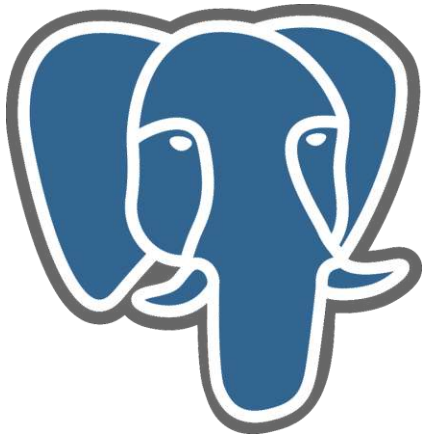
PGConf.Brasil 2018

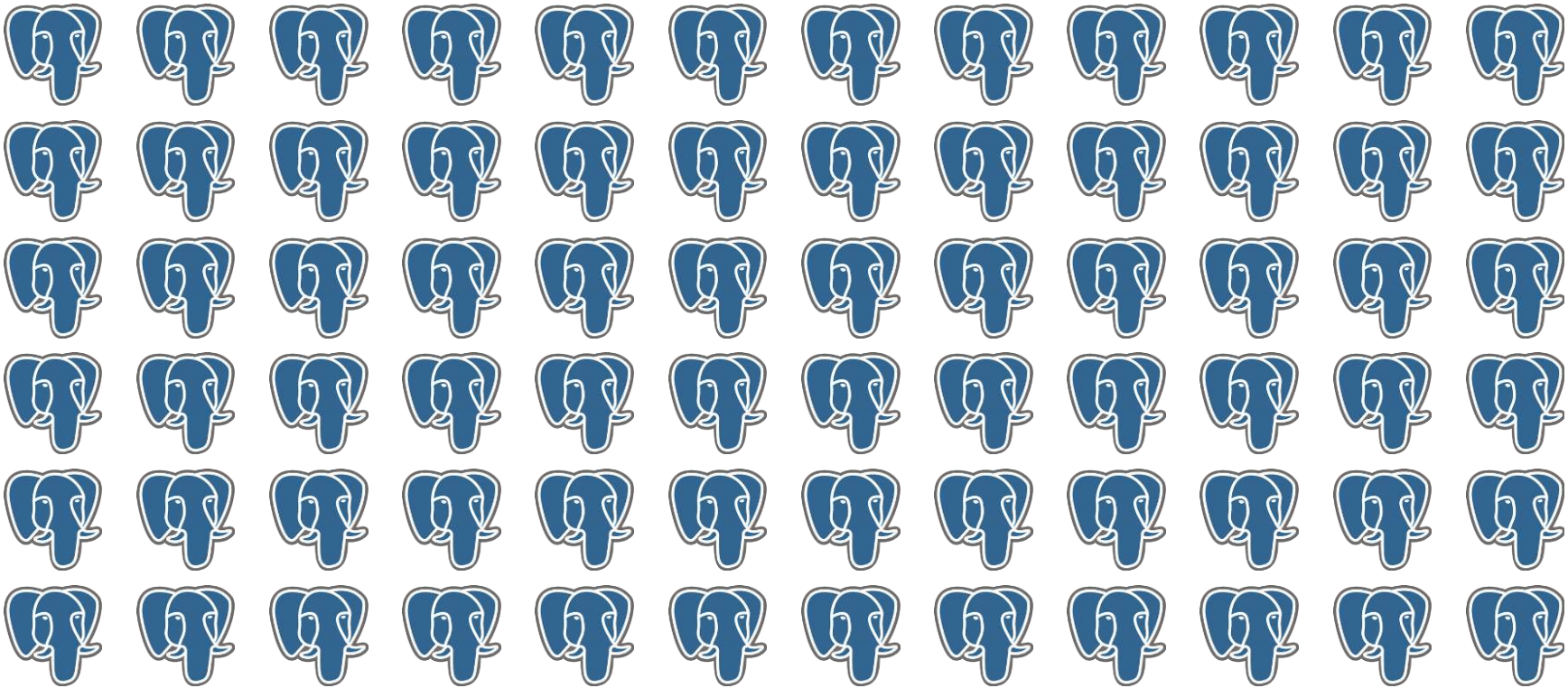
3 e 4'AGO'18 @SãoPaulo

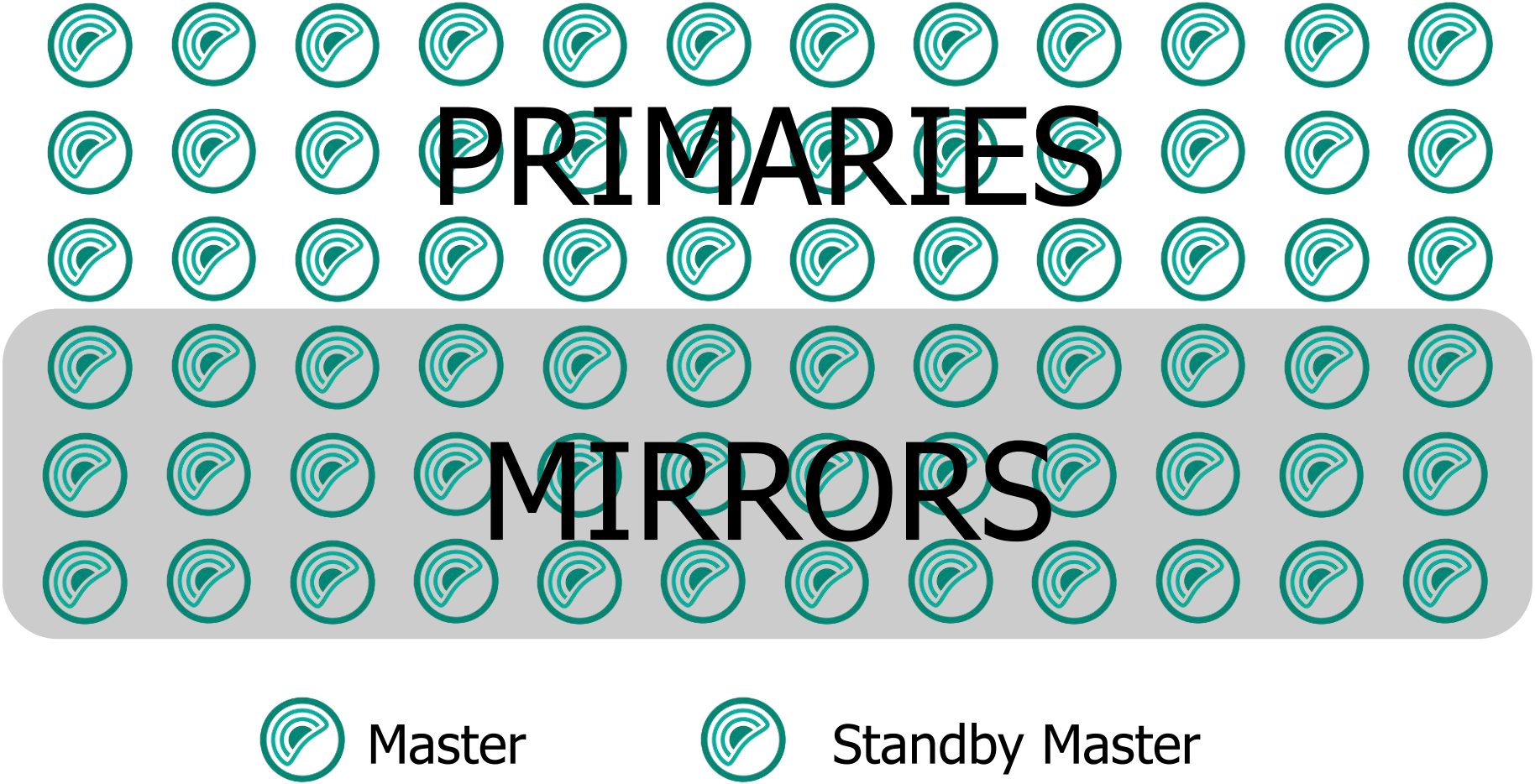
GREENPLUM DATABASE

Greenplum Database: Evolving Advanced Analytics on PostgreSQL

How to make a Greenplum?






PRIMARIES
MIRRORS
Master
Standby Master

PostgreSQL Integration Strategy

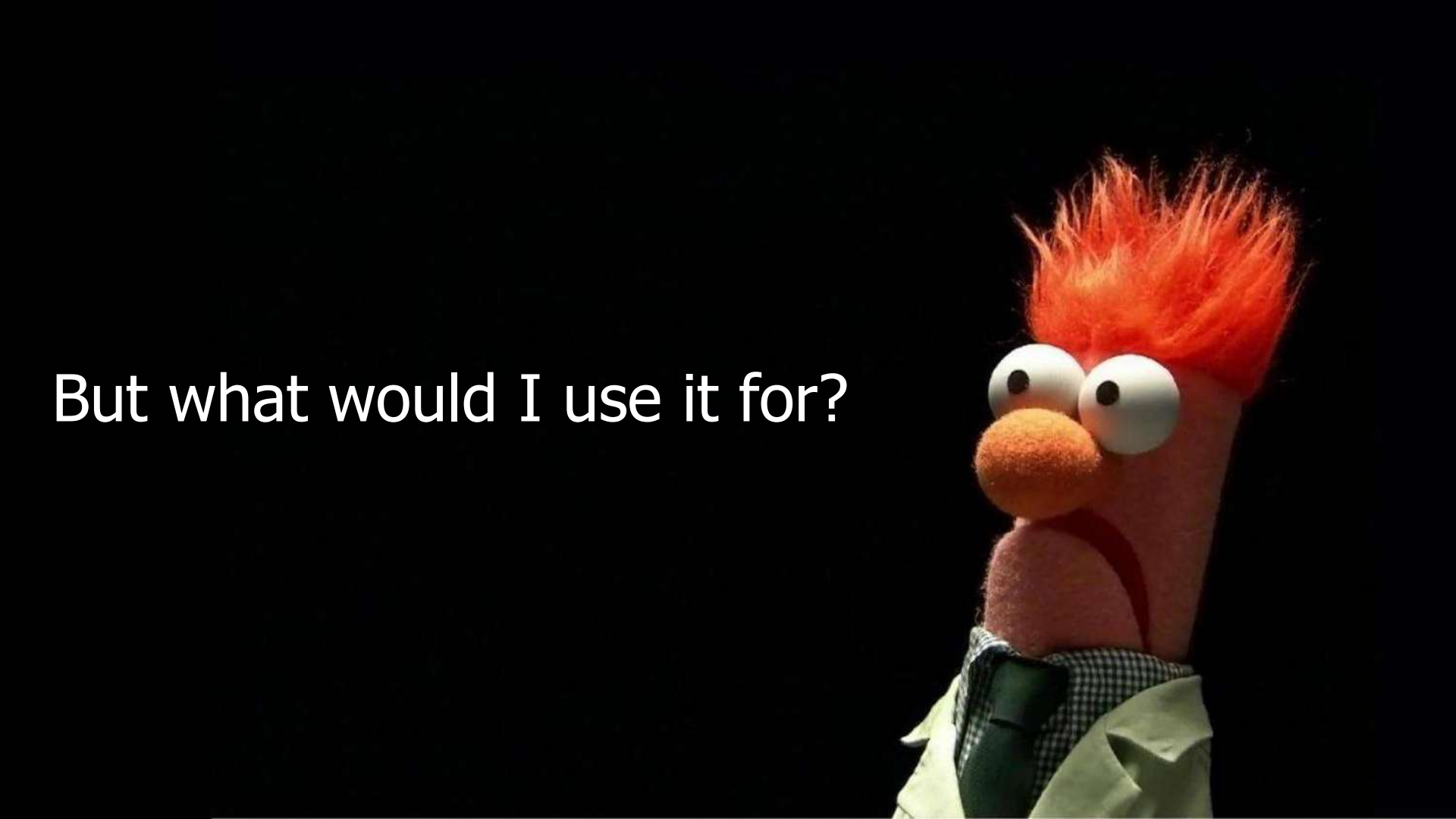
GOALS

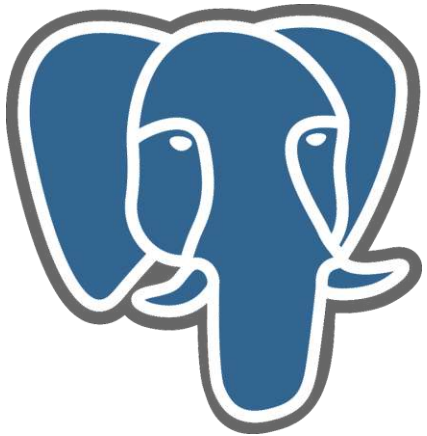
- Reduce Long Term Cost Structure
- World Wide Technical Collaboration
- Reduce Bespoke Technologies
- Avoid Proprietary Pockets

INITIATIVES

- 8.3 in September, 2017
- Based on 9.2 Today
- Goal of 9.4 for next milestone
- Reach PG 11 and stay in sync
- Innovate on Greenplum Database MPP, Keep PG intact

But what would I use it for?



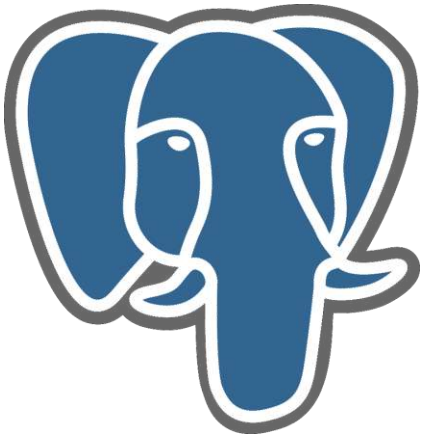


You have data and you want to ask it questions...

Geospatial

Graph

Relational data



Text searching

UDF’s via Python, R or anything you can run in a container!

You have data and you want to ask it questions...

Geospatial

Graph

Relational data

Text searching

UDF’s via Python, R or anything you can run in a container!

… but you have LOTS of data. 10 Terabytes+

For example an internet company

… or you need to to do anomaly detection

Government Agency: Tax Fraud Detection

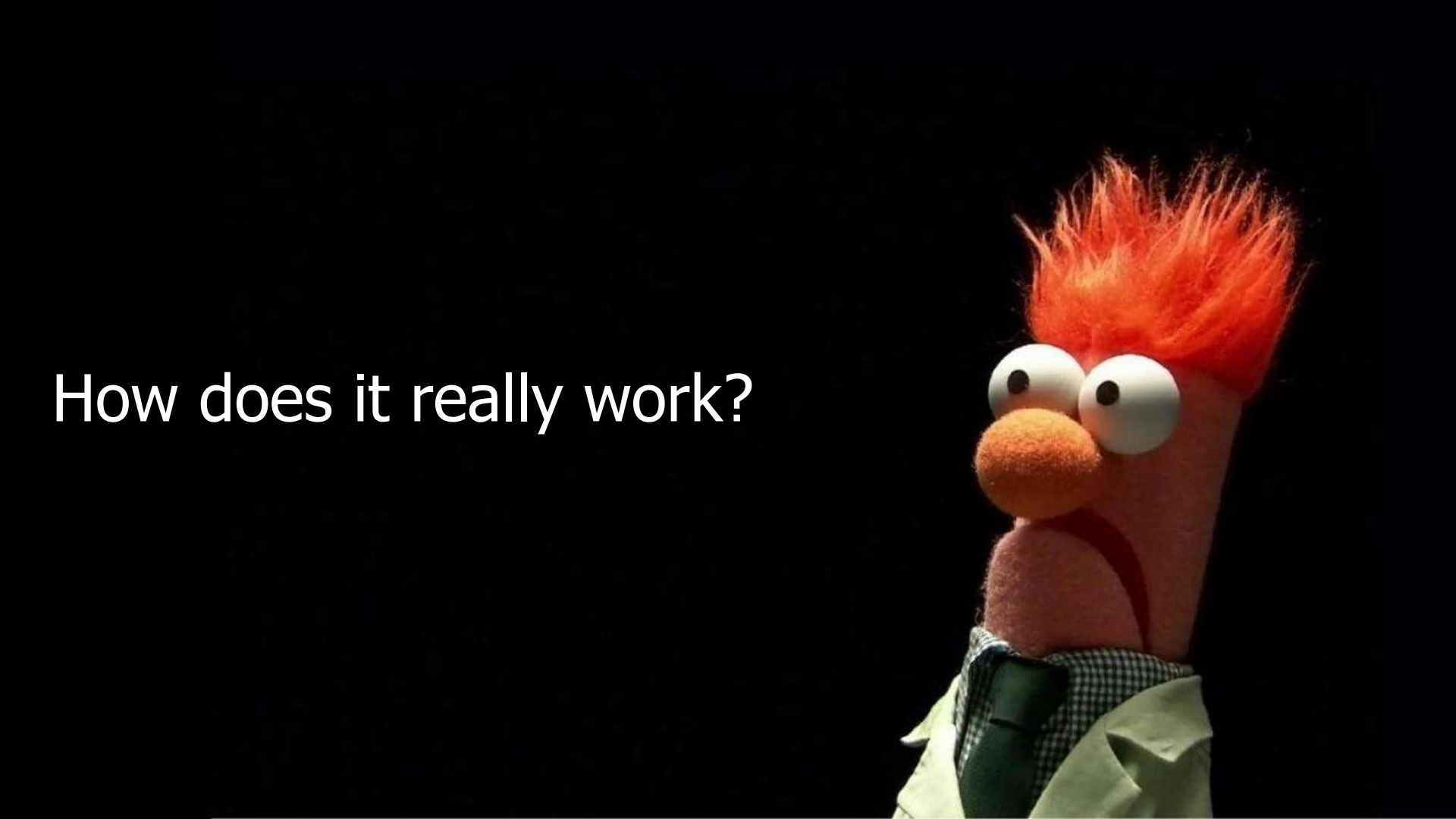
A lot of tax return data submitted in a short period of time!

… or you have very complex questions to answer!

National Institute of Information and Communications Technology (NICT) (of Japan)

- **predict and react to extreme weather events**

How does it really work?



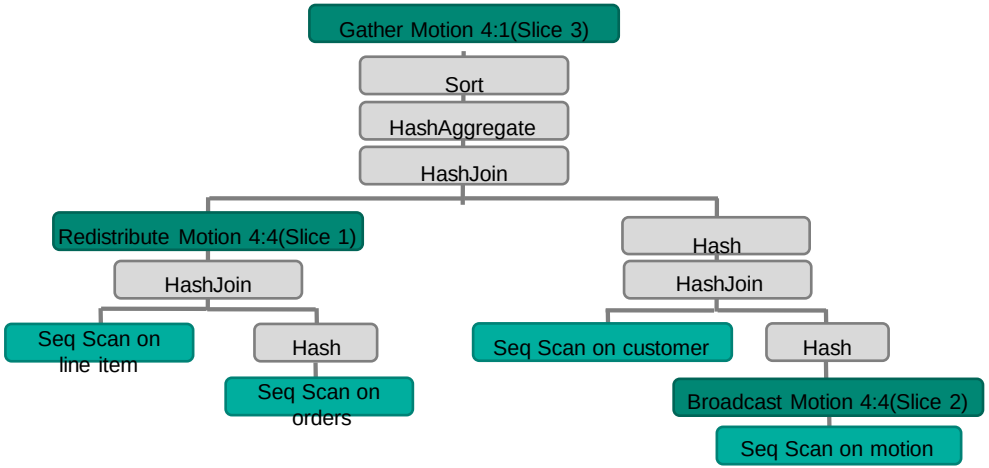
Parallel Query Execution

Extending Postgresql Execution Engine for MPP Operations

Planner creates query execution plan that is MPP aware

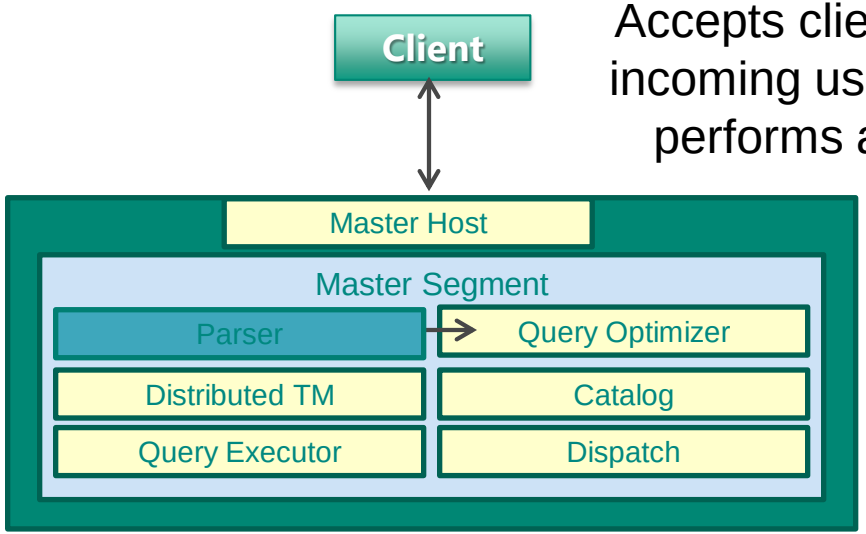
Plans are executed in parallel across segment instances

Motion operators for inter-segment communication



Master Host

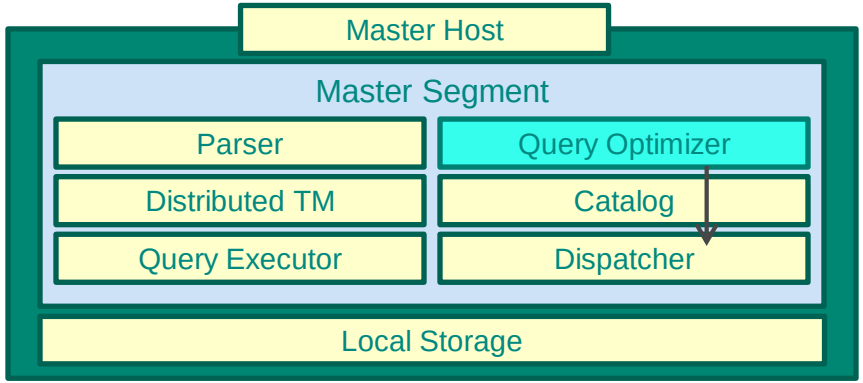
Accepts client connections, incoming user requests and performs authentication



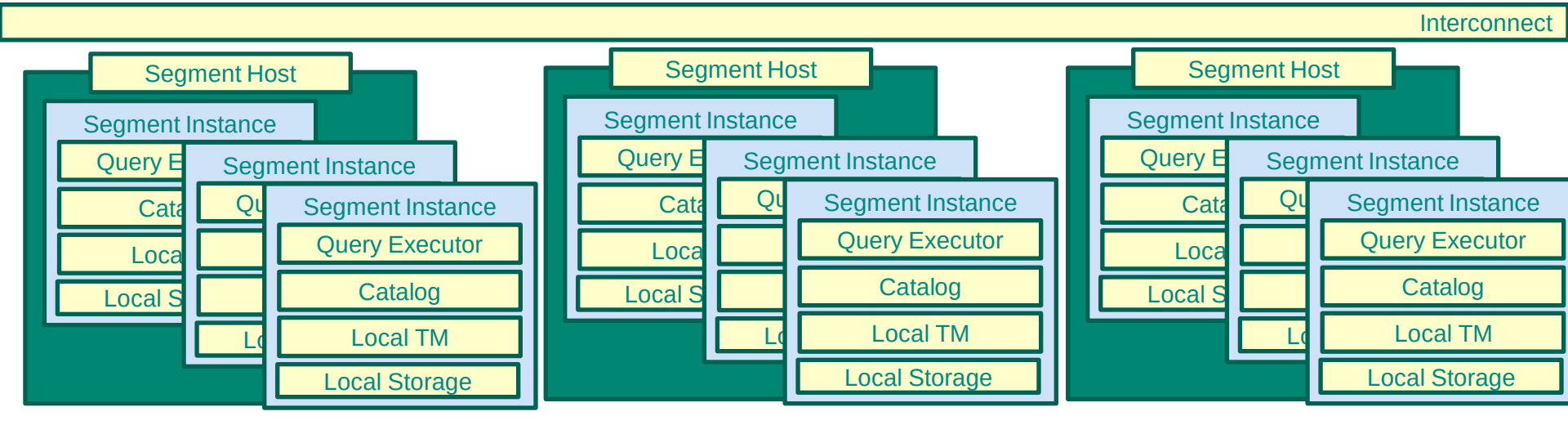
Parser enforces syntax, semantics and produces a parse tree

Query Optimizer

Consumes the parse tree and produces the query plan

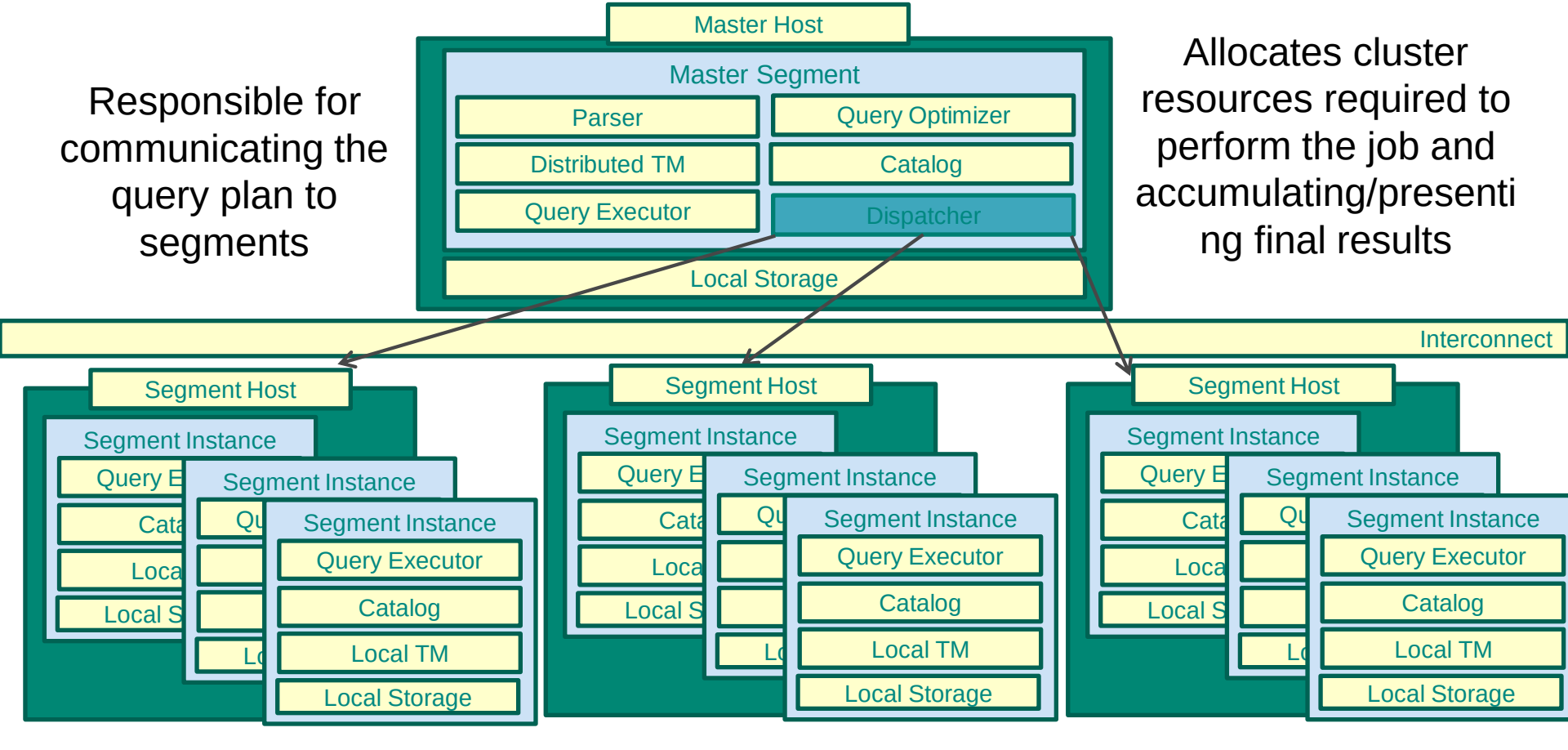


Query execution plan contains how the query is executed



Query Dispatcher

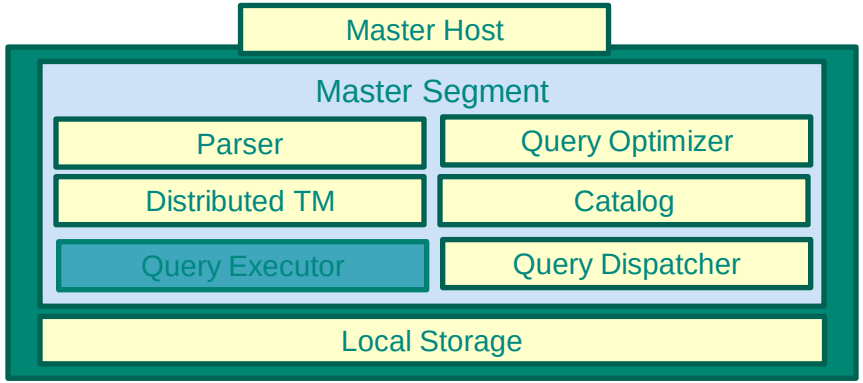
Responsible for communicating the query plan to segments



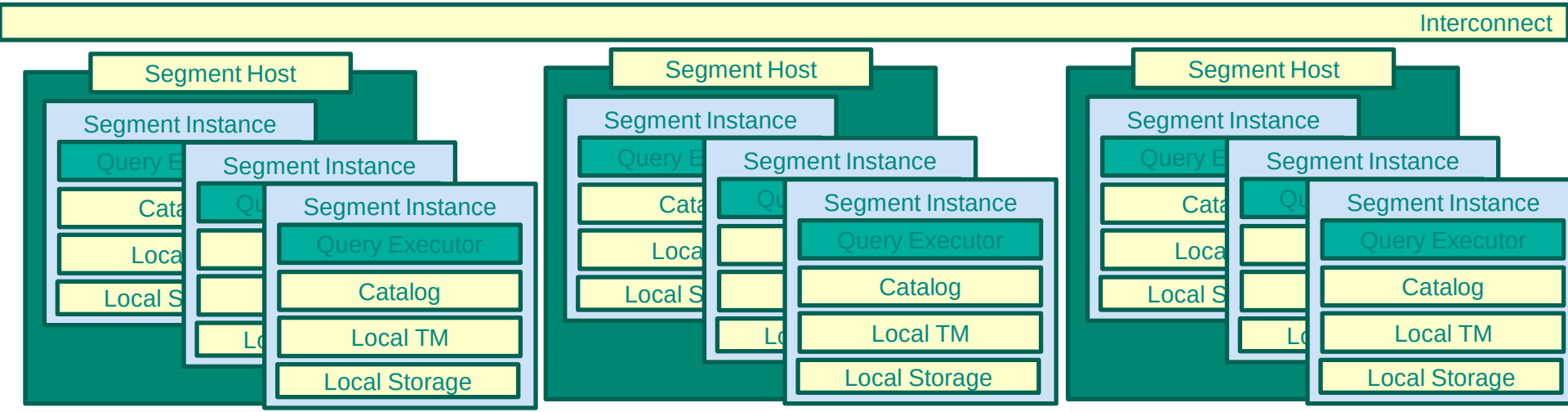
Allocates cluster resources required to perform the job and accumulating/presenting final results

Query Executor

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

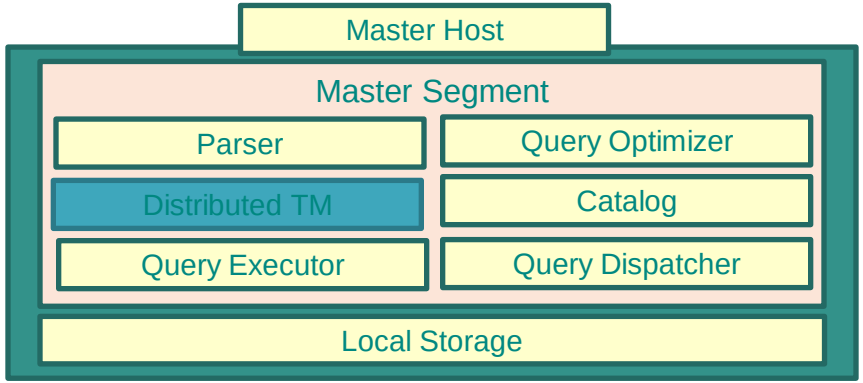


Communicates its intermediate results to other executor processes

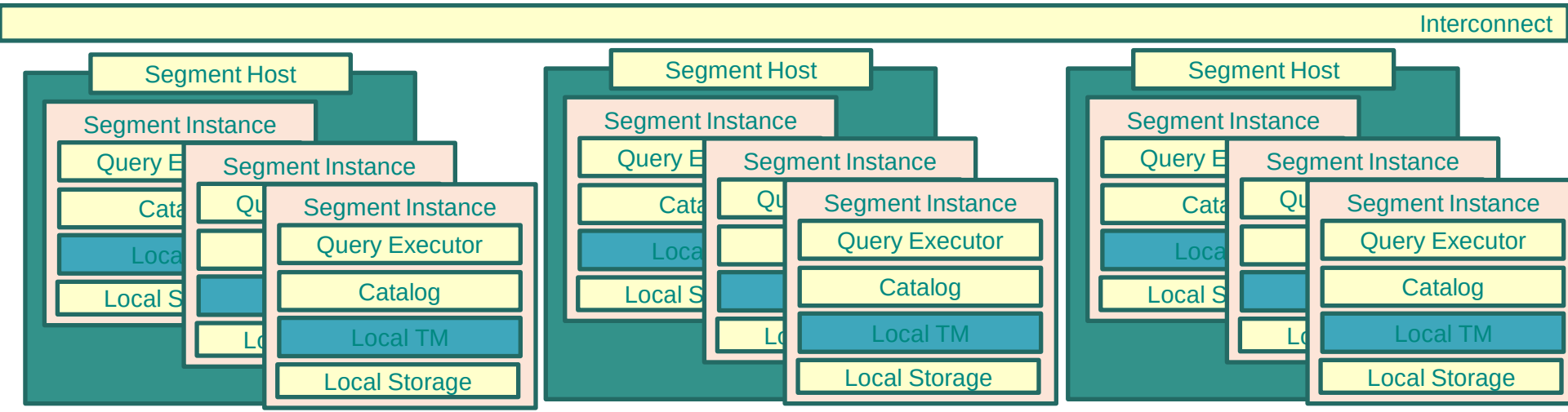


Distributed Transaction Management

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



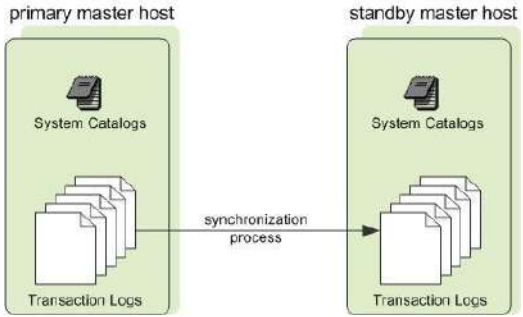
Segments have their own commit and replay logs and decide when to commit, abort for their own transactions



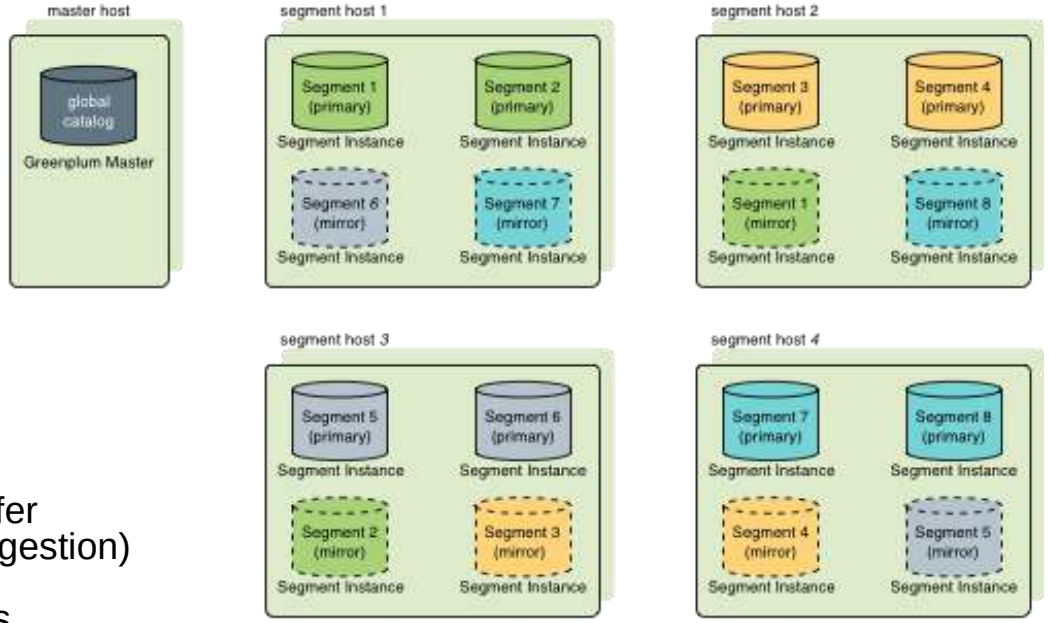
Segment Mirroring

Performant Redundancy

Master View



Segment View



2 copies of each segment

Automatic mirroring

High bandwidth one-to-one data transfer based on file updates (keep up with ingestion)

Automatic failover when hardware fails

Data Distribution

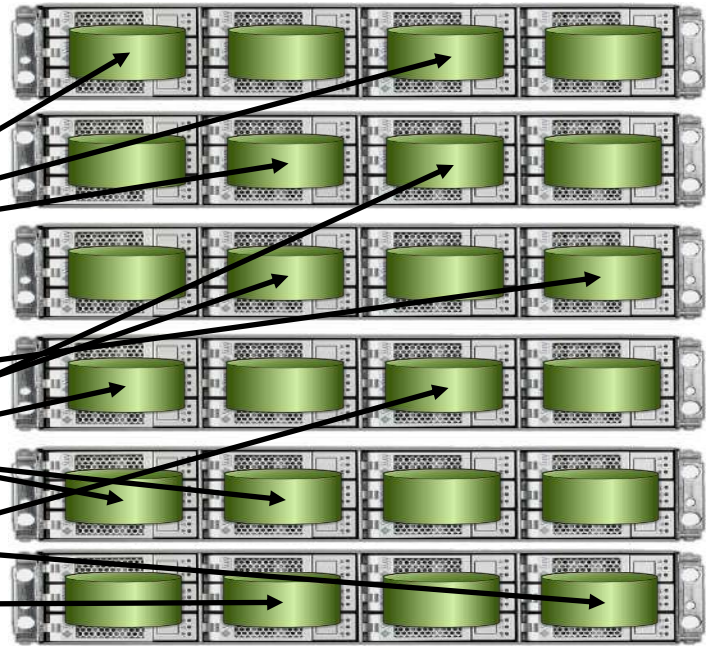
The Key to Parallelism

Example:

SELECT SUM(order_amount) from order;

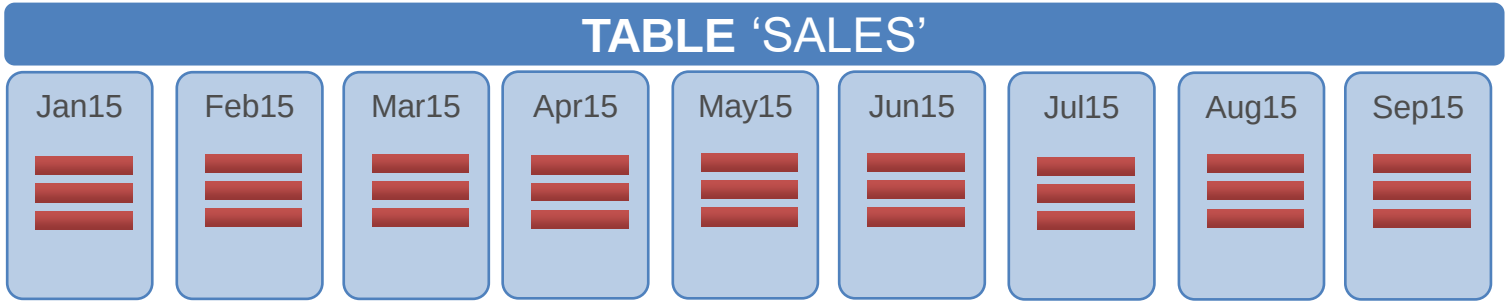
- Data is divided among all hosts
- Can be processed in parallel by queries

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



Vertical Partitioning

Dividing Data By Access Patterns



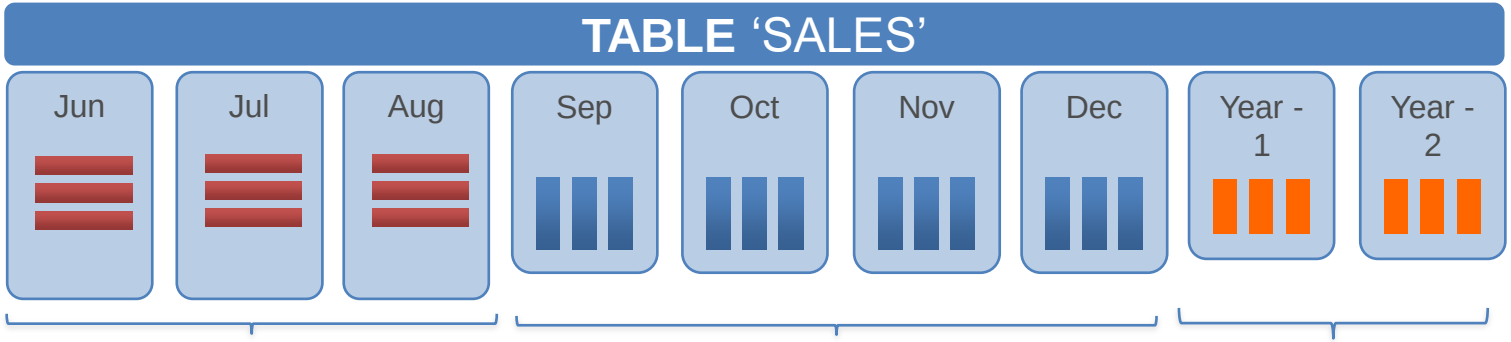
Physical separation of data to enable faster processing with WHERE predicates

Unrequired partitions are not processed

Benefits large fact tables more than small dimension tables

Polymorphic Storage

Logical table with partitioned physical storage



- Row oriented faster when returning all columns
- HEAP for many updates and deletes
- Use indexes for drill through queries

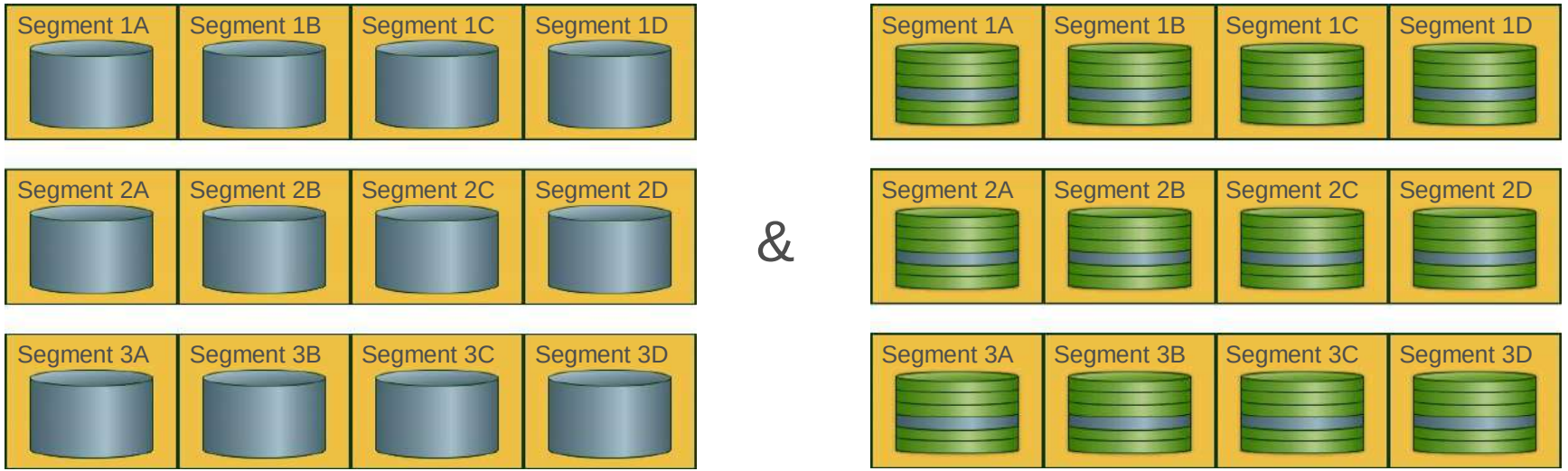
- Columnar storage compresses better
- Optimized for retrieving a subset of the columns when querying
- Compression can be set differently per column: gzip (1-9), quicklz, delta, RLE

- Less accessed partitions on external partitions to seamlessly query all data
- Text, CSV, Binary, Avro, Parquet format
- All major HDP Distros
- S3 Compatible Storage Platforms

Distribution & Partitions

Vertical slices of large fact tables

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



Evenly distribute *orders* data across all segments

Only scans the relevant *order* partitions

Indices

Finding specific items

1 item in millions or billions

Most analytical environments operate on large volumes of data

Sequential scan is the preferred method to read the data

For queries with high selectivity, indexes may improve performance

Drill through queries
Lookup queries

Greenplum Supports Indices:

- Btree
- GIST
- Bitmap
- GIN index (roadmap)
- BRIN index (roadmap)

GPORCA Optimizer

Query Accelerator

8 Years Investment of Doctoral Science for SQL on Big Data

Based on Cascades / Volcano Framework, Goetz Graefe

Handles extremely complex optimizations on big data and MPP clusters

01

Efficiently Processing Complex Correlated Queries

02

Common Table Expression Push Downs

03

Dynamic Partition Elimination

Complex Correlated Queries

SELECT * FROM **part p1**
WHERE **p1.p_size** > 40 OR **p1.p_retailprice** >
(SELECT avg(**p2.p_retailprice**)
FROM **part p2** WHERE **p2.p_brand** = **p1.p_brand**)

GPORCA Decorelates when possible

Avoid Nested Loop

Convert to JOINs

```
                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Gather Motion 3:1  (slice3; segments: 3)  (cost=0.00..60860.79 rows=51199100 width=133)
   ->  Result  (cost=0.00..35484.13 rows=17066367 width=133)
         Filter: public.part.p_size > 40 OR public.part.p_retailprice > (pg_catalog.avg((avg(public.part.p_retailprice))))
         ->  Hash Left Join  (cost=0.00..32115.23 rows=34132734 width=141)
               Hash Cond: public.part.p_brand = public.part.p_brand
               ->  Table Scan on part  (cost=0.00..1961.00 rows=17066367 width=133)
               ->  Hash  (cost=4739.37..4739.37 rows=25 width=19)
                     ->  Broadcast Motion 3:3  (slice2; segments: 3)  (cost=0.00..4739.37 rows=25 width=19)
                           ->  Result  (cost=0.00..4739.36 rows=9 width=19)
                                 ->  HashAggregate  (cost=0.00..4739.36 rows=9 width=19)
                                       Group By: public.part.p_brand
                                       ->  Redistribute Motion 3:3  (slice1; segments: 3)  (cost=0.00..4739.36 rows=9 width=19)
                                             Hash Key: public.part.p_brand
                                             ->  Result  (cost=0.00..4739.36 rows=9 width=19)
                                                   ->  HashAggregate  (cost=0.00..4739.36 rows=9 width=19)
                                                         Group By: public.part.p_brand
                                                         ->  Table Scan on part  (cost=0.00..1961.00 rows=17066367 width=19)
 Optimizer status: PQO version 2.40.0
(18 rows)
```

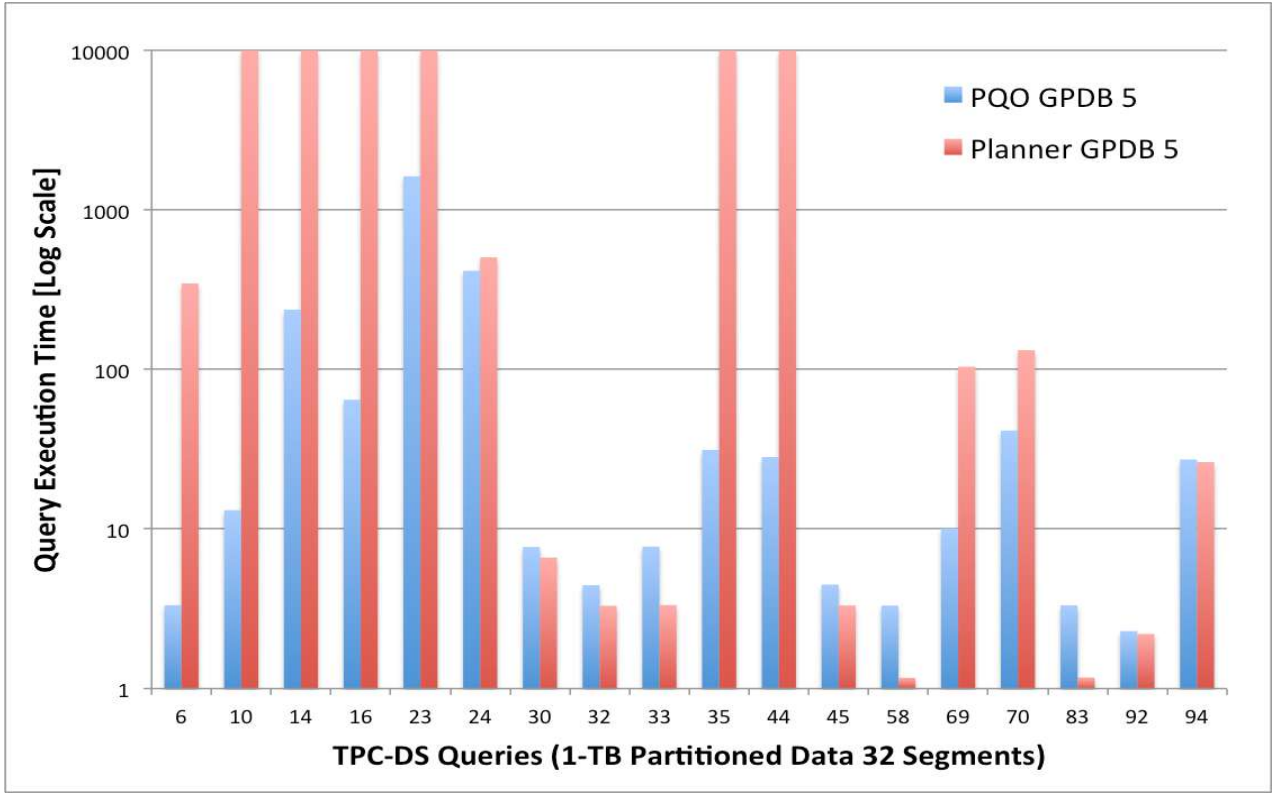
Complex Correlated Queries

GPORCA 100x faster

than PG Based Planner

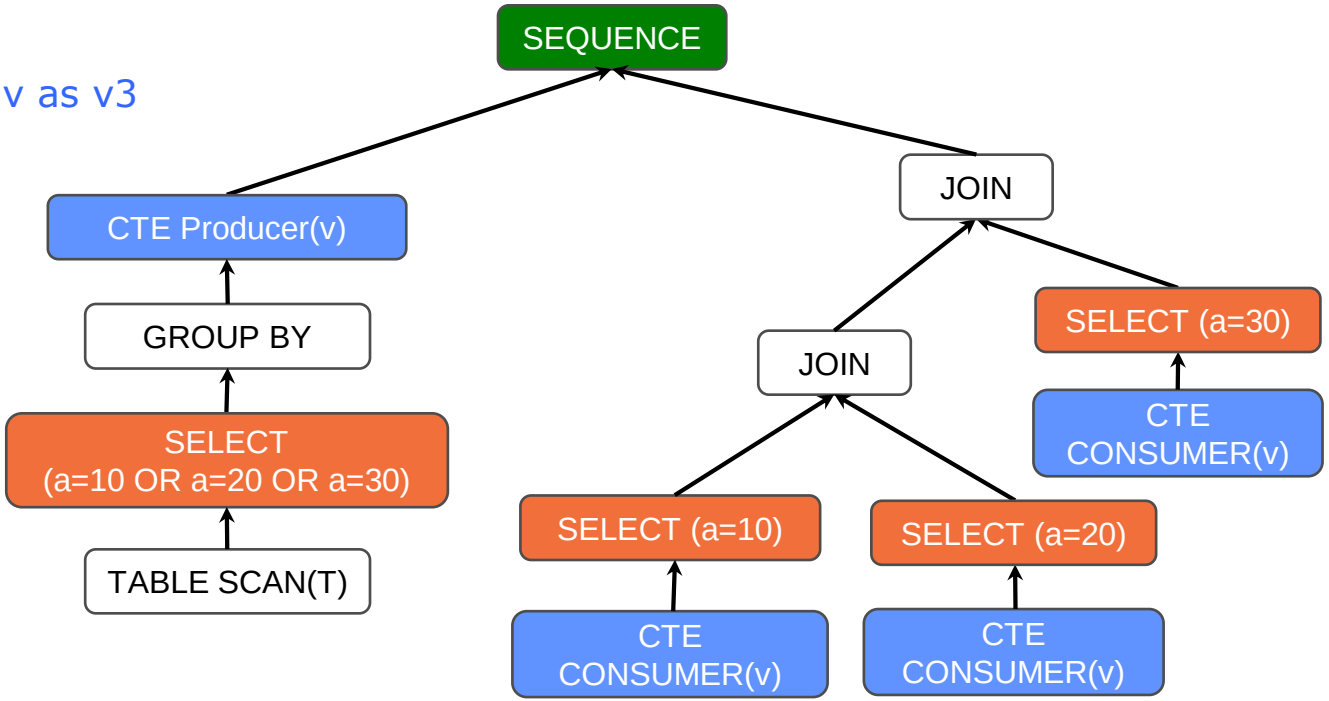
on analytical queries on

large datasets



Pushing Predicates Below CTEs

```
WITH v AS (SELECT a, sum(b) as s FROM T
GROUP BY a)
SELECT *
FROM v as v1, v as v2, v as v3
WHERE v1.a < v2.a
AND v1.s < v3.s
AND v1.a = 10
AND v2.a = 20
AND v3.a = 30;
```



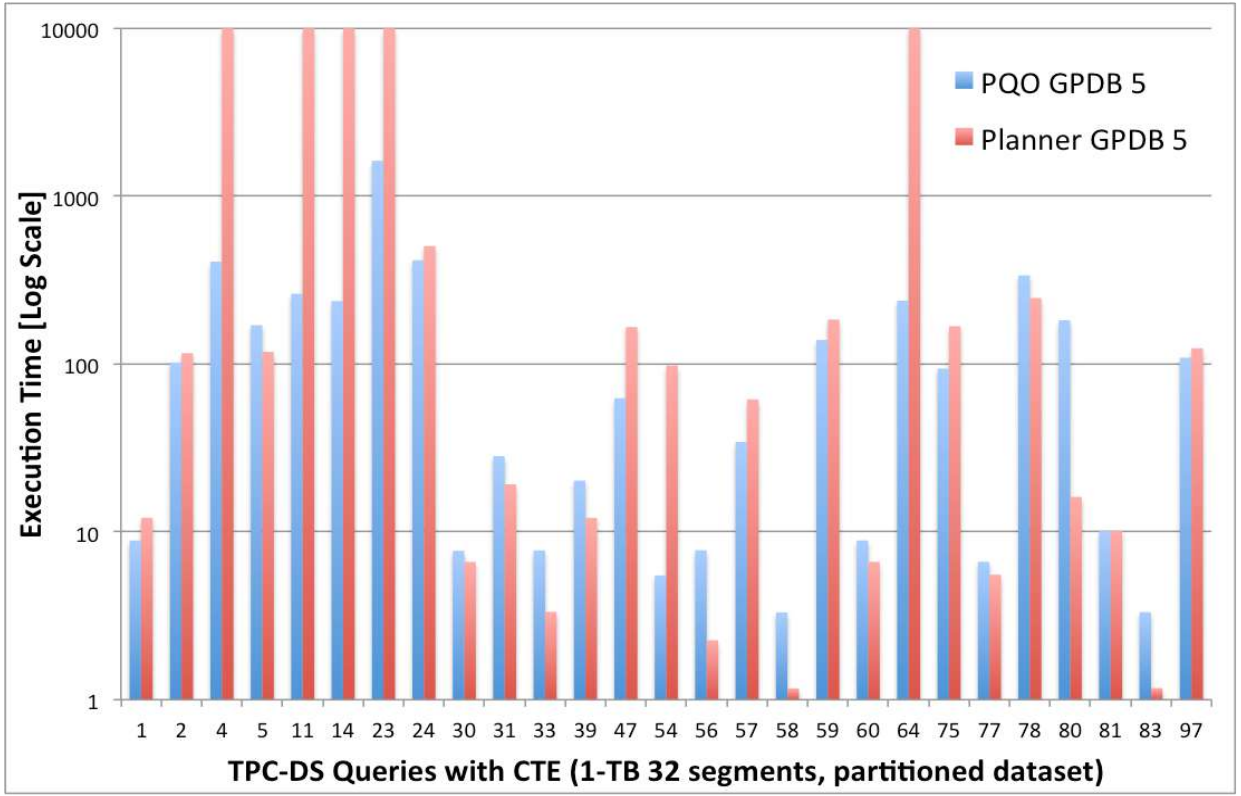
Pushing Predicates Below CTEs

On average

plans generated by

GPORCA 7x faster

than PG Based Planner



Dynamic Partition Elimination

```
bootcamp=# explain SELECT year FROM catalog_sales JOIN date_dim ON (date_id=date_dim.id) GROUP BY year;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Gather Motion 2:1  (slice3; segments: 2)  (cost=0.00..863.06 rows=1 width=4)
   ->  GroupAggregate  (cost=0.00..863.06 rows=1 width=4)
         Group By: date_dim.year
         ->  Sort  (cost=0.00..863.06 rows=1 width=4)
               Sort Key: date_dim.year
               ->  Redistribute Motion 2:2  (slice2; segments: 2)  (cost=0.00..863.06 rows=1 width=4)
                     Hash Key: date_dim.year
                     ->  HashAggregate  (cost=0.00..863.06 rows=1 width=4)
                           Group By: date_dim.year
                           ->  Hash Join  (cost=0.00..863.05 rows=60 width=4)
                                 Hash Cond: catalog_sales.date_id = date_dim.id
                                 ->  Dynamic Table Scan on catalog_sales (dynamic scan id: 1)  (cost=0.00.. rows=5000 width=4)
                                 ->  Hash  (cost=100.00..100.00 rows=50 width=4)
                                       ->  Partition Selector for catalog_sales(dynamic scan id: 1)(cost=10... rows=50 width=4)
                                             Filter: catalog_sales.id = date_dim.id
                                             ->  Broadcast Motion 2:2  (slice1; segments: 2)  (cost=0.00..431.00 rows=12 width=8)
                                                   ->  Table Scan on date_dim  (cost=0.00..431.00 rows=6 width=8)
 Settings:  optimizer=on
 Optimizer status: PQO version 2.40.0
(19 rows)
```

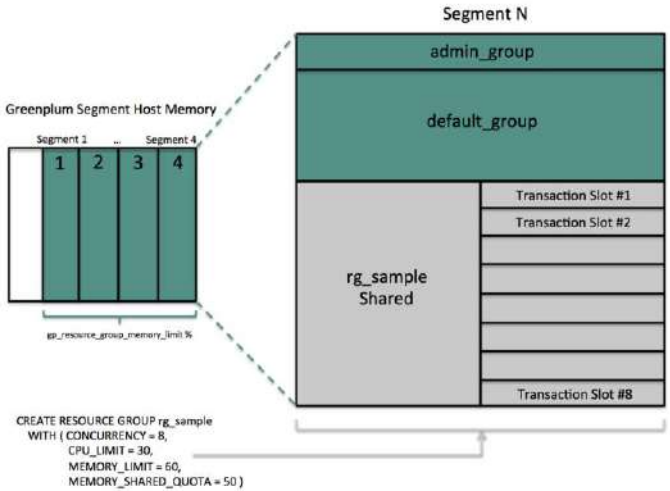
SQL Containerization: Greenplum Resource Groups

GOALS

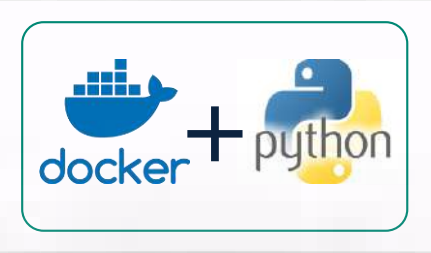
- Provides resource isolation for query multi-tenancy and mixed workloads
- Enhances stability and manageability of Greenplum

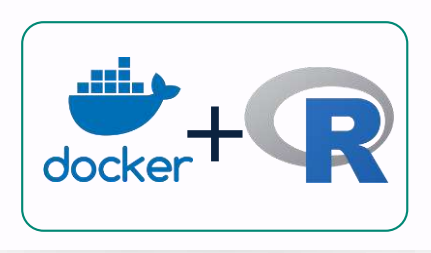
CAPABILITIES

- Specify CPU Max Per Group
- Burst Above Max Limit if available
- Specify Max Memory Per Group And Memory Per Query
- Specify Max Concurrency Per Group
- Leverages Linux Cgroups for implementation
- Able to pin workload to CPU cores
- Transaction scope not Statement scope



Containerized Compute Environments





Key Features

- Foundational work for containerized Python and R compute environments
- Brings trusted execution of Python and R inside Greenplum, as well as Anaconda Python and Python 2.7
- Uses Docker Containers for sandboxing the execution environment for user functions, preventing the user from harming the host system and accessing the things end user should not access

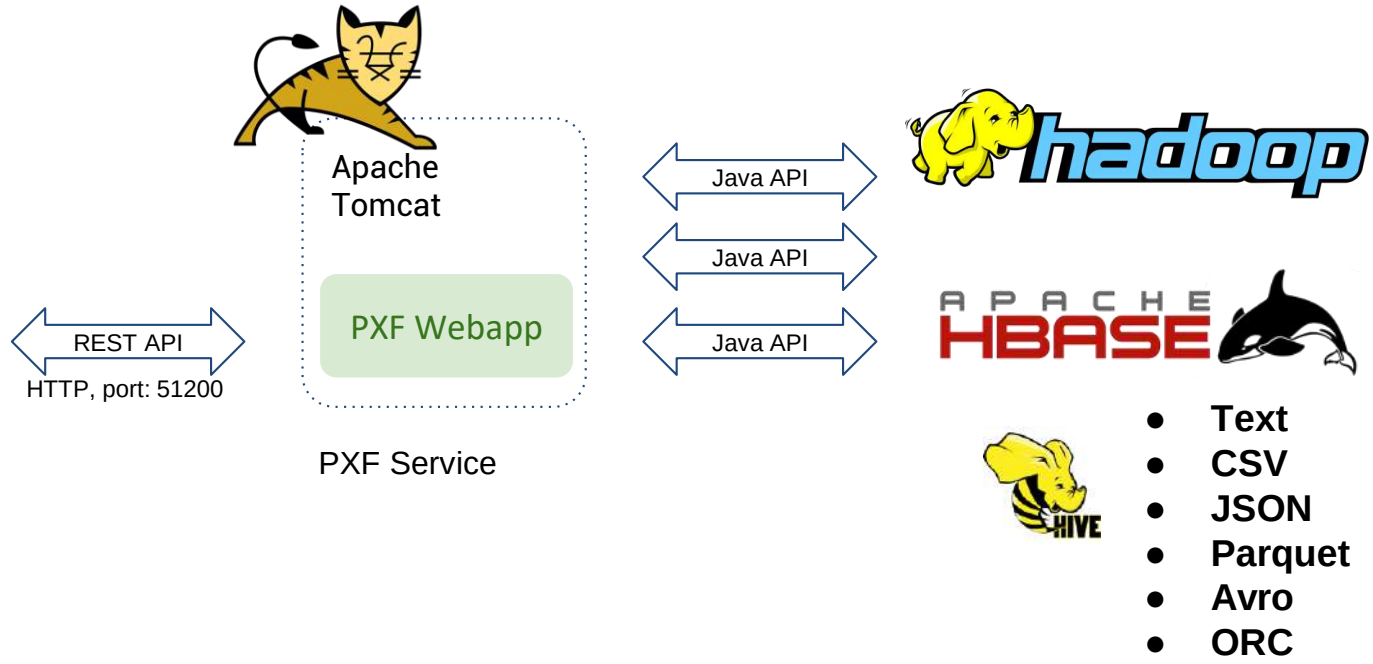
New Greenplum Backup & Restore Utility

- Released GA in February 2018
- Improved Locking Profile
- Same Locks as Read-Only User

- Enhanced monitoring and reporting
- Plugins Architecture
- MPP pg_dump

PXF: Accelerated Hadoop Access

Unlock external data source with power of Greenplum Query



Apache Madlib Advanced Analytics Library

Key Features

- Open-source library for scalable in-database analytics; provides data-parallel implementations of mathematical, statistical and machine learning methods for structured and unstructured data
- Apache Top Level Project from July 2017

Graph Analytics

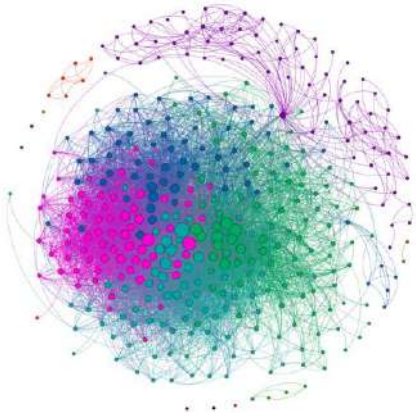
Natural Phenomena Have Graph Data Structure

Example: Social Network, Computer Network, Industrial Components, etc.

Familiar SQL interface

Algorithms:

- All Pairs Shortest Path (APSP)
- Breadth-First Search
- Average Path Length
- Closeness Centrality
- Graph Diameter
- In-Out Degree
- PageRank
- Single Source Shortest Path (SSSP)
- Weakly Connected Components



```
SELECT madlib.pagerank(
                        'vertex',              -- Vertex table
                        'id',                  -- Vertix id column
                        'edge',                -- Edge table
                        'src=src, dest=dest',  -- Comma delimted string of edge arguments
                        'pagerank_out',        -- Output table of PageRank
                        NULL,                  -- Default damping factor (0.85)
                        NULL,                  -- Default max iters (100)
                        0.00000001,            -- Threshold
                        'user_id');            -- Grouping column name
```

Vertex Table

Vertex	Vertex Params
0	...
1	...
2	...
3	...

Edge Table

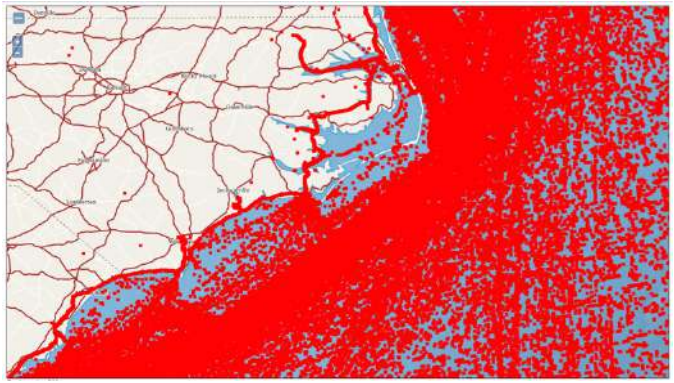
Source Vertex	Dest Vertex	Edge Weight	Edge Params
0	3	1.0	...
1	0	5.0	...
1	2	3.0	...
2	3	8.0	...
3	0	3.0	...
3	1	2.0	...

PostGIS @ Scale

Key Features

- Supports for Spatial objects/types/functions such as Points, Lines, Polygons, Perimeter, Area, Intersection, Contains, Distance, Longitude/Latitude
- Raster support
- Round Earth calculations
- Spatial Indexes & Bounding Boxes

- For example, the query for all ship traffic of the coast of North Carolina looked like this: SELECT * FROM <table> WHERE <geom> && ST_MakeEnvelope(-78, 33, -75, 36, 4326);



Pivotal Greenplum v6 (targeted March 2019)

- Merge PostgreSQL 9.3 or 9.4 into GPDB
    - Column Level Permissions, Recursive CTE, GIN Index Support, Unlogged Tables, Range Types, higher speed short queries, more
- Safe In Place Major Upgrades
- Write Ahead Logging (WAL) for internal cluster mirroring
- Online Expand with Jump Consistent Hash
- Replicated Tables
- Distributed Deadlock Detection

Runs In All Platforms

- Infrastructure Agnostic: A portable, 100% software solution
- Same platform, no switching/migration cost

Greenplum Database Vision

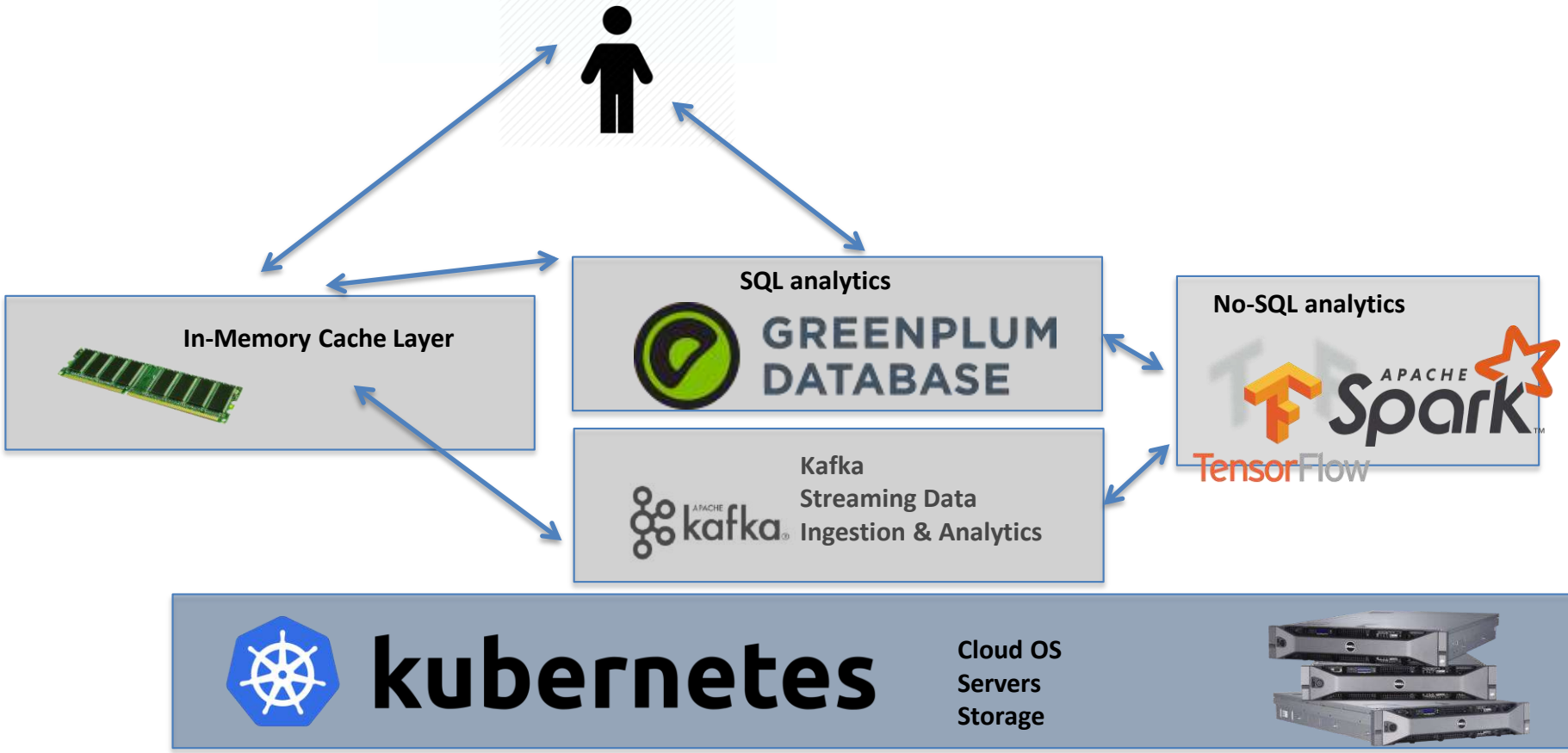
PostgreSQL as industry standard OSS RDBMS core engine

Elastic Flexible MPP Deployments

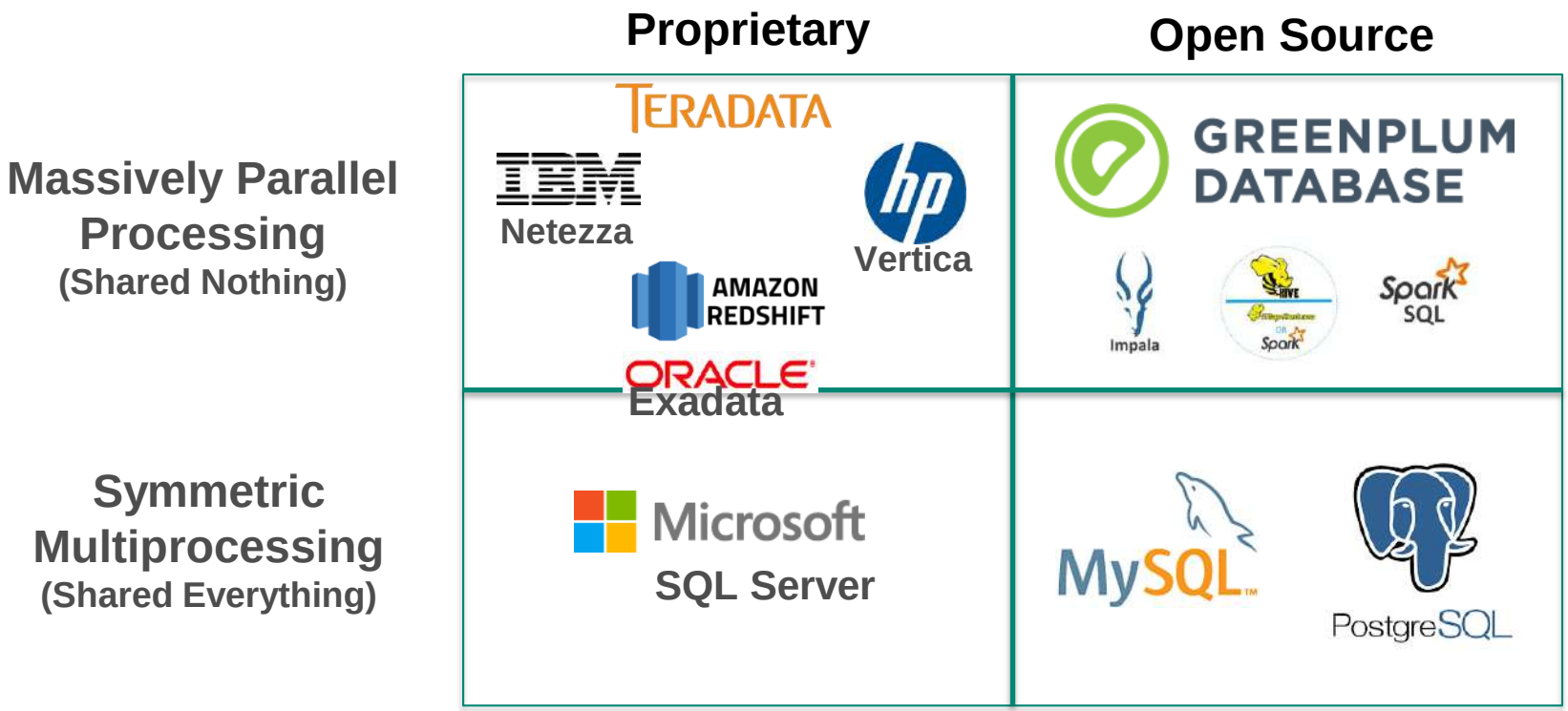
Mixed Workload, High Concurrency, Mission Critical Use Cases

Open Source EcoSystem Integration, Avoid Data Silos

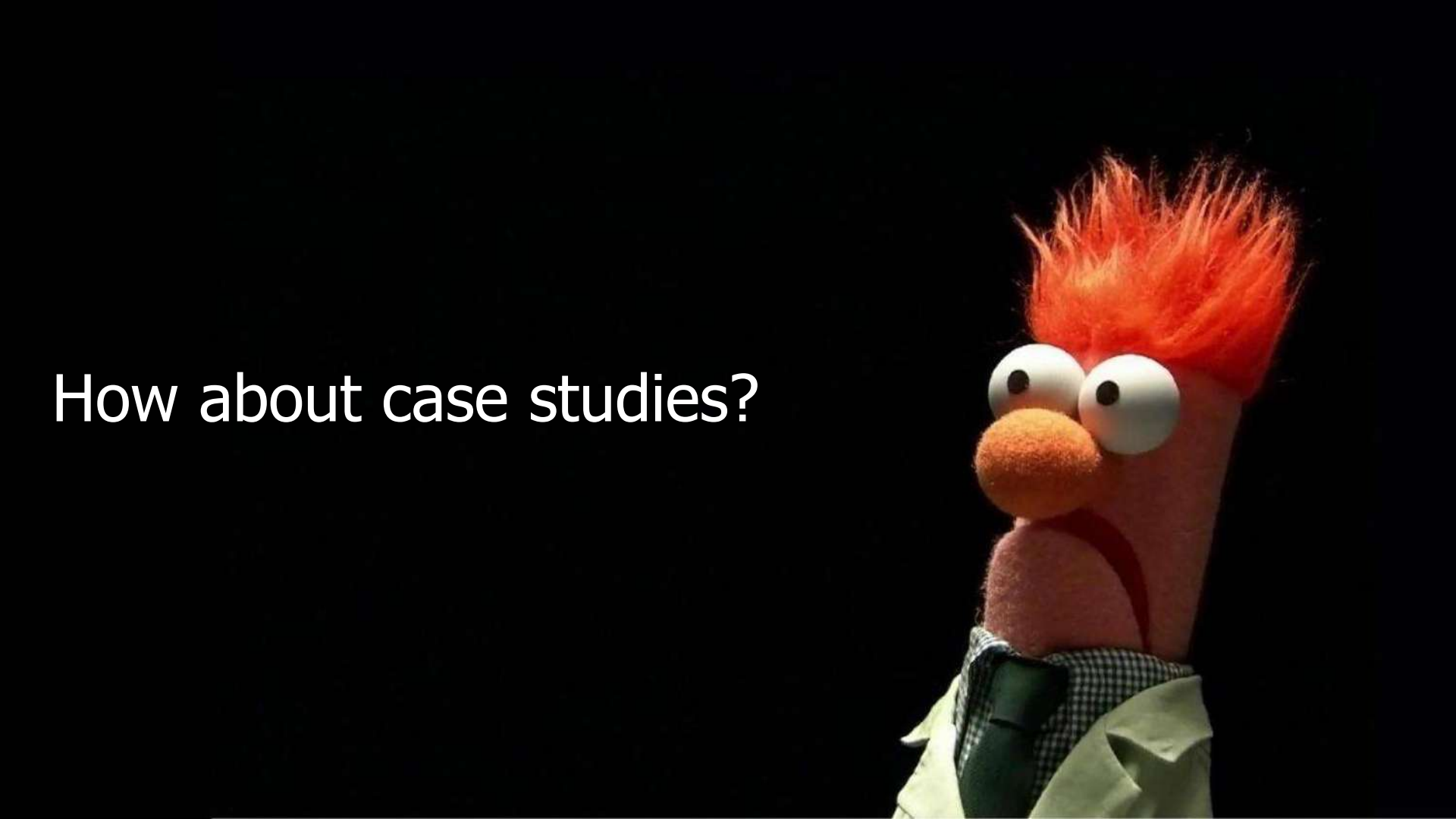
Data Analytics Architecture of Future



Big SQL Competition



How about case studies?



Wall Street Risk Calculations: Crush Your Deadline

A modern MPP architecture enables rapid development and processes information on-demand.

- Millions and Billions of Risk Calculations Can be Stored and Queried
- Daily reports can be generated in under an hour
- Global Stress Tests can be run daily not weekly
- Run New AdHoc Reports Based on Spontaneous Ideas
- Chief Risk Officer: “Without Greenplum We Could Not Have Achieved These Results”

Anomalous Data Movement Use Case

Protect the integrity of internal operations

- Firm needs to consolidate activity from system access logs of all types
- Firm needs to audit internal system usage
- Ability to correlate and join data sources not just act on events
- Determine the difference between normal and abnormal behavior
- Learn over time based on incidents and false positive training
- Detect internal abuse of systems or access
- Detect Advanced Persistent Threats

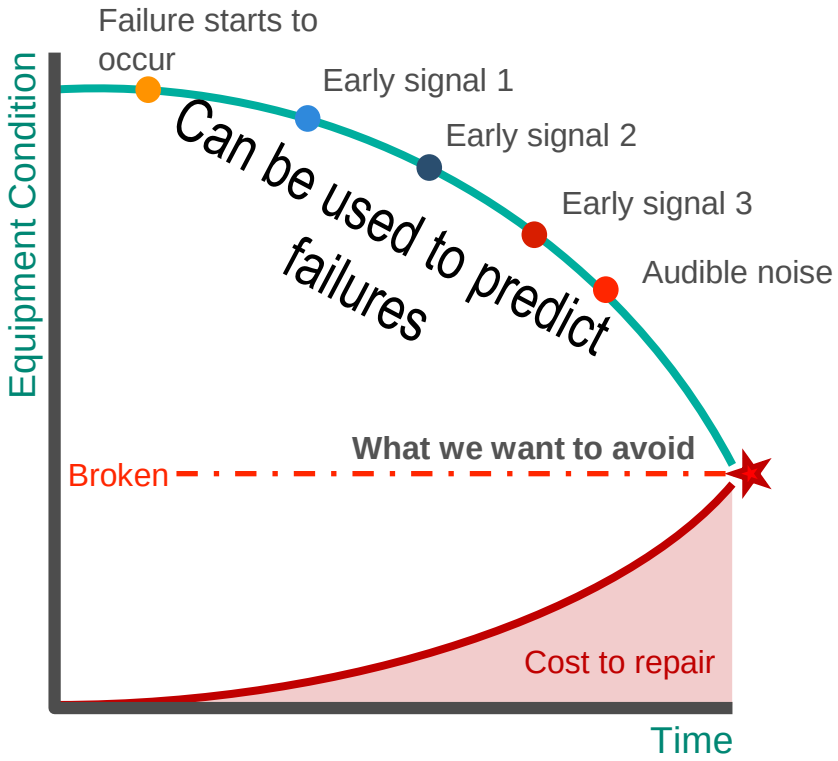
Predictive Maintenance Analytics

Goal

- Failing equipment causes issues with operations
- Unable to store & process fire-hose of data
- Start maintenance before equipment will fail
- Avoid costly un-required activity

Solution

- **High velocity** data ingestion
- Store PBs of data
- Machine learning and SQL analytics
- Very **low latency** and **high speed** data access

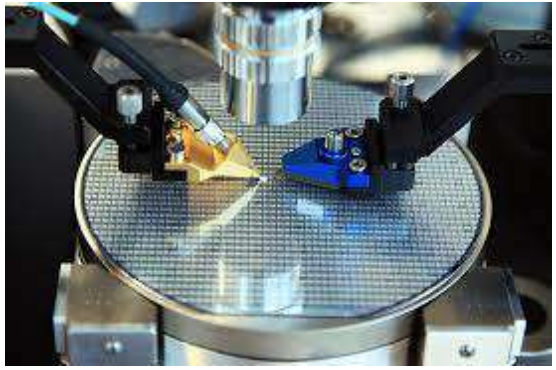


…and other use cases…

DEMAND FORECASTING

IOT REPORTING

YIELD ANALYTICS



CHURN REDUCTION

How can I get involved?

Greenplum Community Update

Open Source BootStrap from Zero Oct 2015

- Github is cool!

https://github.com/greenplum-db/gpdb

  - 392 Project Watchers
  - 2549 Project Stars
  - 782 Project Forks
  - 170 Contributors
  - 4433 PRs (51 open)
  - 605 issues (160 open)

Greenplum Mailing Lists

- 357 gpdb-users@greenplum.org subscribers
- 287 gpdb-dev@greenplum.org subscribers

Greenplum Slack Channel

- 183 https://greenplumslack.herokuapp.com/ members

Greenplum YouTube Channel

- 762 https://www.youtube.com/greenplumdatabase subscribers
- 101 Videos

PGConf Brasil 2018
Greenplum Database: Evolving Advanced Analytics on PostgreSQL