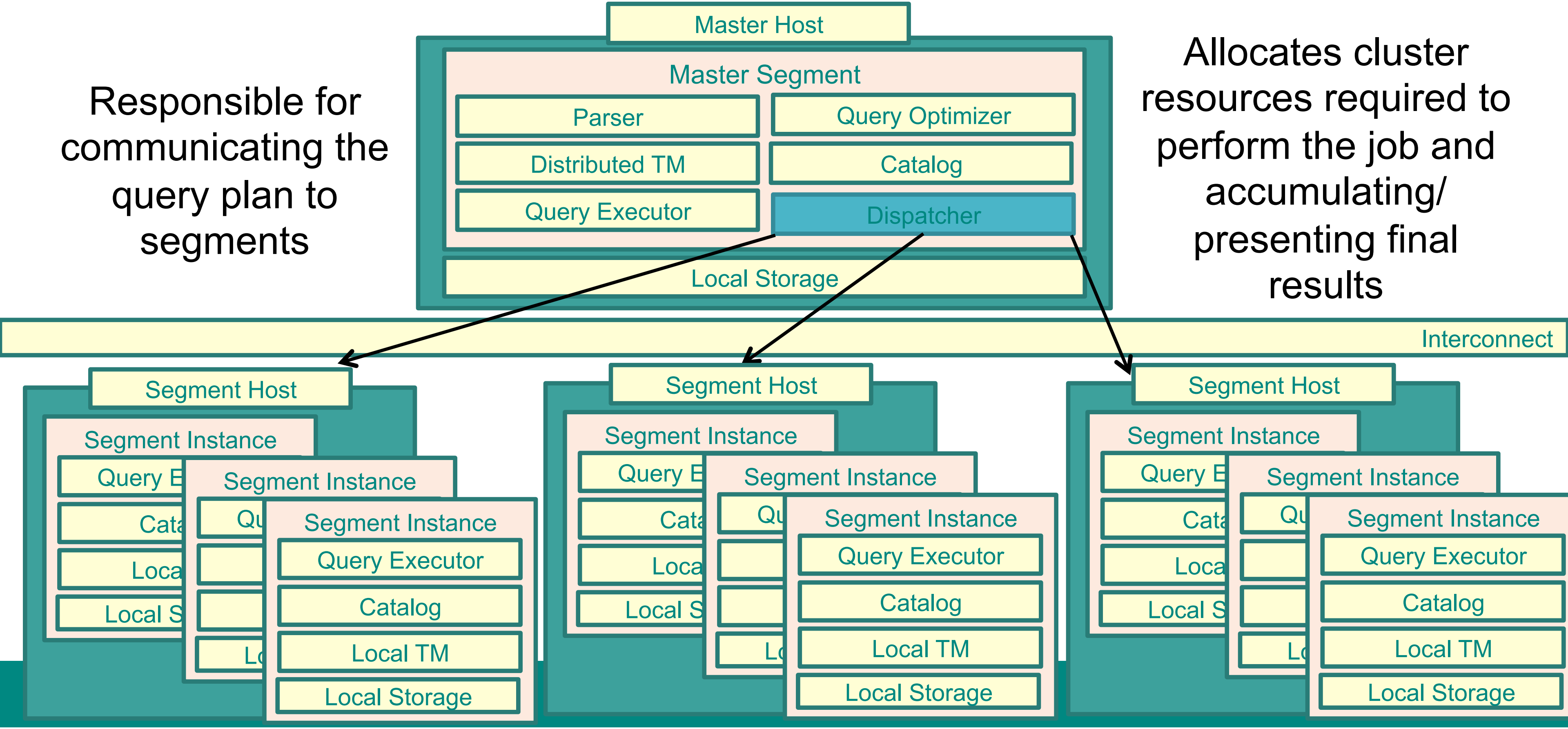
Consumes the parse tree and produces the query plan



Query plan contains how the query is executed (e.g. Hash join versus Merge join)

Query Dispatcher

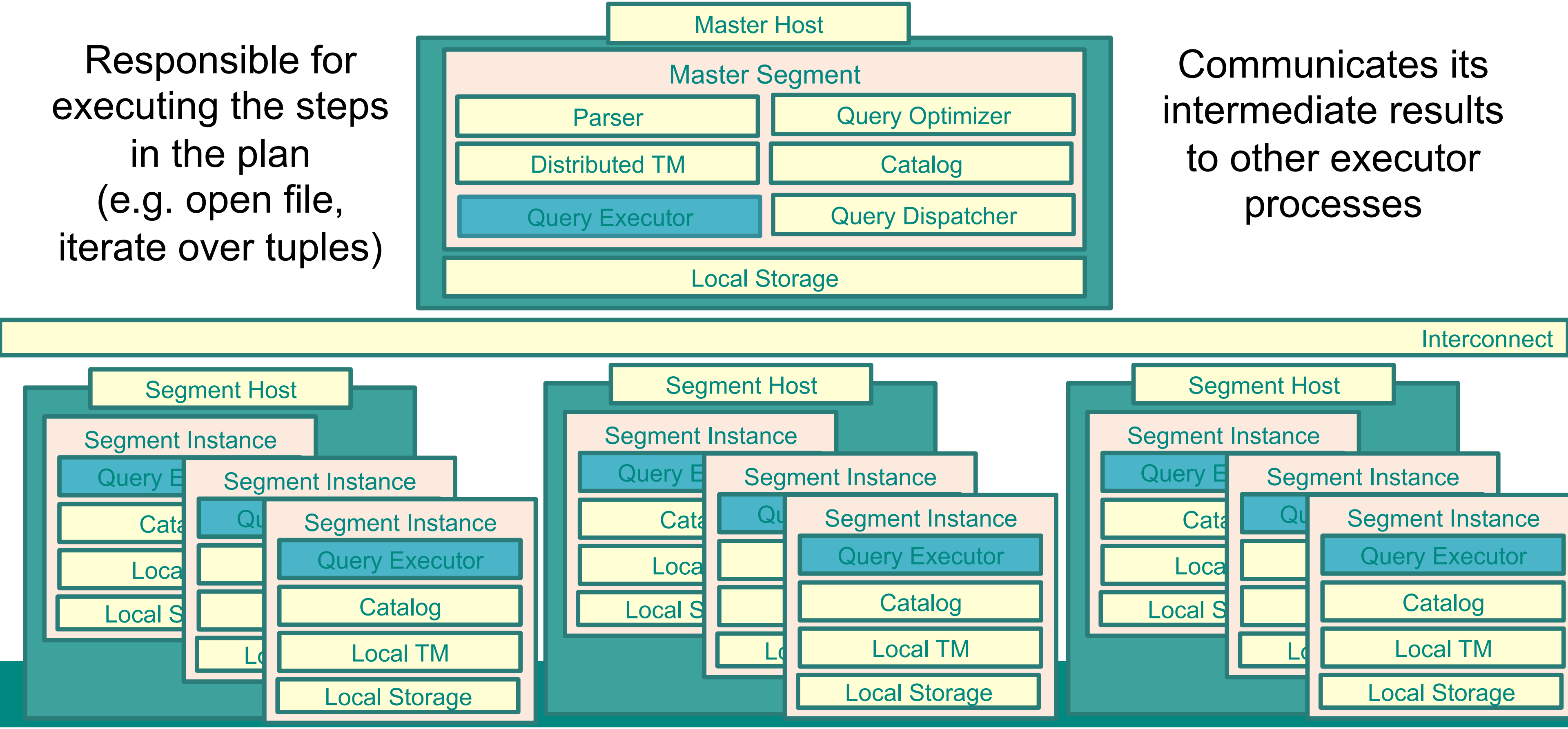
Responsible for communicating the query plan to segments



Allocates cluster resources required to perform the job and accumulating/ presenting final results

Query Executor

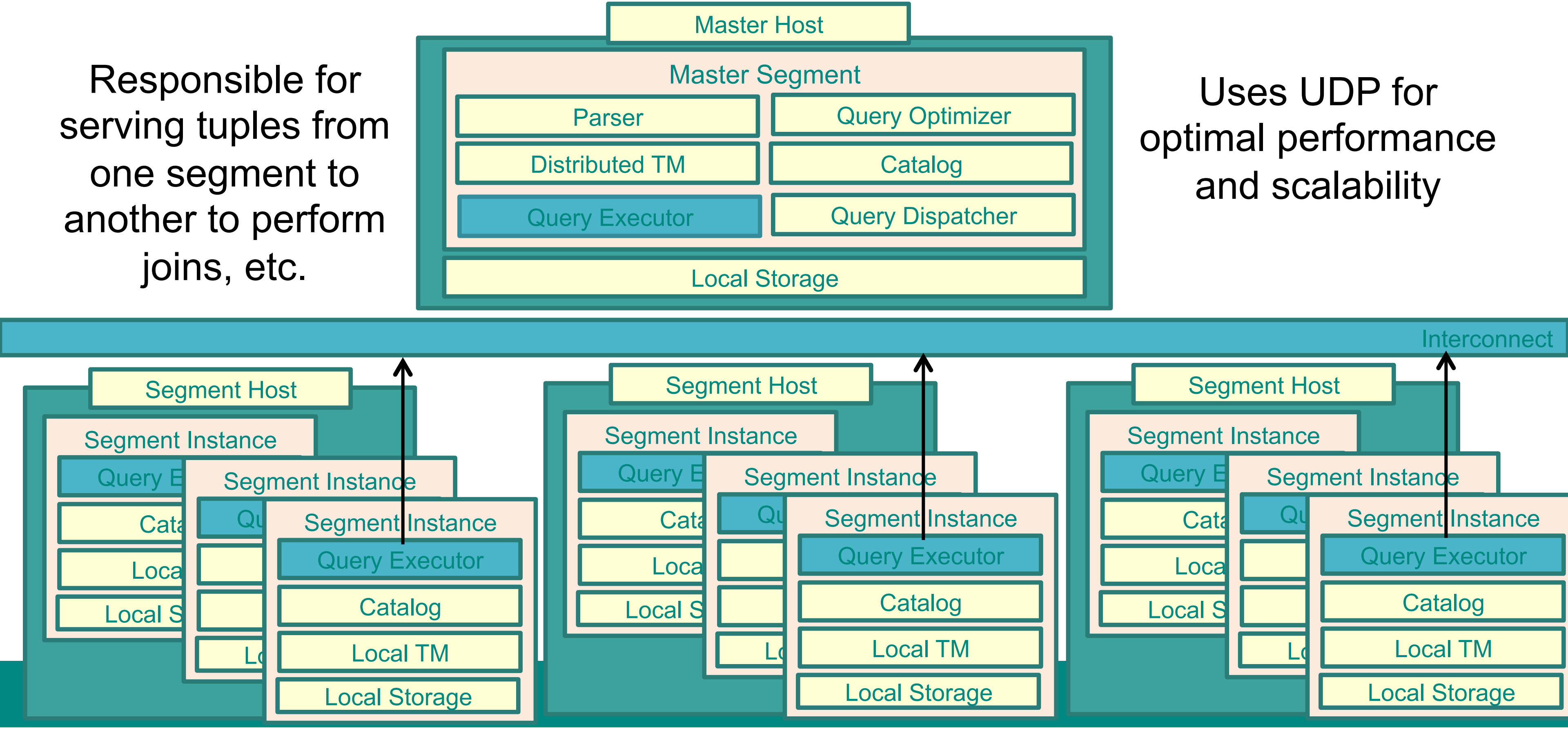
Responsible for executing the steps in the plan
(e.g. open file, iterate over tuples)



Communicates its intermediate results to other executor processes

Interconnect

Responsible for serving tuples from one segment to another to perform joins, etc.

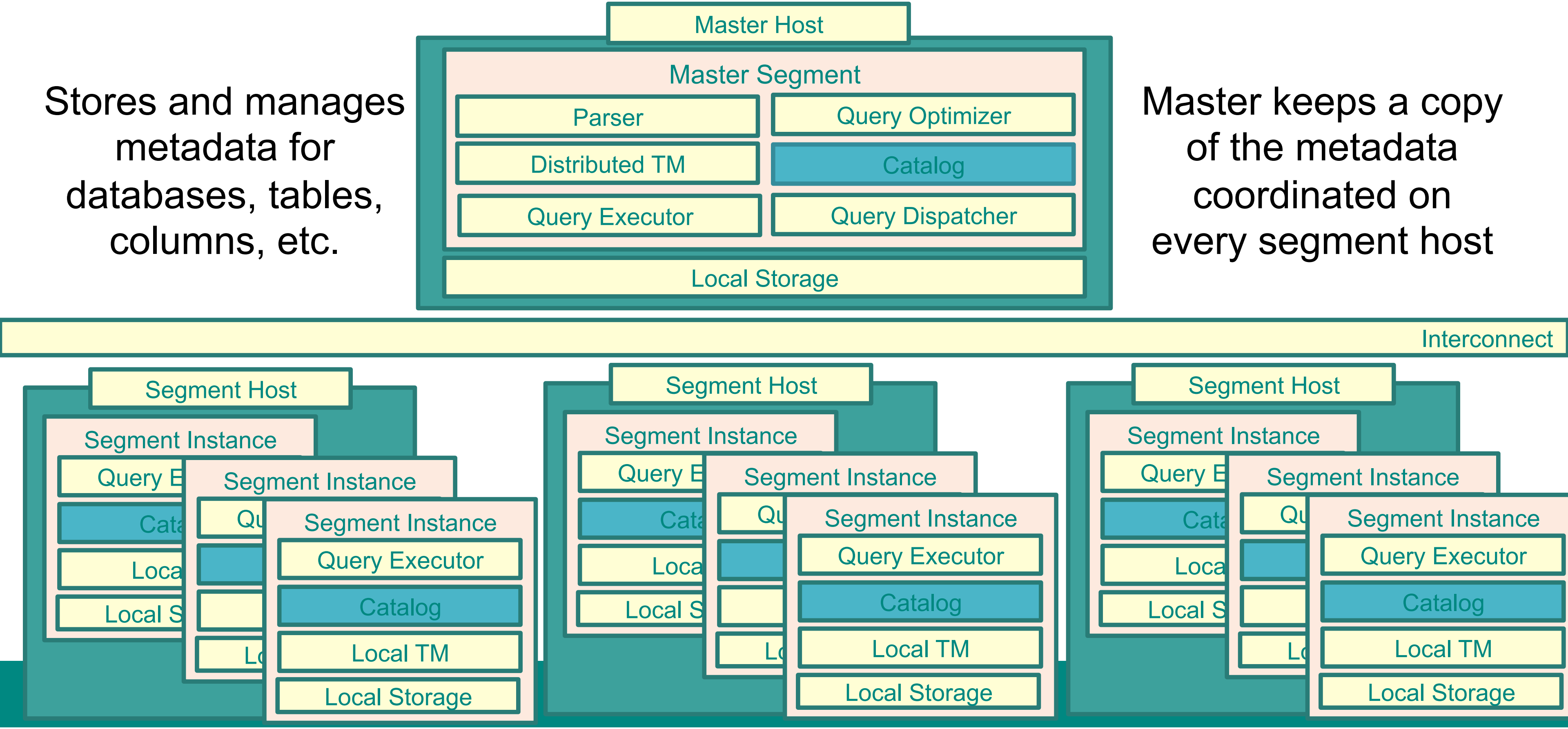


Uses UDP for optimal performance and scalability

System Catalog

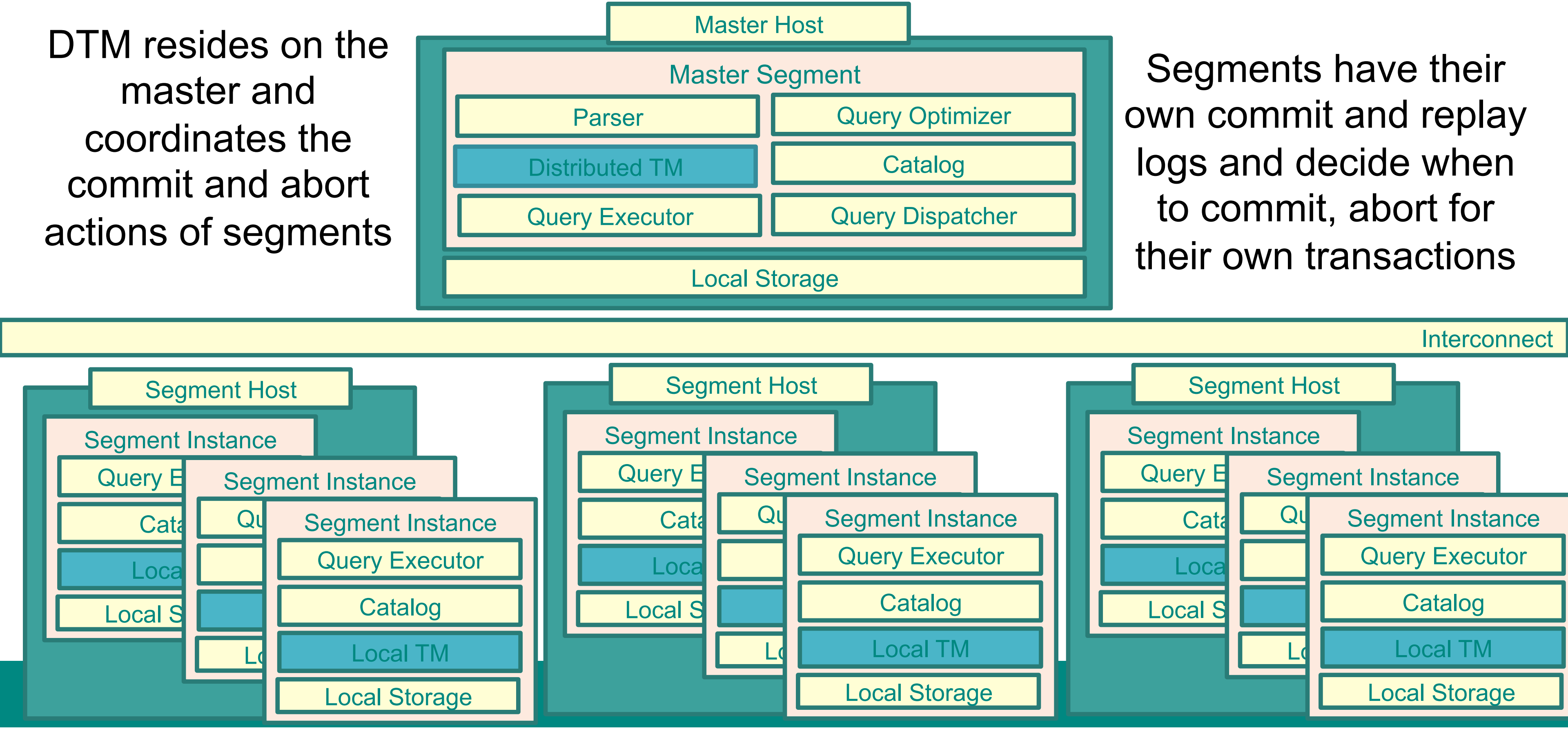
Stores and manages metadata for databases, tables, columns, etc.

Master keeps a copy of the metadata coordinated on every segment host



Distributed Transaction Management

DTM resides on the master and coordinates the commit and abort actions of segments



Segments have their own commit and replay logs and decide when to commit, abort for their own transactions

GPDB High Availability

- Master Host mirroring
  - Warm Standby Master Host
    - Replica of Master Host system catalogs
  - Eliminates single point of failure
  - Synchronization process between Master Host and Standby Master Host
    - Uses replication logs
- Segment mirroring
  - Creates a mirror segment for every primary segment
    - Uses a file block replication process
  - If a primary segment becomes unavailable automatic failover to the mirror

Master Mirroring

- Warm Standby Master enabled at initialization or on an active system using gpinitstandby
- If Master Host becomes unavailable the replication process is stopped
  - Replication logs are used to reconstruct the state of the master at the time of failure
  - Standby Master Host can be activated to start at the last successful transaction completed by the Master Host
    - Use gpactivatestandby

Segment Mirroring

- Enabled at initialization or on an active system using gpaddmirrors
- Can be configured on same array of hosts or a system outside of the array
- If a primary segment becomes unavailable automatic failover to the mirror

Pivotal™

Fault Detection and Recovery

- ftsprobe fault detection process monitors and scans segments and database processes at configurable intervals
- Use gpstate utility to verify status of primary and mirror segments
- Query *gp_segment_configuration* catalog table for detailed information about a failed segment
  - $ psql -c "SELECT * FROM gp_segment_configuration WHERE status='d';"
- When ftsprobe cannot connect to a segment it marks it as down
  - Will remain down until administrator manually recovers the failed segment using gprecoverseg utility
- Automatic failover to the mirror segment
  - Subsequent connection requests are switched to the mirror segment

Pivotal™

Agenda

- *Introduction*
- *GPDB Architecture Overview*
- **Distribution and Partitioning**
- Loading External Data
- Maintenance Procedures
- Performance Considerations
- Competitive

CREATE TABLE

Define Data Distributions

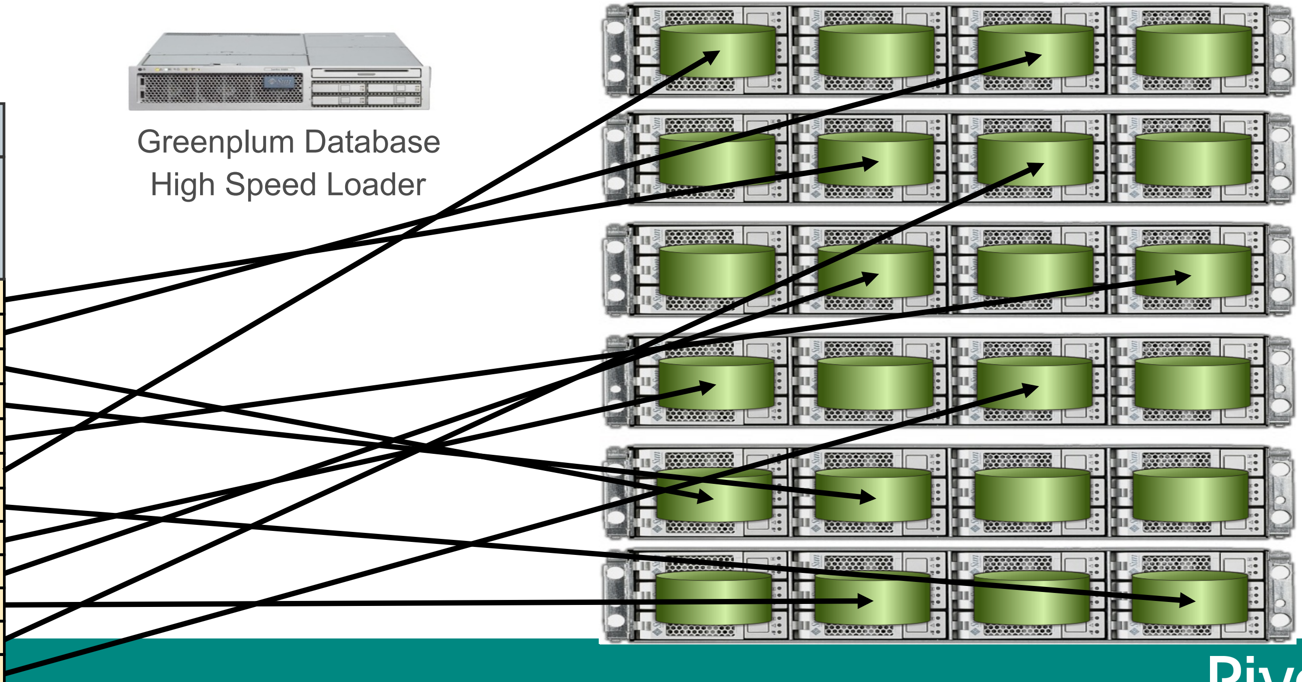
- Every table has a distribution method
- DISTRIBUTED BY (column)
  - Uses a hash distribution
- DISTRIBUTED RANDOMLY
  - Uses a random distribution which is not guaranteed to provide a perfectly even distribution

```
=> CREATE TABLE products
    (name varchar(40), prod_id integer, supplier_id integer)
    DISTRIBUTED BY (prod_id);
```

Data Distribution: The Key to Parallelism

The **primary** strategy and **goal** is to spread data **evenly** across as many nodes (and disks) as possible

Order		
Order #	Order Date	Customer ID
43	Oct 20 2005	12
64	Oct 20 2005	111
45	Oct 20 2005	42
46	Oct 20 2005	64
77	Oct 20 2005	32
48	Oct 20 2005	12
50	Oct 20 2005	34
56	Oct 20 2005	213
63	Oct 20 2005	15
44	Oct 20 2005	102
53	Oct 20 2005	82
55	Oct 20 2005	55



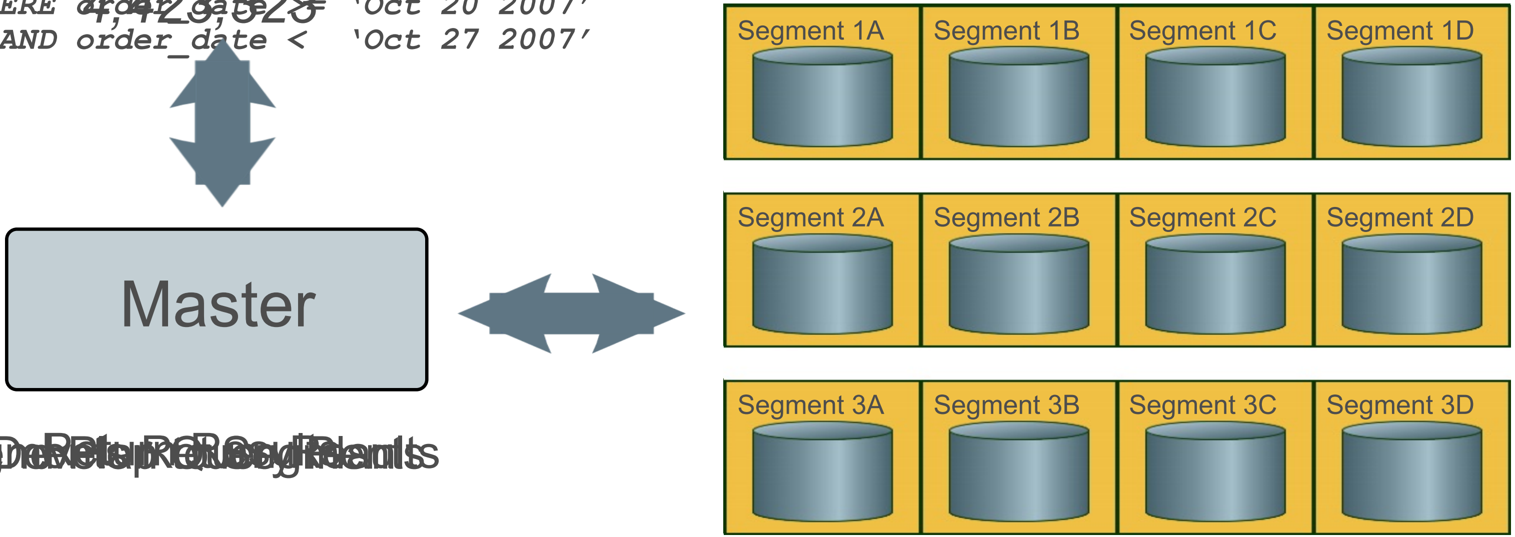
Greenplum Database High Speed Loader

Pivotal

Parallel Data Scans

```
SELECT COUNT(*)
  FROM orders
 WHERE order_date >= 'Oct 20 2007'
   AND order_date <   'Oct 27 2007'
```

4,423,323



Develop Query Plan

Segments Return Results

Each Segment Scans Data Simultaneously

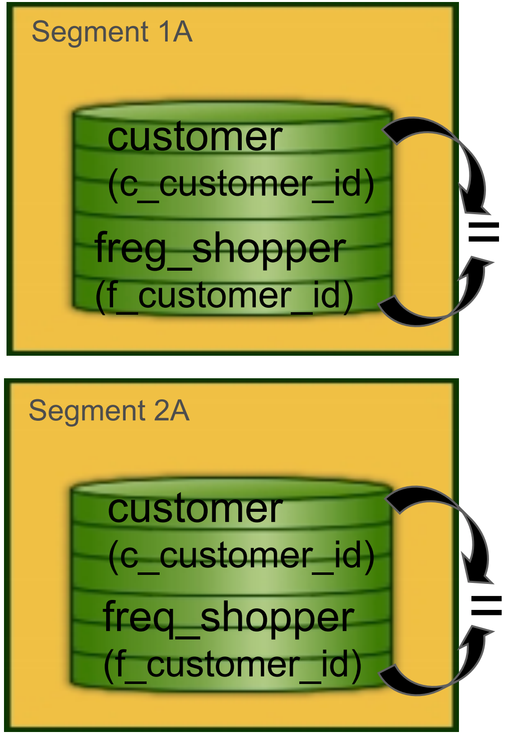
DISTRIBUTED RANDOMLY

- Uses a random algorithm
  - Distributes data across all segments
  - Minimal data skew but not guaranteed to have a perfectly even distribution
- Any query that joins to a table that is distributed randomly will require a motion operation
  - Redistribute motion
  - Broadcast motion

DISTRIBUTED BY (*column_name*)

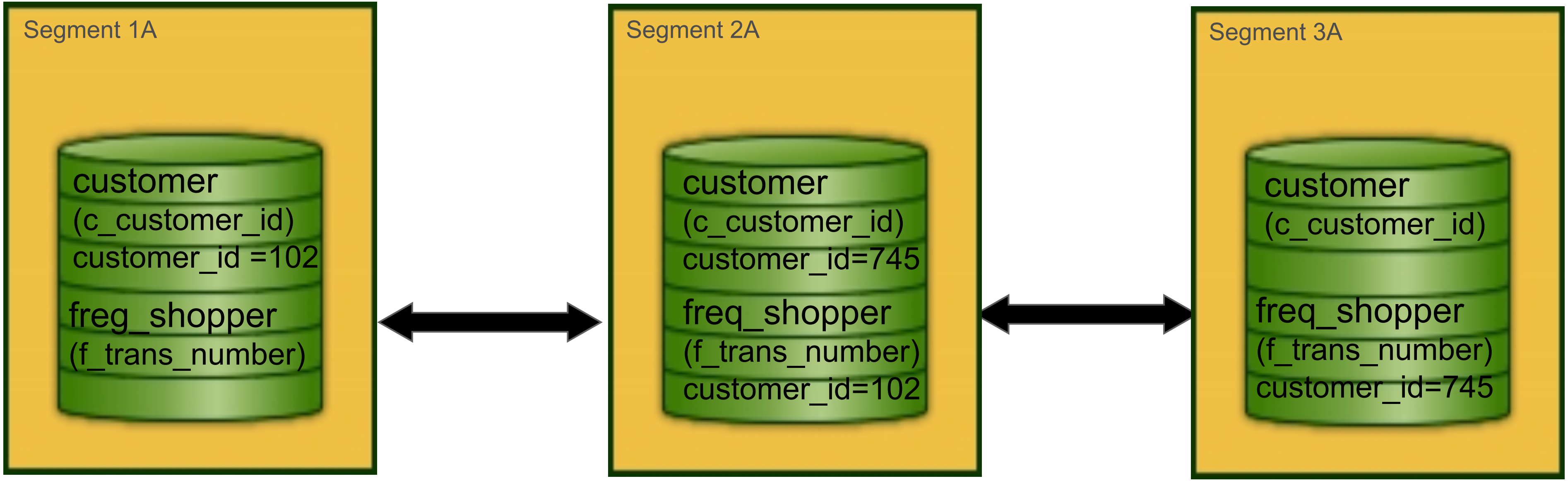
- For large tables significant performance gains can be obtained with local joins (co-located joins)
  - Distribute on the same column for tables commonly joined together
    - WHERE clause
- Join is performed within the segment
  - Segment operates independently of other segments
- Eliminates or minimizes motion operations
  - Broadcast motion
  - Redistribute motion

Use the Same Distribution Key for Commonly Joined Tables



Distribute on the same key used in the join (WHERE clause) to obtain local joins

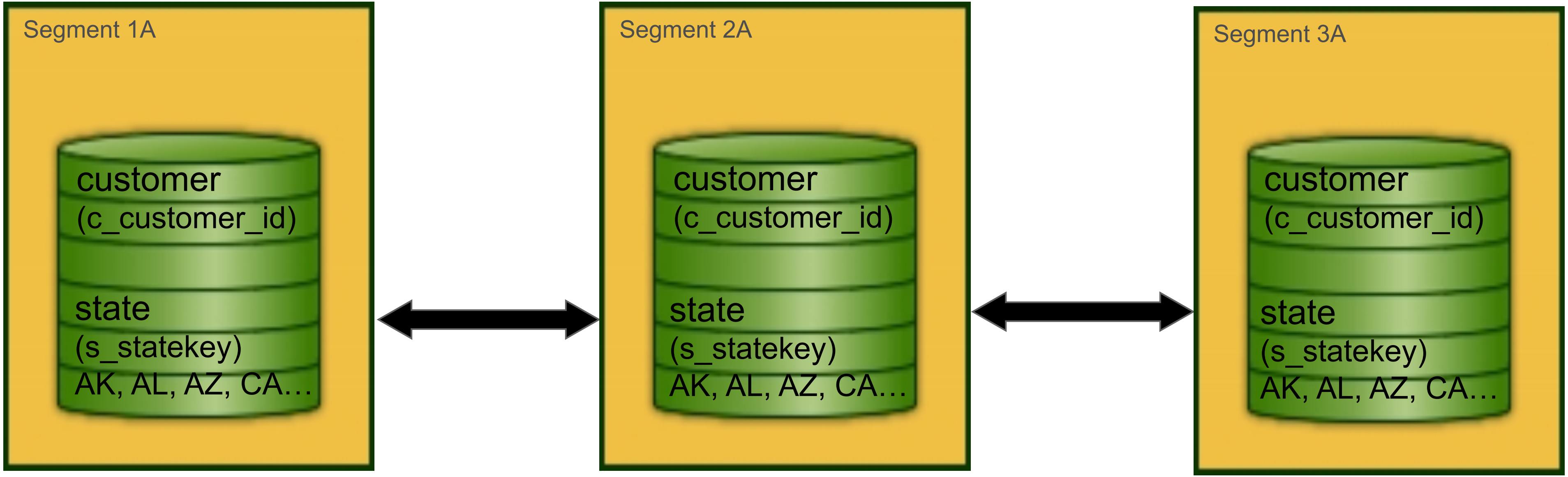
Redistribution Motion



WHERE customer.c_customer_id = freg_shopper.f_customer_id

freq_shopper table is dynamically redistributed on f_customer_id

Broadcast Motion



WHERE customer.c_statekey = state.s_statekey

The state table is dynamically broadcasted to all segments

Commonly Joined Tables Use the Same Data Type for Distribution Keys

customer (c_customer_id)	745::int
freq_shopper (f_customer_id)	745::varchar(10)

- Values might appear the same but they are stored differently at the disk level
- Values might appear the same but they HASH to different values
  - Resulting in *like* rows being stored on different segments
  - Requiring a redistribution before the tables can be joined

Hash Distributions: Data Skew and Computational Skew

- Select a distribution key with unique values and high cardinality that will not result in data skew
  - Do not distribute on boolean keys and keys with low cardinality
    - The system distributes rows with the same hash value to the same segment instance therefore resulting in the data being located on only a few segments
- Select a distribution key that will not result in computational skew (in flight when a query is executing)
  - Operations on columns that have low cardinality or non-uniform distribution

CREATE TABLE

Define Partitioning

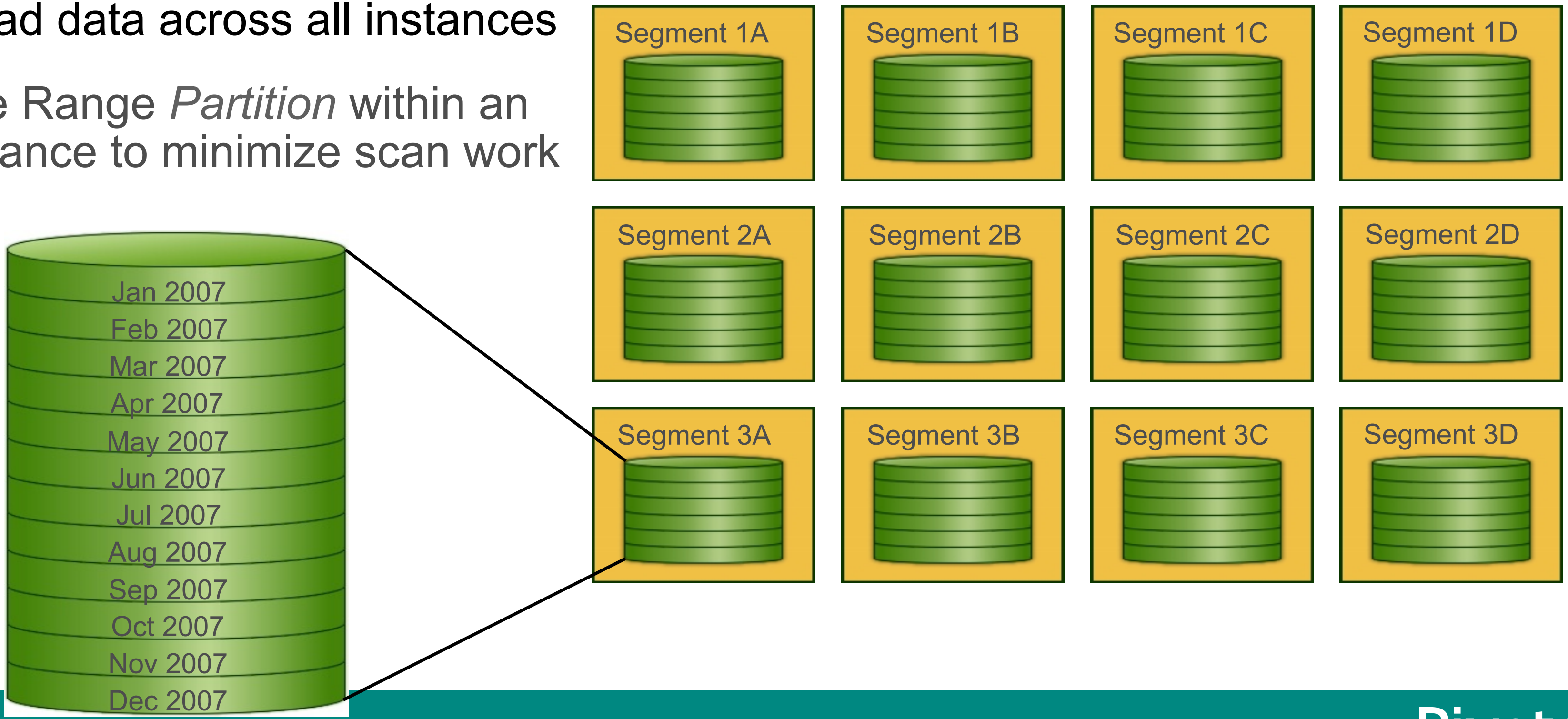
- Reduces the amount of data to be scanned by reading only the relevant data needed to satisfy a query
  - The **goal** is to achieve partition elimination
- Supports range partitioning and list partitioning
- Uses table inheritance and constraints
  - Persistent relationship between parent and child tables

Pivotal™

Multi-Level Partitioning….

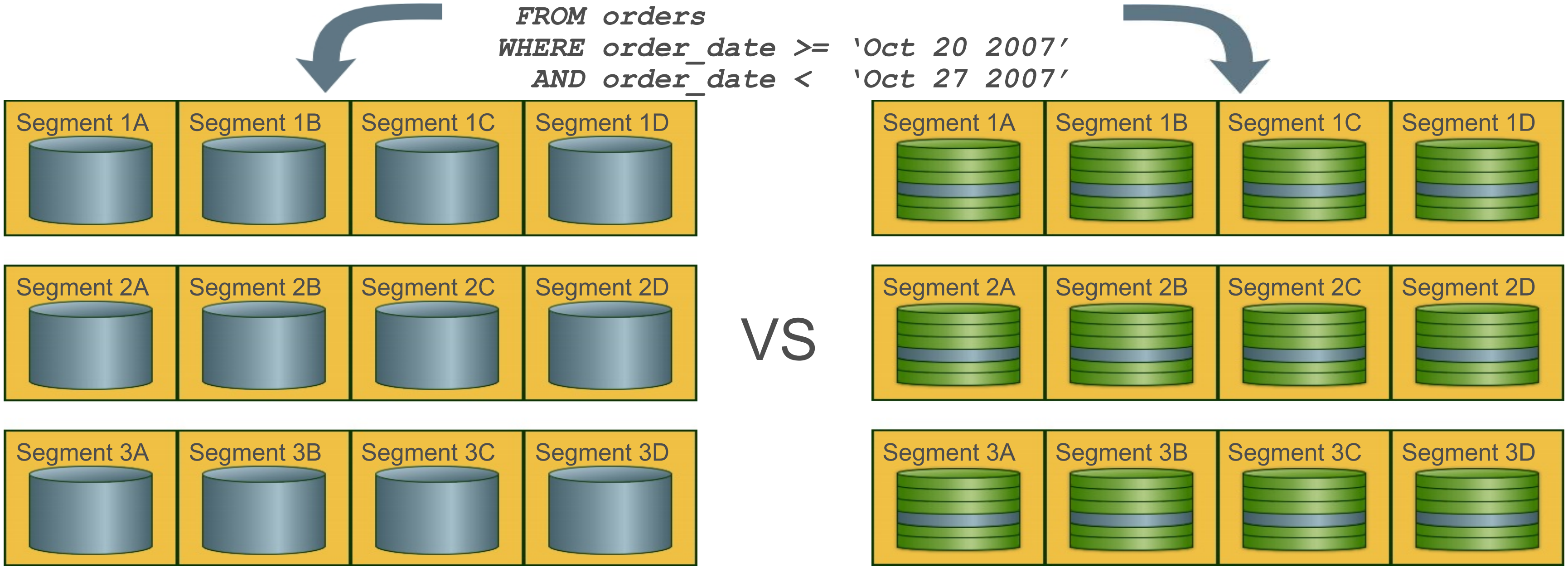
Use Hash *Distribution* to evenly spread data across all instances

Use Range *Partition* within an instance to minimize scan work



…Further Improve Scan Times

```
SELECT COUNT(*)
  FROM orders
 WHERE order_date >= 'Oct 20 2007'
   AND order_date <  'Oct 27 2007'
```



Hash Partition

Multi-Level Partition

Pivotal™

Partitioning Guidelines

- Use table partitioning on large tables to improve query performance
  - Table partitioning is not a substitute for distributions
- Use if the table can be divided into rather equal parts based on a defining criteria
  - For example, range partitioning on date
  - No overlapping ranges or duplicate values
- And the defining partitioning criteria is the same access pattern used in query predicates
  - WHERE date = ‘1/30/2012’

Pivotal™

Agenda

- *Introduction*
- *GPDB Architecture Overview*
- *Distribution and Partitioning*
- **Loading External Data**
- Maintenance Procedures
- Performance Considerations
- Competitive

GPDB Data Loading Options

Loading Method	Common Uses	Examples

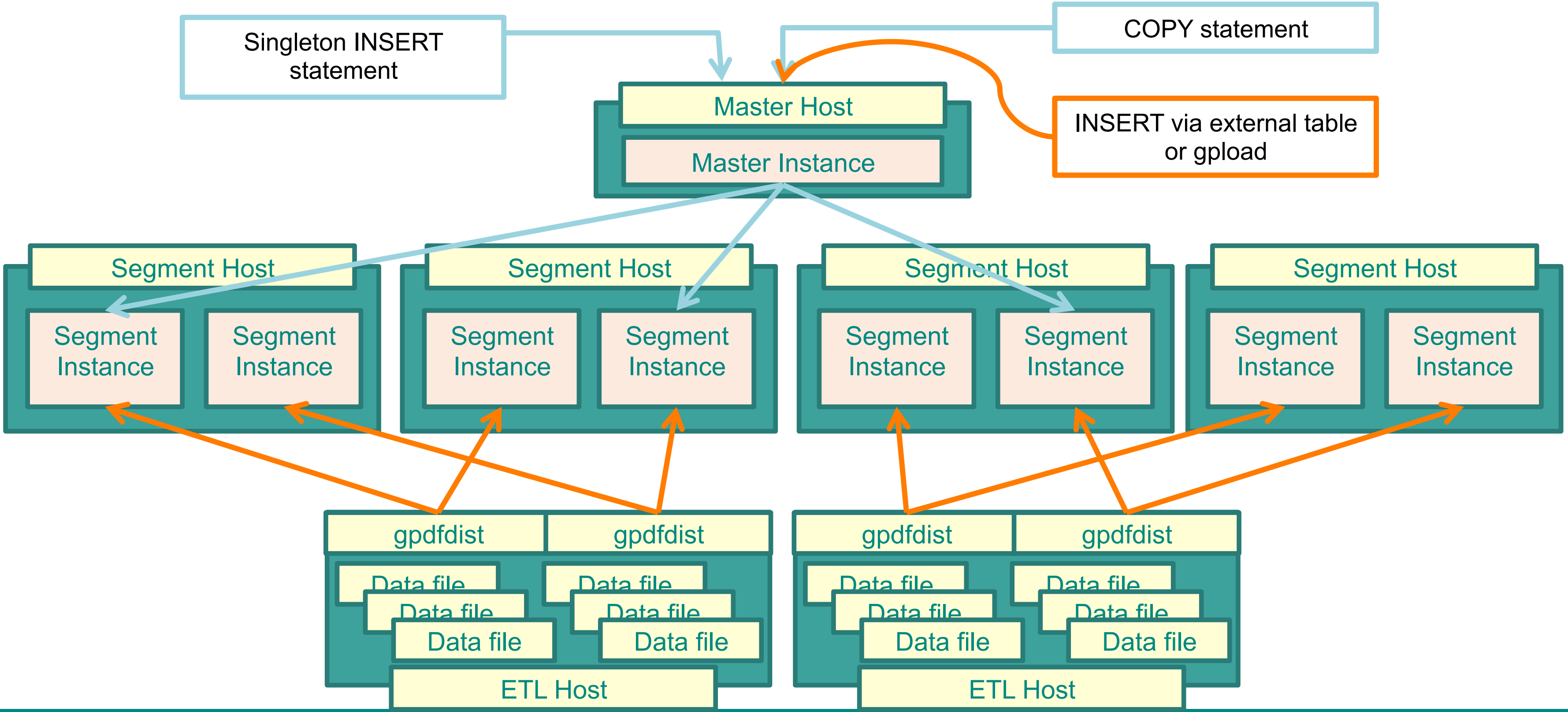
INSERTS	• Operational Workloads
	• OBDC/JDBC Interfaces
		INSERT INTO performers
		(name, specialty)
		VALUES
		(‘Sinatra’, ‘Singer’);

COPY	• Quick and easy data in • Legacy PostgreSQL applications • Output sample results from SQL statements	COPY performers FROM ‘/tmp/comedians.dat’ WITH DELIMITER ‘|’;

External Tables	• High speed bulk loads
	• Parallel loading using gpfdist protocol
	• Local file, remote file, executable or HTTP based sources
		INSERT INTO craps_bets
		SELECT g.bet_type
		, g.bet_dttm
		, g.bt_amt
		FROM x_allbets b
		JOIN games g
		ON (g.id = b.game_id)
		WHERE g.name = ‘CRAPS’;

GPLOAD	• Simplifies external table method (YAML wrapper) • Supports Insert, Merge & Update	gpload –f blackjack_bets.yml

Example Load Architectures



Pivotal

External Tables

- Access external files as if they were regular database tables
- Used with gpfdist provides full parallelism to load or unload data
- Query using SQL
- Create views for external tables
- Readable external tables for loading data
  - Perform common ETL tasks
- Writeable external tables for unloading data
  - Select data from database table to insert into writeable external table
  - Send data to a data stream

File Based External Tables

- Specify format of input files
  - FORMAT clause
- Specify location of external data sources (URIs)
- Specify protocol to access external data sources
  - gpfdist
    - Provides the best performance
    - Segments access external files in parallel up to the value of gp_external_max_segments (Default 64)
  - gpfdists
    - Secure version of gpfdist
  - file://
    - Segments access external files in parallel based on the number of URIs

Web Based External Tables

- Command based
  - Output of shell command or scripts defines web table data
  - EXECUTE command
- URL based
  - Accesses data on a web server using HTTP protocol
  - Web data files must reside on a web server that segment hosts can access

Pivotal™

Load Using Regular External Tables

- File based (flat files)
  - gpfdist provides the best performance

```
=# CREATE EXTERNAL TABLE ext_expenses (name text,
date date, amount float4, category text, description text)
LOCATION
( ‘gpfdist://etlhost:8081/*.txt’, ‘gpfdst://etlhost:8082/*.txt’)
FORMAT ’TEXT' (DELIMITER ‘|’ );
```

```
$ gpfdist –d /var/load_files1/expenses –p 8081 –l /home/gpadmin/log1 &

$ gpfdist –d /var/load_files2/expenses –p 8082 –l /home/gpadmin/log2 &
```

Load Using External Web Tables

- Shell command or script based

```
=# CREATE EXTERNAL WEB TABLE log_output
(linenum int, message text)
EXECUTE '/var/load_scripts/get_log_data.sh' ON HOST
FORMAT 'TEXT' (DELIMITER '|');
```

- URL based

```
=# CREATE EXTERNAL WEB TABLE ext_expenses (name text,
date date, amount float4, category text, description text)
LOCATION ( 'http://intranet.company.com/expenses/sales/file.csv’,
)
FORMAT 'CSV' ( HEADER );
```

COPY

- Quick and easy
- Recommended for small loads
  - Not recommended for bulk loads
- Load from file or standard input
- Is not parallel uses a single process on the master
  - Can improve performance by running multiple COPY commands concurrently
  - Data must be divided across all concurrent processes
- Source file must be accessible by the master

GPLOAD

- Interface to readable external tables
  - Invokes gpfdist for parallel loading
- Creates external table based on source data defined
- Uses load specification defined in a YAML formatted control file
  - INPUT
    - Hosts, ports, file structure
  - OUTPUT
    - Target Table
    - MODES: INSERT, UPDATE, MERGE
    - BEFORE & AFTER SQL statements

Agenda

- *Introduction*
- *GPDB Architecture Overview*
- *Distribution and Partitioning*
- *Loading External Data*
- **Maintenance Procedures**
- Performance Considerations
- Competitive

ANALYZE and Database Statistics

- Updated statistics are critical for the Query Planner to generate optimal query plans
  - When a table is analyzed table information about the data is stored into system catalog tables
- Always run ANALYZE after loading data
- Run ANALYZE after INSERT, UPDATE and DELETE operations that significantly changes the underlying data
- The gp_autostats_on_change_threshold can be used in conjunction with gp_autostats_mode to auto analyze during these operations

ANALYZE [table [(column [, ...])]]

- For very large tables it may not be feasible to run ANALYZE on the entire table
- ANALYZE may be performed for specific columns
- Run ANALYZE for
  - Columns used in a JOIN condition
  - Columns used in a WHERE clause
  - Columns used in a SORT clause
  - Columns used in a GROUP BY or HAVING Clause

VACUUM

- VACUUM reclaims physical space on disk from deleted or updated rows or aborted load/insert operations
- VACUUM collects table-level statistics such as the number of rows and pages
- Run VACUUM after
  - Large DELETE operations
  - Large UPDATE operations
  - Failed load operations

Free Space Map

- Expired rows are tracked in the free space map
- Free space map size must be large enough to hold all expired rows
- VACUUM can not reclaim space occupied by expired rows that overflow the free space map
  - VACUUM FULL reclaims all expired rows space
    - Is an expensive operation
    - Takes an exceptionally long time to finish
- max_fsm_pages
- max_fsm_relations

Pivotal™

Agenda

- *Introduction*
- *GPDB Architecture Overview*
- *Distribution and Partitioning*
- *Loading External Data*
- *Maintenance Procedures*
- **Performance Considerations**
- Competitive

GPDB Optimization and Performance Tuning

- Review optimization, tuning and best practices provided in the applicable topic modules within this immersion
  - Distributions
  - Partitioning
  - Storage Orientation
  - Compression
  - Indexes
  - Loading
  - ANALYZE
  - Query Plans
  - VACUUM

Data Types and Byte Alignment

- Lay out columns in heap tables as follows
  - 8 byte first (bigint, timestamp)
  - 4 byte next (int, date)
  - 2 byte last (smallint)
- Put distribution and partition columns up front
  - Two 4 byte columns = an 8 byte column
- For example
  Int, Bigint, Timestamp, Bigint, Timestamp, Int (distribution key), Date (partition key), Bigint, Smallint --> Int (distribution key), Date (partition key), Bigint, Bigint, Timestamp, Bigint, Timestamp, Int, Smallint

Set Based versus Row Based

- PL/SQL and other procedural languages utilize cursors to operate on one record at a time
  - Typically, these programs are written by programmers not database experts
  - Looping over a set of records returned by a query results in an additional query plan per record
- GPDB performs better when dealing with operations over a set of records
  - 1 query plan for the whole set of records versus 1 main query plan + 1 per row

Agenda

- *Introduction*
- *GPDB Architecture Overview*
- *Distribution and Partitioning*
- *Loading External Data*
- *Maintenance Procedures*
- *Performance Considerations*
- **Competitive**

Pivotal™

Competitive Solutions

	GPDB	TD	VERT	EXA	NZ
Using as ETL engine	🟢	🟢	🟠	🟠	🟡

Competitive Solutions

	GPDB	TD	VERT	EXA	NZ
Using as ETL engine	🟢	🟢	🟠	🟠	🟡
Using as BI datasource	🟡	🟢	🟢	🟡	🟠

Competitive Solutions

	GPDB	TD	VERT	EXA	NZ
Using as ETL engine	🟢	🟢	🟠	🟠	🟡
Using as BI datasource	🟡	🟢	🟢	🟡	🟠
Extensibility	🟢	🟠	🟢	🟠	🟡

Competitive Solutions

	GPDB	TD	VERT	EXA	NZ
Using as ETL engine	🟢	🟢	🟠	🟠	🟡
Using as BI datasource	🟡	🟢	🟢	🟡	🟠
Extensibility	🟢	🟠	🟢	🟠	🟡
Openness	🟢	🔴	🟡	🔴	🔴

Competitive Solutions

	GPDB	TD	VERT	EXA	NZ
Using as ETL engine	🟢	🟢	🟠	🟠	🟡
Using as BI datasource	🟡	🟢	🟢	🟡	🟠
Extensibility	🟢	🟠	🟢	🟠	🟡
Openness	🟢	🔴	🟡	🔴	🔴
HW Flexibility	🟢	🟠	🟢	🔴	🔴

Competitive Solutions

	GPDB	TD	VERT	EXA	NZ
Using as ETL engine	🟢	🟢	🟠	🟠	🟡
Using as BI datasource	🟡	🟢	🟢	🟡	🟠
Extensibility	🟢	🟠	🟢	🟠	🟡
Openness	🟢	🔴	🟡	🔴	🔴
HW Flexibility	🟢	🟠	🟢	🔴	🔴
TCO	🟡	🔴	🟠	🔴	🟠

Competitive Solutions

	GPDB	TD	VERT	EXA	NZ
Using as ETL engine	🟢	🟢	🟠	🟠	🟡
Using as BI datasource	🟡	🟢	🟢	🟡	🟠
Extensibility	🟢	🟠	🟢	🟠	🟡
Openness	🟢	🔴	🟡	🔴	🔴
HW Flexibility	🟢	🟠	🟢	🔴	🔴
TCO	🟡	🔴	🟠	🔴	🟠
Vendor Forus	🟢	🟢	🟡	🟠	🔴

Summary

- Greenplum is the first open source MPP database
- With over 400 enterprise customers
- Representing more than 10 years of development
- With great performance and scalability
- And great extensibility and analytical capabilities
- ***Community is yet young – feel free to contribute!***

Pivotal

BUILT FOR THE SPEED OF BUSINESS