ORCA

Query optimization as a service

@addisonhuddy

20 min

- Little about query optimization
- Introduce ORCA
- ORCA Internals
- Pairing: adding a transformation
- ORCA Roadmap

Why care about query optimization?

- Turns queries (SQL, MapReduce, ...) into an execution plan
- Data growth > Processing growth
- So many optimizers!

Why care about ORCA?

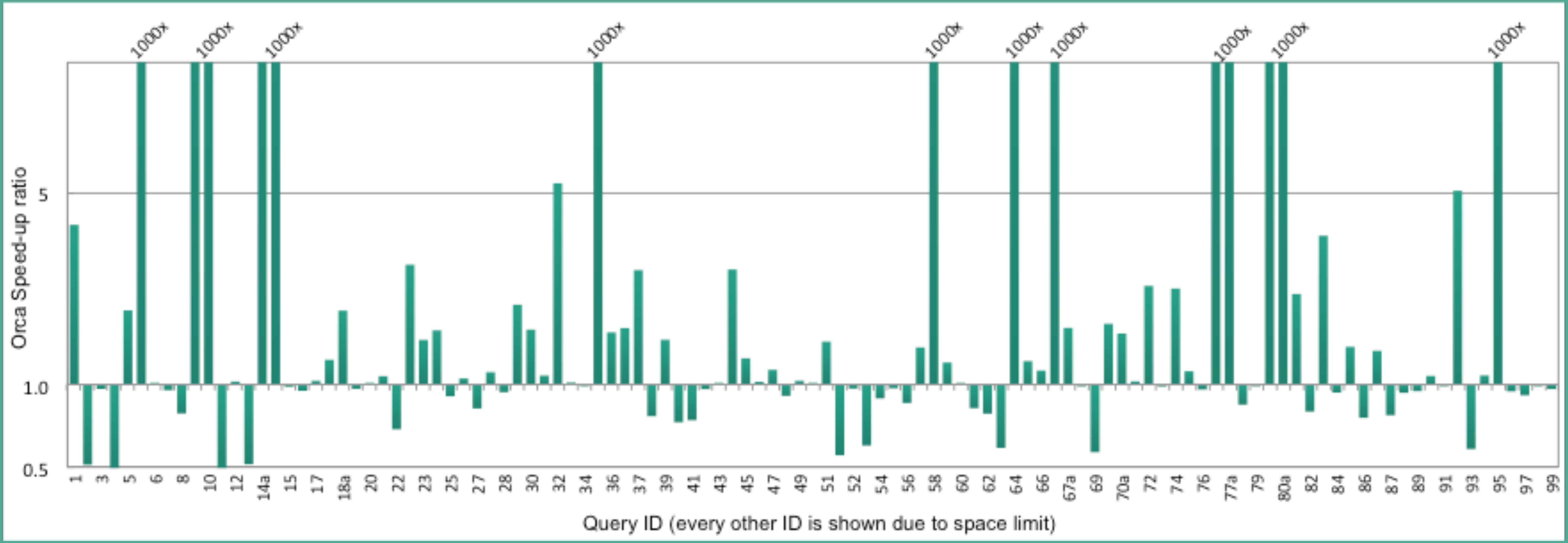
— Modular
— Extensible
— Plugable

=> R&D test-bed

HAWQ

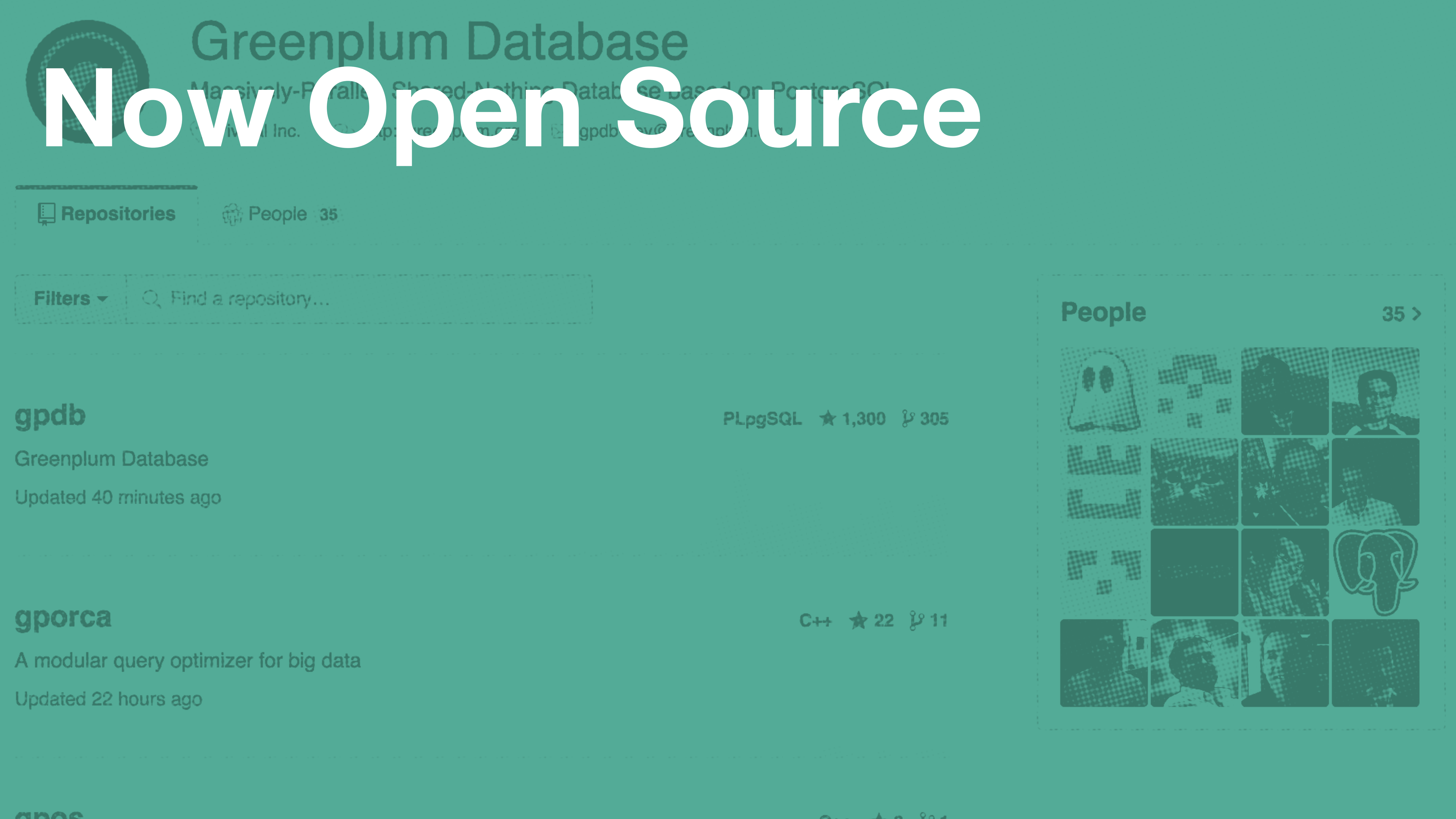
GREENPLUM DATABASE

TPC-DS 5X Faster[1]



[1] © 2014 ACM, used with permission. Figure 3. TCP-DS performance testing results of Pivotal Greenplum with Pivotal Query Optimizer vs. Pivotal Greenplum with "planner" query optimizer.

Now Open Source



Greenplum Database

Massively-Parallel Shared-Nothing Database based on PostgreSQL

Pivotal Inc. http://greenplum.org gpdb-dev@greenplum.org

Repositories People 35

Filters Find a repository…

gpdb PLpgSQL ★ 1,300 305

Greenplum Database

Updated 40 minutes ago

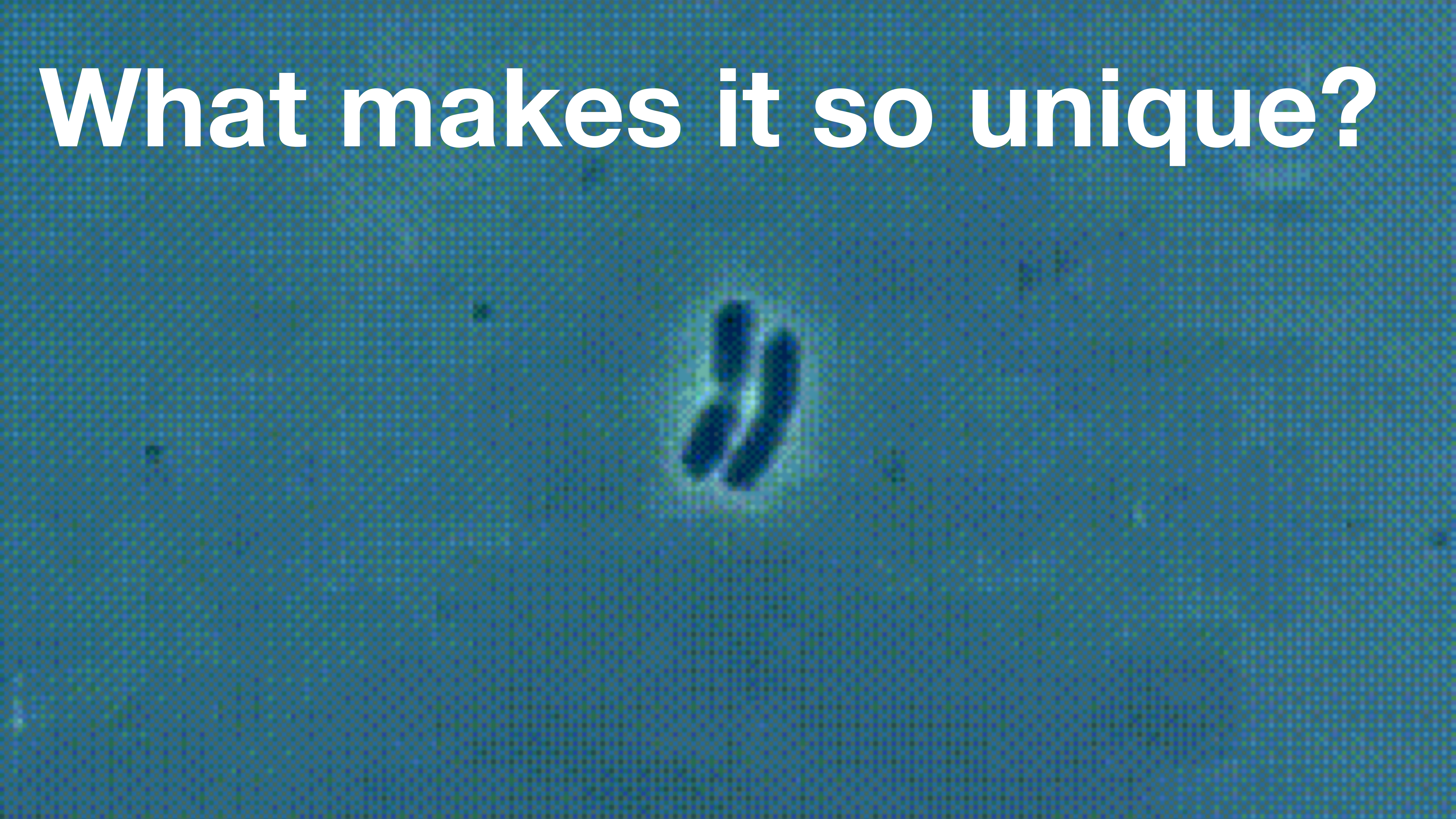
gporca C++ ★ 22 11

A modular query optimizer for big data

Updated 22 hours ago

People 35

What makes it so unique?



Key Features

- Smarter partition elimination
- Subquery unnesting
- Common table expressions (CTE)
- Join ordering
- Sort order optimization
- Skew awareness

Logical & || Physical Operators

Pre-processing -> Exploration -> Implementation -> Optimization

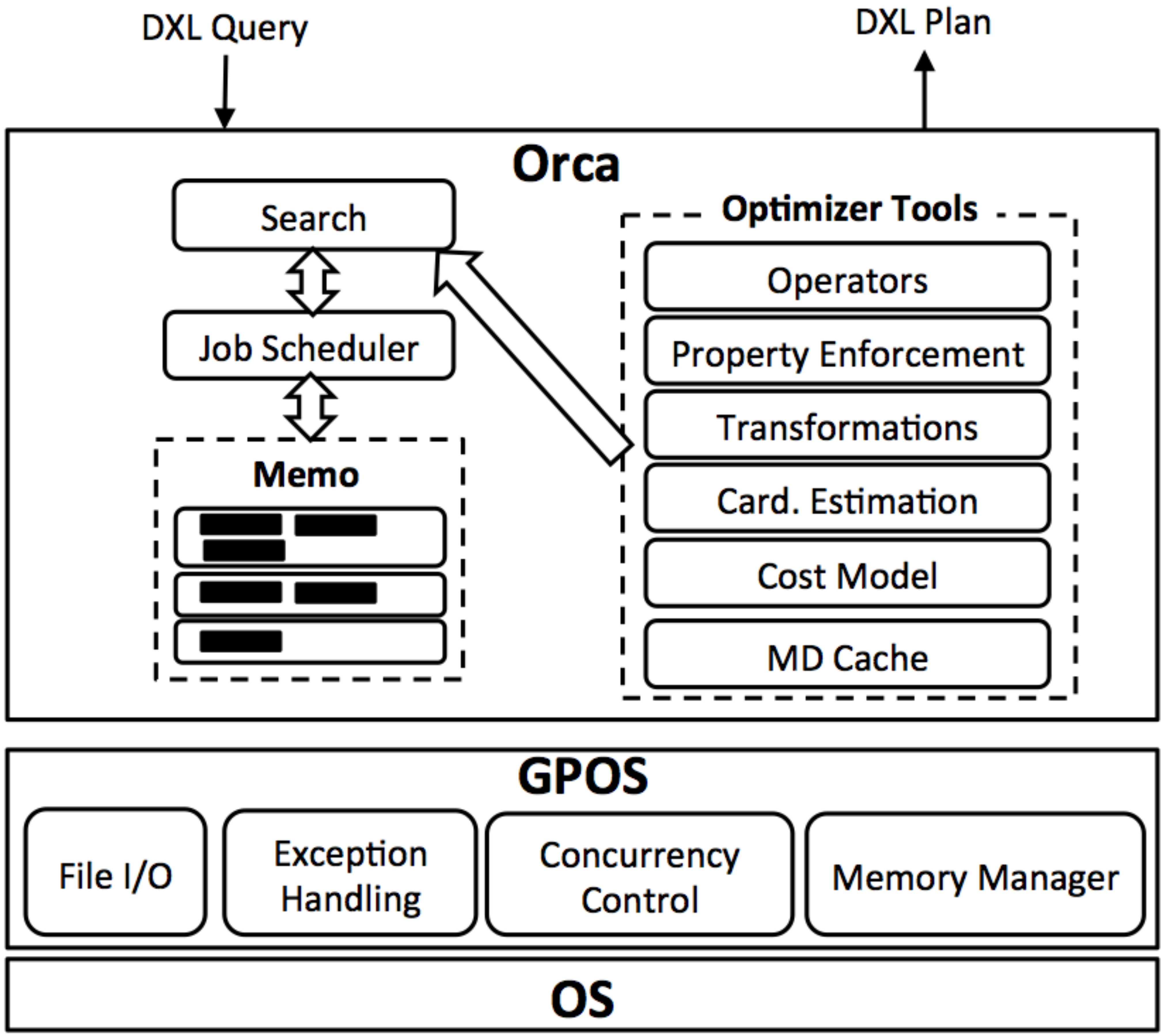
ALL possible logical plans are turned into its physical operators
- Dr. Venkatesh "Venky" Raghavan

Many Logical transformations

- Join Ordering Algorithm
- Expand NAry Join Min Card Algorithm
- Expand NAry Join Dynamic Programming
- Select to Filter
- Select to IndexGet
- Simplify Select With Subquery

+77 more from what I could count on the plane

1,000,000,000


DXL Query
DXL Plan
Orca
Search
Job Scheduler
Memo
Optimizer Tools
Operators
Property Enforcement
Transformations
Card. Estimation
Cost Model
MD Cache
GPOS
File I/O
Exception Handling
Concurrency Control
Memory Manager
OS

Let's Pair

Idea

Split an aggregate into a pair of local and global aggregate.

```
SELECT sum(c) FROM foo GROUP BY b

CREATE TABLE foo (a int, b int, c int) distributed by (a);
```

```
CXformSplitGbAgg

// HEADER FILES
~/orca/libgpopt/include/gpopt/xforms

// SOURCE FILES
~/orca/libgpopt/src/xforms
```

Pattern

```
GPOS_NEW(pmp)
CExpression
(
    pmp,
    GPOS_NEW(pmp) CLogicalGbAgg(pmp),
  // logical aggregate operator
    GPOS_NEW(pmp) CExpression(pmp, GPOS_NEW(pmp) CPatternLeaf(pmp)),
  // relational child
    GPOS_NEW(pmp) CExpression(pmp, GPOS_NEW(pmp) CPatternTree(pmp))
  // scalar project list
    ));
```

What?

```
...
->  Redistribute Motion 2:2  (slice1; segments: 2)  (cost=0.00..431.00 rows=1 width=12)
            Hash Key: b
            Rows out:  Avg 1.5 rows x 2 workers at destination.  Max 2 rows (seg0) with 1.105 ms to end, start offset by 15 ms.
            ->  Result  (cost=0.00..431.00 rows=1 width=12)
                        Rows out:  Avg 1.5 rows x 2 workers.  Max 2 rows (seg1) with 0.273 ms to first row, 0.276 ms to end, start offset by 16 ms.
                        ->  GroupAggregate  (cost=0.00..431.00 rows=1 width=12)
                                    Group By: b
                                    Rows out:  Avg 1.5 rows x 2 workers.  Max 2 rows (seg1) with 0.272 ms to first row, 0.275 ms to end, start offset by 16 ms.
                                    ->  Sort  (cost=0.00..431.00 rows=1 width=8)
                                                Sort Key: b
                                                Rows out:  Avg 1.5 rows x 2 workers.  Max 2 rows (seg1) with 0.260 ms to first row, 0.262 ms to end, start offset by 16 ms.
                                                Executor memory:  58K bytes avg, 58K bytes max (seg0).
                                                Work_mem used:  58K bytes avg, 58K bytes max (seg0). Workfile: (0 spilling, 0 reused)
                                                ->  Table Scan on foo2  (cost=0.00..431.00 rows=1 width=8)
                                                            Rows out:  Avg 1.5 rows x 2 workers.  Max 2 rows (seg1) with 0.074 ms to first row, 0.075 ms to end, start offset by 16 ms.
```

Pre-condition Check

```
// Compatibility function for splitting aggregates
virtual
BOOL FCompatible(CXform::EXformId exfid){
    return (CXform::ExfSplitGbAgg != exfid);}
```

The Actual Transformation

```
void Transform
    (
    CXformContext *pxfctxt,
    CXformResult *pxfres,
    CExpression *pexpr
    ) const;
```

Register Transformation

```
void CXformFactory::Instantiate()
{
….
Add(GPOS_NEW(m_pmp) CXformSplitGbAgg(m_pmp));
….
}
```

What can't it do?

Balance

Improved performance for short running queries

Not yet feature complete

- External parameters
- Cubes
- Multiple grouping sets
- Inverse distribution functions
- Ordered aggregates
- Indexed expressions

PostgreSQL

Four pieces of low hanging fruit

1. Distinguish between Physical and Logical
2. Move expression evaluation inside ORCA
3. ORCA assumes indexes are all forward access
4. Constraint evaluation from a `NOT NULL`

Get Involved

github.com/greenplum-db

gpdb-dev@greenplum.org

Pivotal Tracker: bit.ly/1m1WGDn

White Paper: bit.ly/1ntrE8v

@addisonhuddy

Slides

Slides: github.com/addisonhuddy/presentations