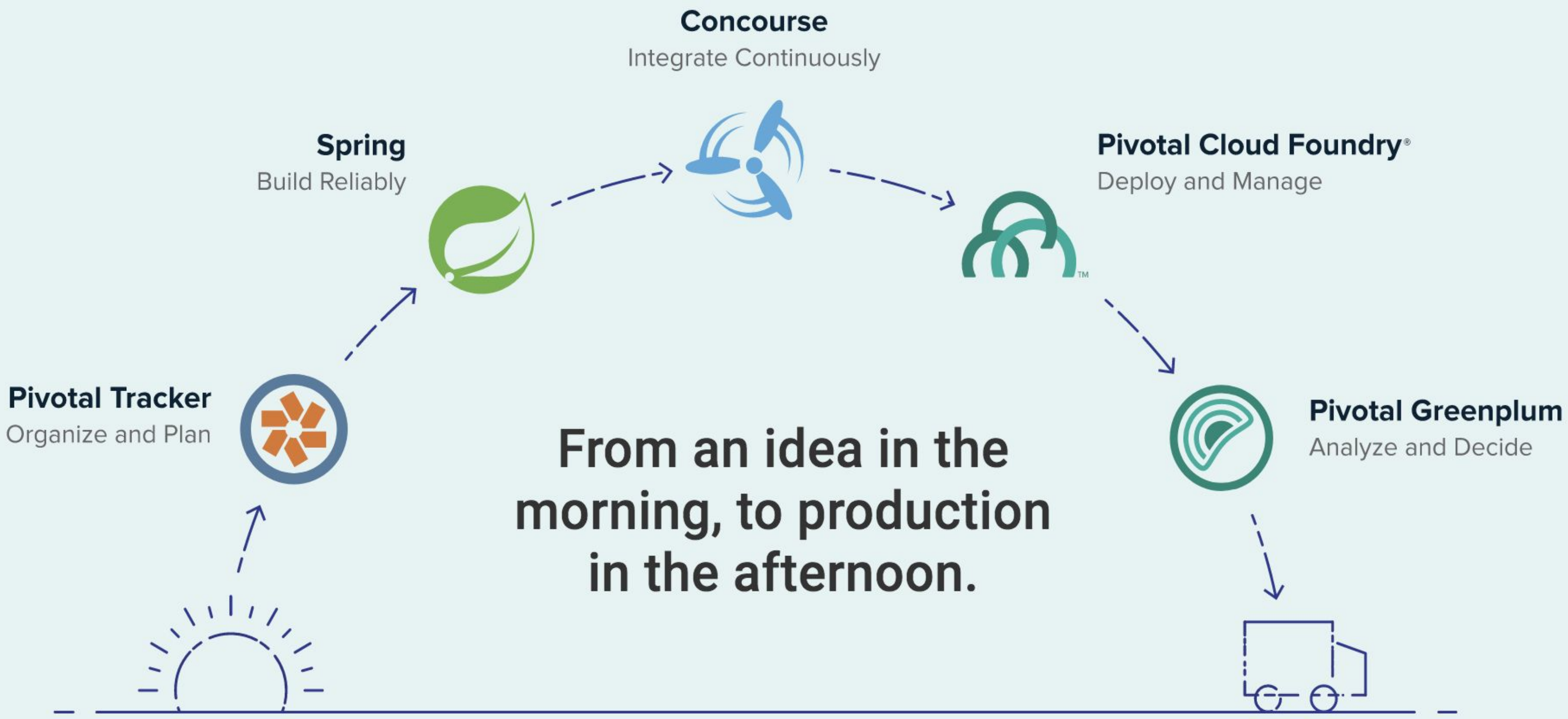
Pivotal

The Internals of GPORCA Optimizer

Xin Zhang (xzhang@pivotal.io)

PGConf Seattle 2017
Nov 2017

Disclaimer



From an idea in the morning, to production in the afternoon.

Open Source Software (OSS)

Oct 2015, Greenplum Database Open Sourced

(based on **PostgreSQL 8.2**)

Sep 7th 2017, Greenplum Database **5.0** w/ GPORCA

(based on **PostgreSQL 8.3**)

Oct 23rd 2017, Greenplum Database **6.0 Alpha**

(based on **PostgreSQL 8.4**)

...

...

Sep 2015, Shin joined Pivotal

Jan 2016, GPORCA Open Source V1.636
http://engineering.pivotal.io/post/gporca-open-source/

Jan 2017, GPORCA V2.0 (merge GPOS)

Sep 6th 2017, GPORCA V2.42.0

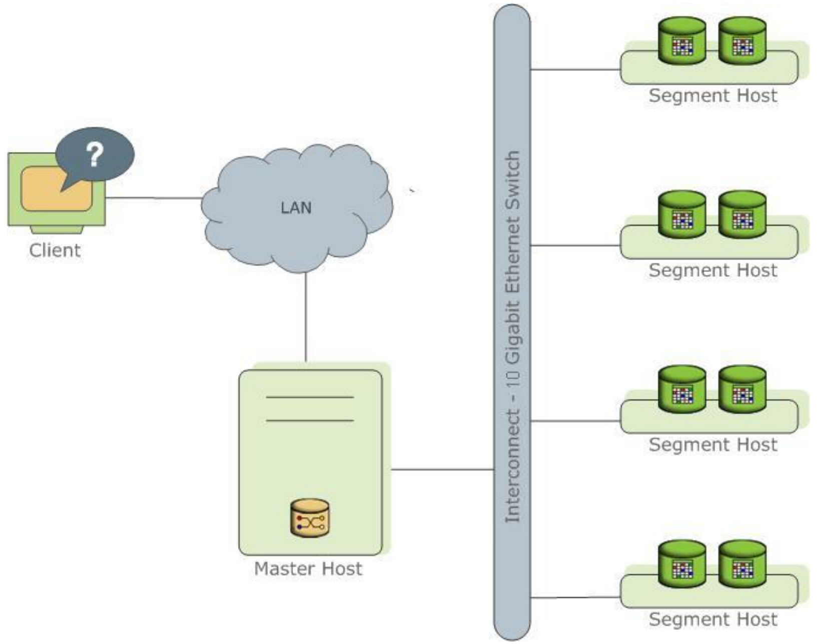
Nov 8th 2017, GPORCA V2.48.6

Greenplum (GPDB): MPP + PostgreSQL

- Shared Nothing Architecture
- Data Distributed on Cluster
- Query Processed Locally in Parallel
- Each Segment is a PostgreSQL Instance

GREENPLUM DATABASE

http://greenplum.org/



MPP Query Processing Example 1/3

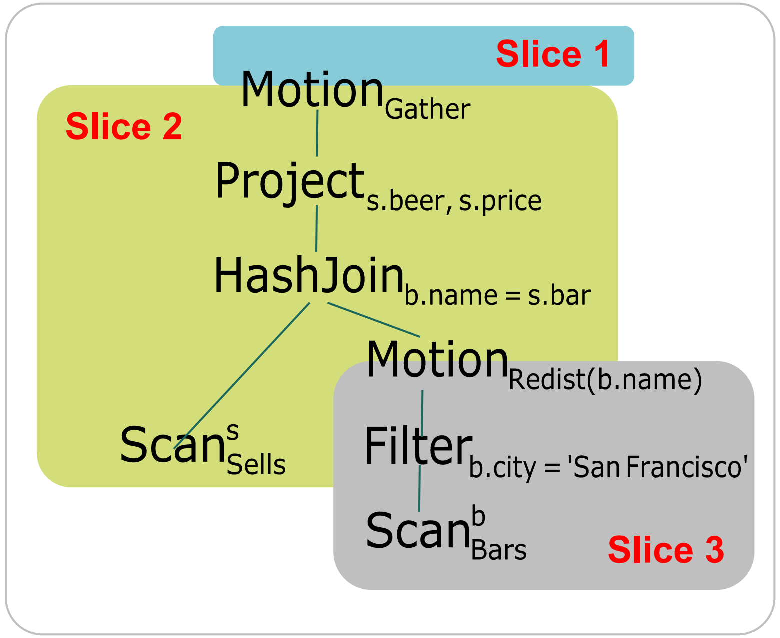
```
SELECT s.beer, s.price
FROM Bars b, Sells s
WHERE b.name = s.bar
AND b.city = 'San Francisco'
```

- `Bars` is distributed randomly ➔ any `bar` on any host
- `Sells` is distributed by `bar` ➔ `sells` of same `bar` on same host

MPP Query Processing Example 2/3

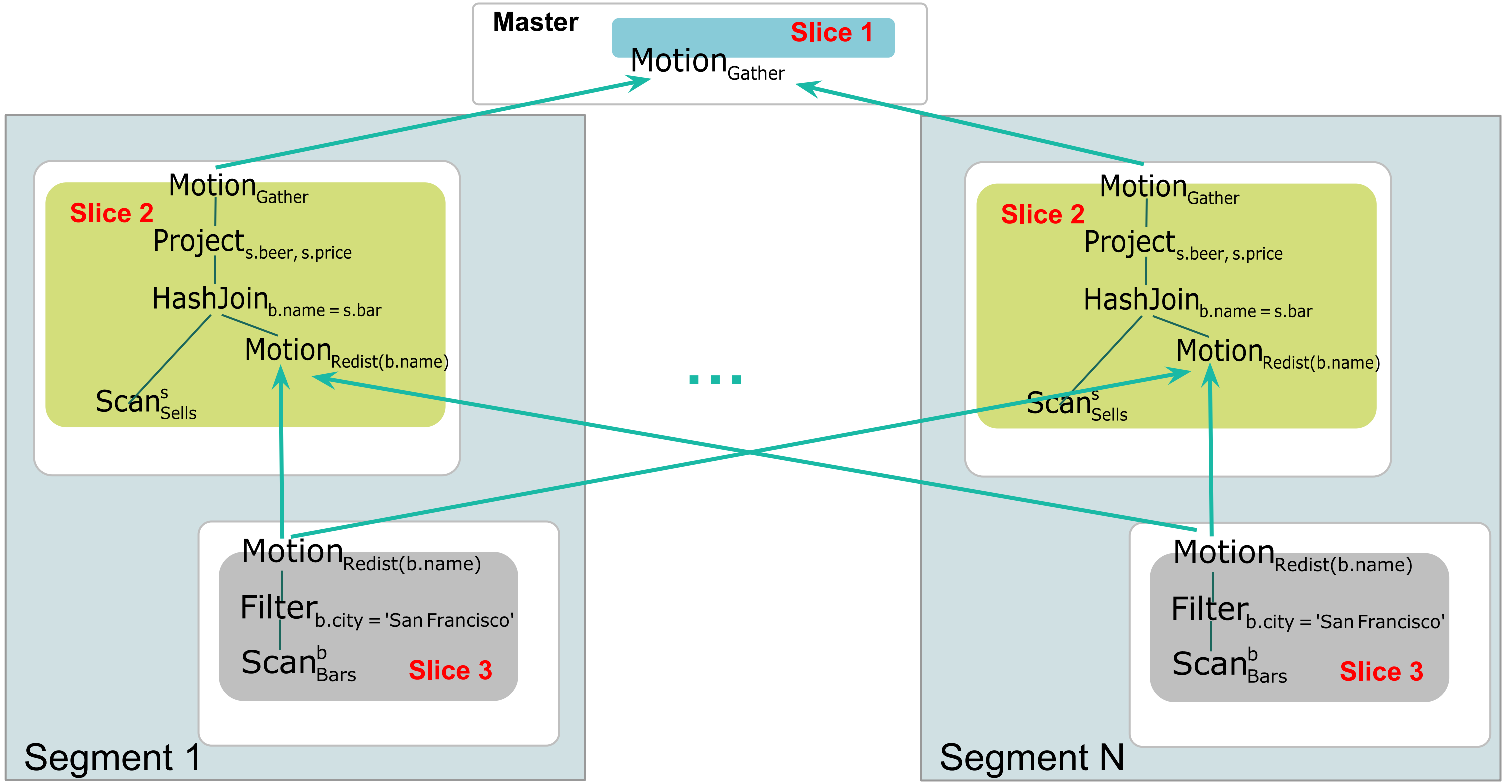
```
SELECT s.beer, s.price
FROM Bars b, Sells s
WHERE b.name = s.bar
AND b.city = 'San Francisco'
```

- `Bars` is distributed randomly
- `Sells` is distributed by `bar`



Slice 1
Slice 2
Slice 3
$\text{Motion}_{\text{Gather}}$
$\text{Project}_{\text{s.beer, s.price}}$
$\text{HashJoin}_{\text{b.name = s.bar}}$
$\text{Motion}_{\text{Redist(b.name)}}$
$\text{Scan}^{s}_{\text{Sells}}$
$\text{Filter}_{\text{b.city = 'San Francisco'}}$
$\text{Scan}^{b}_{\text{Bars}}$

MPP Query Processing Example 3/3



Master
Slice 1
Motion$_{Gather}$
Segment 1
Motion$_{Gather}$
Slice 2
Project$_{s.beer, s.price}$
HashJoin$_{b.name = s.bar}$
Motion$_{Redist(b.name)}$
Scan$^{s}_{Sells}$
Motion$_{Redist(b.name)}$
Filter$_{b.city = 'San Francisco'}$
Scan$^{b}_{Bars}$
Slice 3
...
Segment N
Motion$_{Gather}$
Slice 2
Project$_{s.beer, s.price}$
HashJoin$_{b.name = s.bar}$
Motion$_{Redist(b.name)}$
Scan$^{s}_{Sells}$
Motion$_{Redist(b.name)}$
Filter$_{b.city = 'San Francisco'}$
Scan$^{b}_{Bars}$
Slice 3

Pivotal

GPORCA find the best MPP plan

Pivotal

Outline

Context

Architecture

Develop Walkthrough

Build & Test

Contribute

GPORCA

CONTEXT

GPORCA = Greenplum ORCA = ...

Jul 2015, **PQO**: Pivotal Query Optimizer (Released in GPDB 4.3.5.0)

Jun 2014, **ORCA**: SIGMOD 2014 Paper

In source code **GPOPT**

`src/backend/gpopt`

psql: `select gp_opt_version();`

Great SQL Surface

GPORCA can optimize all

99 TPC-DS Queries

(and variations)

Reference: Orca: A Modular Query Optimizer Architecture for Big Data, SIGMOD 2014

Great SQL Surface of GPORCA

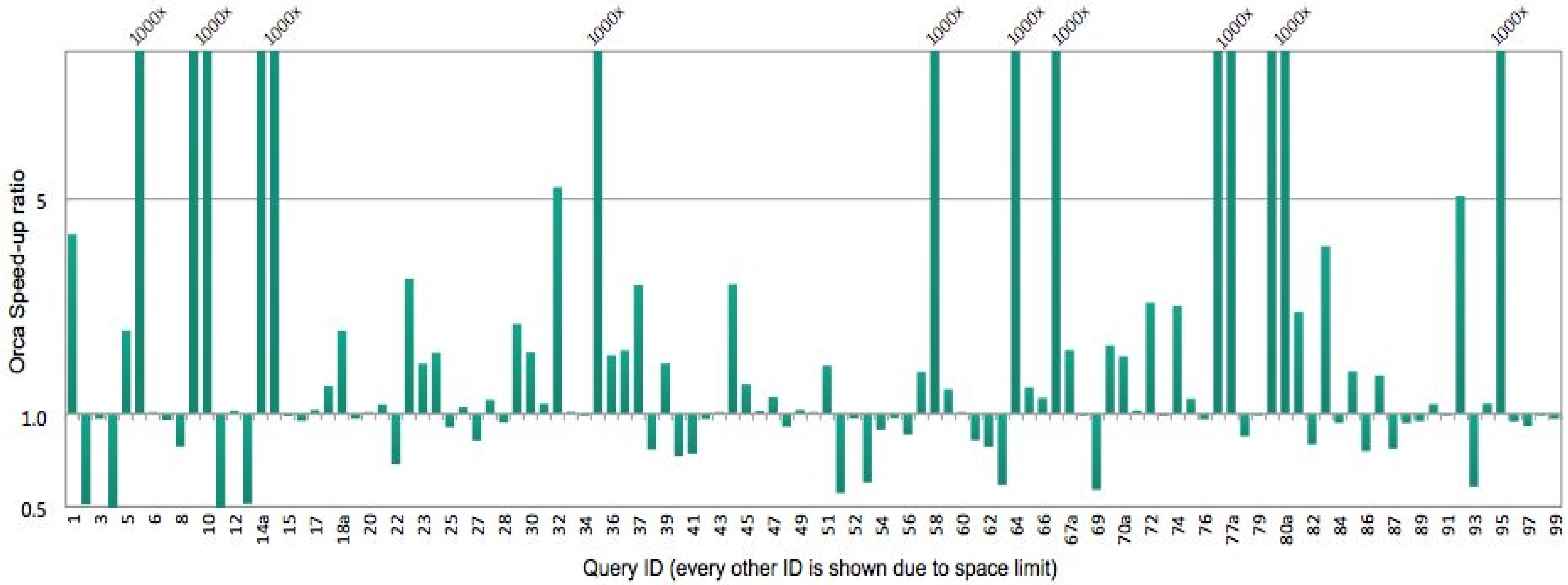
~60 Logical Operators

~50 Physical Operators

~40 Scalar Operators

~110 Transformation Rules

Great Query Performance vs. Planner



TPC-DS 10TB, 16 nodes, 48 GB/node

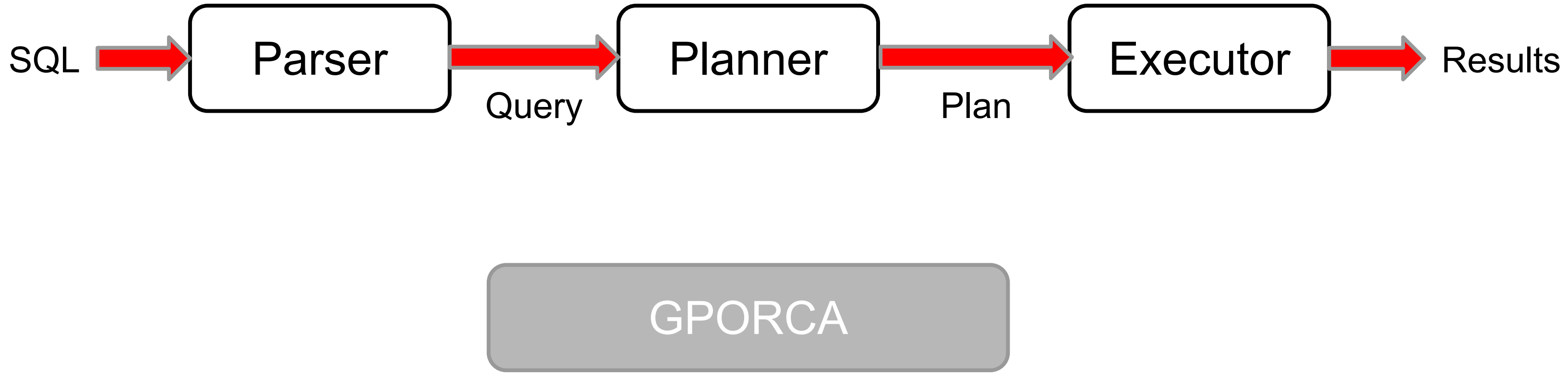
Key Features vs. Planner

- Smarter partition elimination
- Multi-level partitioning support (4.3.6)
- Subquery unnesting
- Common table expressions (CTE)
- Join on castable data types
- Improved join ordering (better join order)
- Join-Aggregate reordering (push/pull AGG over join)
- Sort order optimization (avoid sort multiple times)
- Skew awareness (redistribution vs. broadcast)

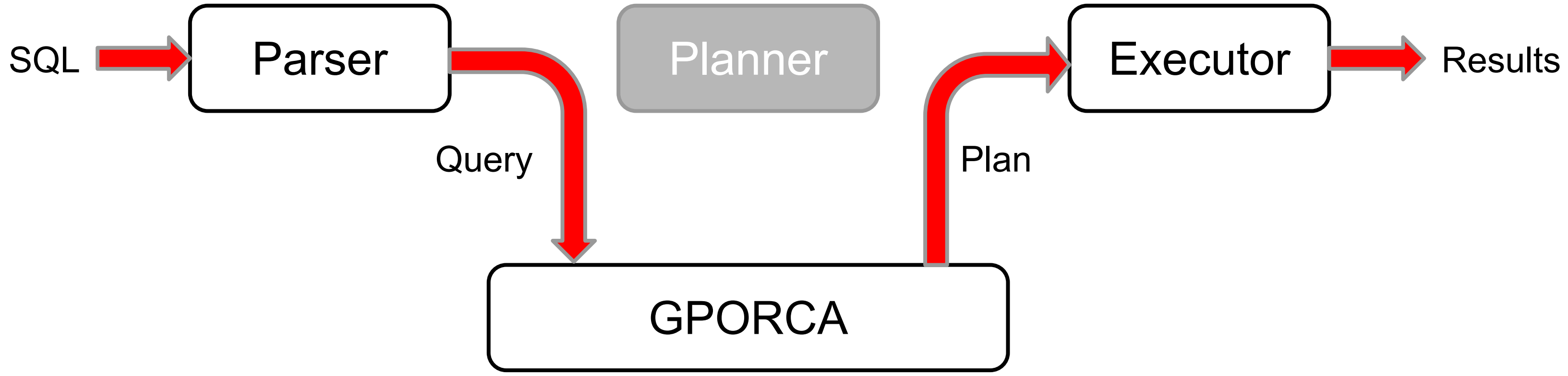
GPORCA

ARCHITECTURE

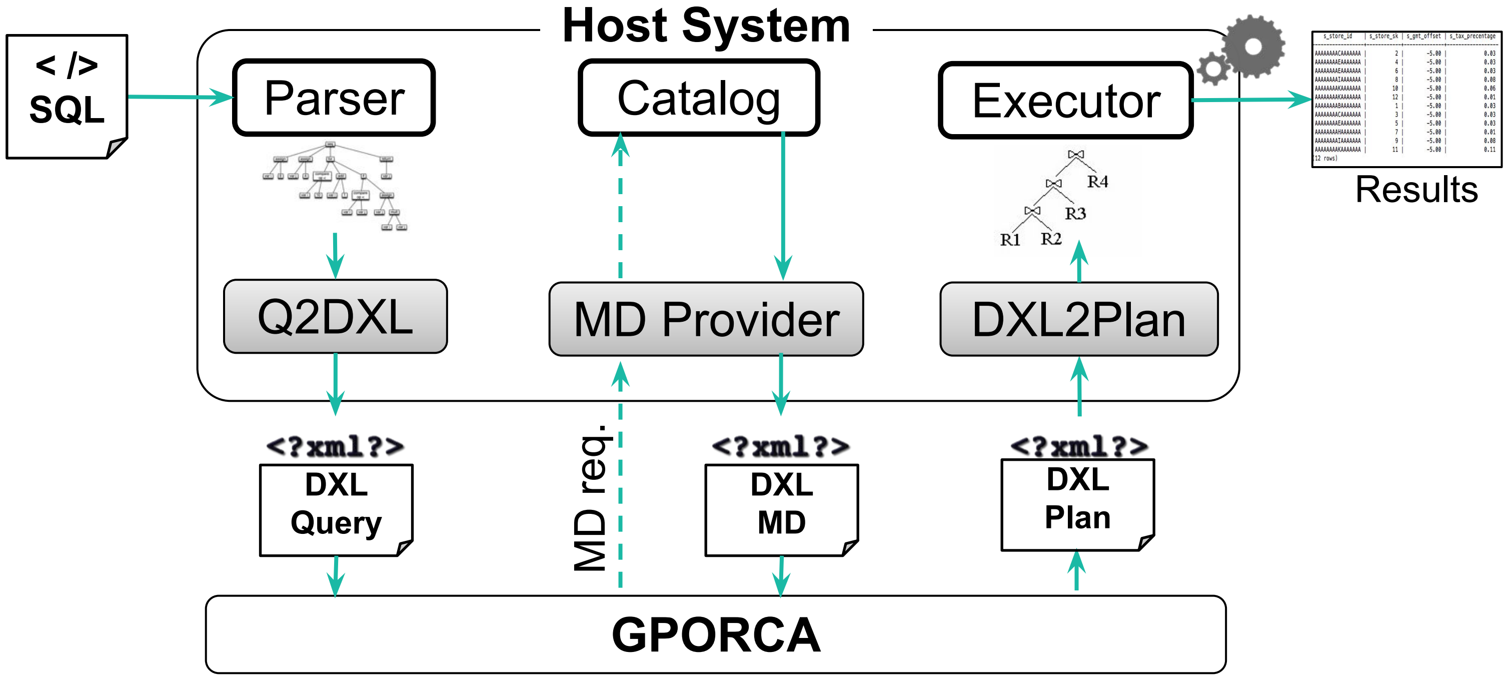
GPDB Using Planner (`set optimizer=off`)



GPDB Using GPORCA (`set optimizer=on`)



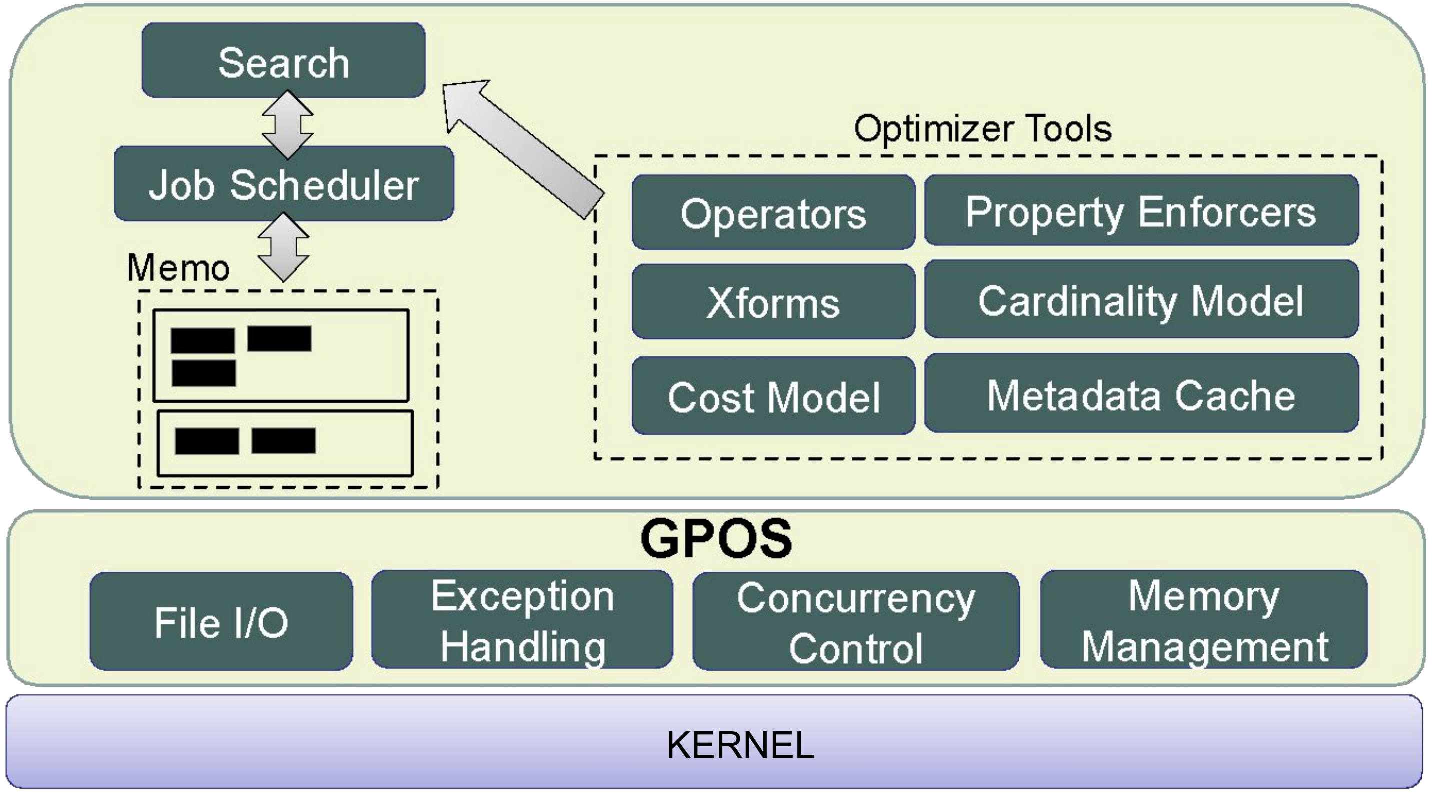
Multi-Host with DXL (Data eXchange Language) API



```
   s_store_id    | s_store_sk | s_gmt_offset | s_tax_precentage
-----------------+------------+--------------+------------------
AAAAAAAACAAAAAAA |          2 |        -5.00 |             0.03
AAAAAAAAEAAAAAAA |          4 |        -5.00 |             0.03
AAAAAAAAEAAAAAAA |          6 |        -5.00 |             0.03
AAAAAAAAIAAAAAAA |          8 |        -5.00 |             0.08
AAAAAAAAKAAAAAAA |         10 |        -5.00 |             0.06
AAAAAAAAKAAAAAAA |         12 |        -5.00 |             0.01
AAAAAAAABAAAAAAA |          1 |        -5.00 |             0.03
AAAAAAAACAAAAAAA |          3 |        -5.00 |             0.03
AAAAAAAAEAAAAAAA |          5 |        -5.00 |             0.03
AAAAAAAAHAAAAAAA |          7 |        -5.00 |             0.01
AAAAAAAAIAAAAAAA |          9 |        -5.00 |             0.08
AAAAAAAAKAAAAAAA |         11 |        -5.00 |             0.11
(12 rows)
```

Pivotal

GPORCA Architecture



Five Optimization Steps in Orca

[Step 0] Pre-Process: e.g., predicates pushdown

[Step 1] Exploration: All equivalent logical plans

[Step 2] Statistics Derivation: histograms

[Step 3] Implementation: Logical to Physical

[Step 4] MPP Optimization: Enforcing and Costing

Step 0 Pre-Process with 25 iterations (TL, DR)

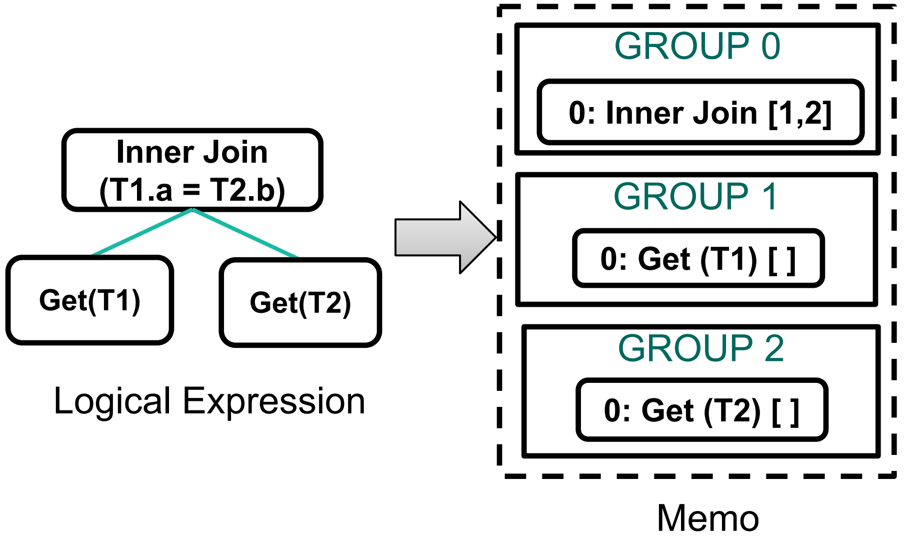
(1) Remove unused CTE anchors
(2) Remove intermediate superfluous limit
(3) Trim unnecessary existential subqueries
(4) Remove superfluous outer references from the outer spec in limits, grouping columns, partition/order columns in window operators
(5) Remove superfluous equality
(6) Simplify quantified subqueries
(7) Preliminary unnesting of scalar subqueries
(8) Unnest AND/OR/NOT predicates and ensure predicates are array IN or NOT IN where applicable
(9) Infer predicates from constraints
(10) Eliminate self comparisons
(11) Remove duplicate AND/OR children
(12) Fatorize common expressions
(13) Infer filters out of components of disjunctive filters
(14) Pre-process window functions
(15) Collapse cascaded union/union all
(16) Eliminate unused computed columns
(17) Normlize expression
(18) Transform outer join into inner join
(19) Collapse cascaded inner joins
(20) Generate more predicates from constraints
(21) Eliminate empty subtrees
(22) Collapse casecade of projects
(23) Insert project for scalar subquery return outer reference
(24) Reorder children of scalar comparison operator Rewrite IN subquery to EXIST subquery with a predicate

Step 1 Exploration 1/2: Memoization

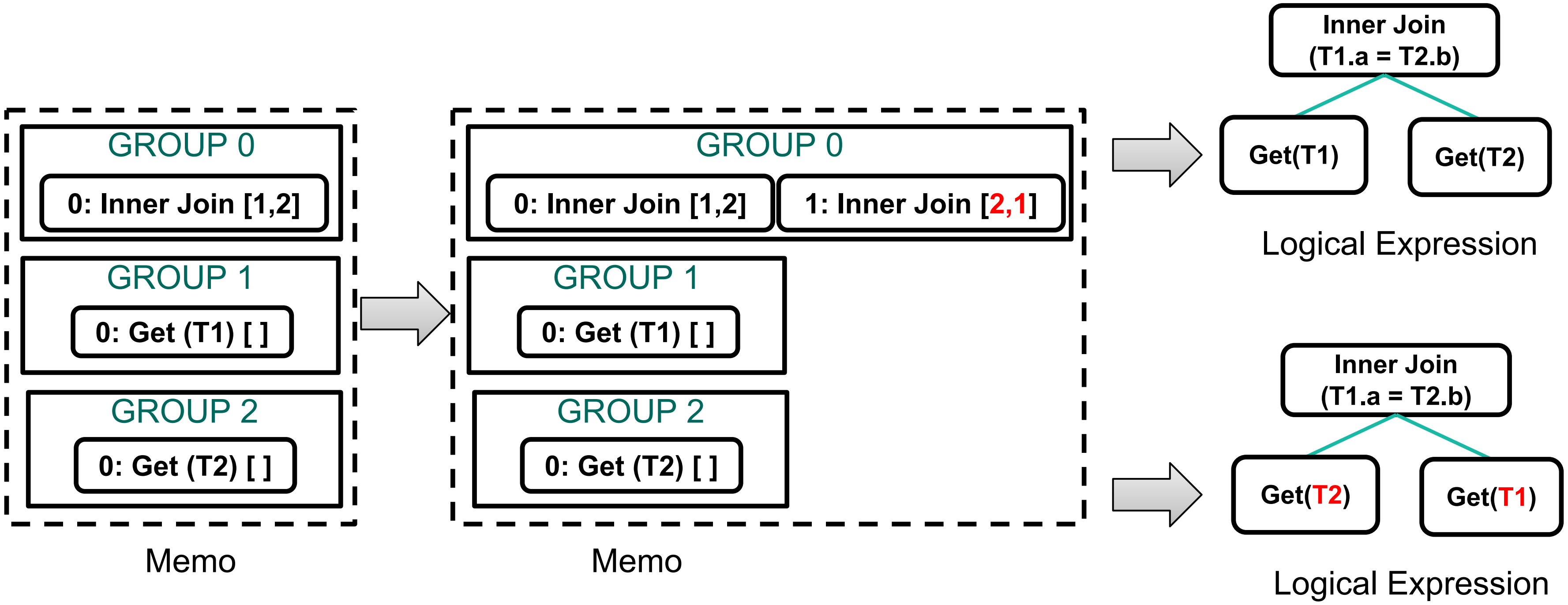
Compact in-memory data structure capturing plan space:

Group: Container of equivalent expressions

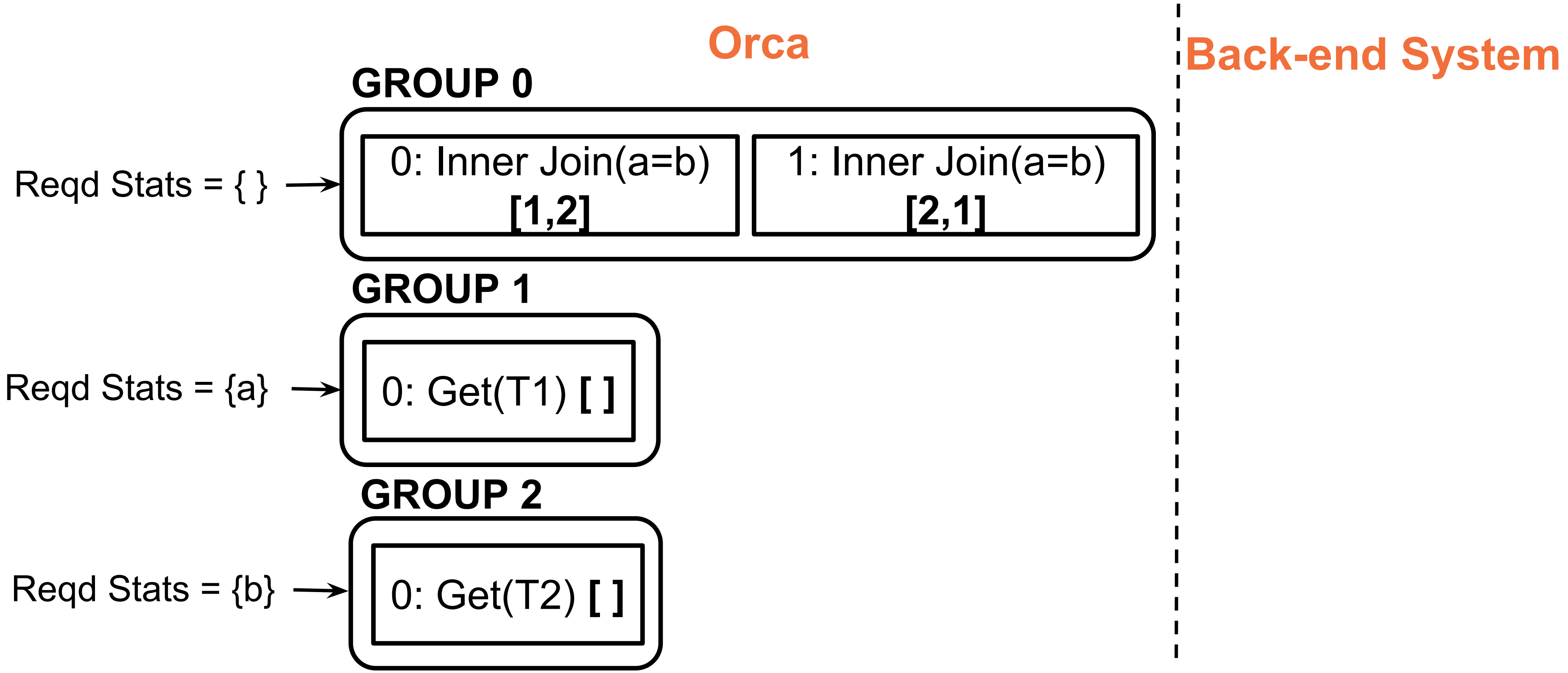
Group Expression: Operator that has other groups as its children



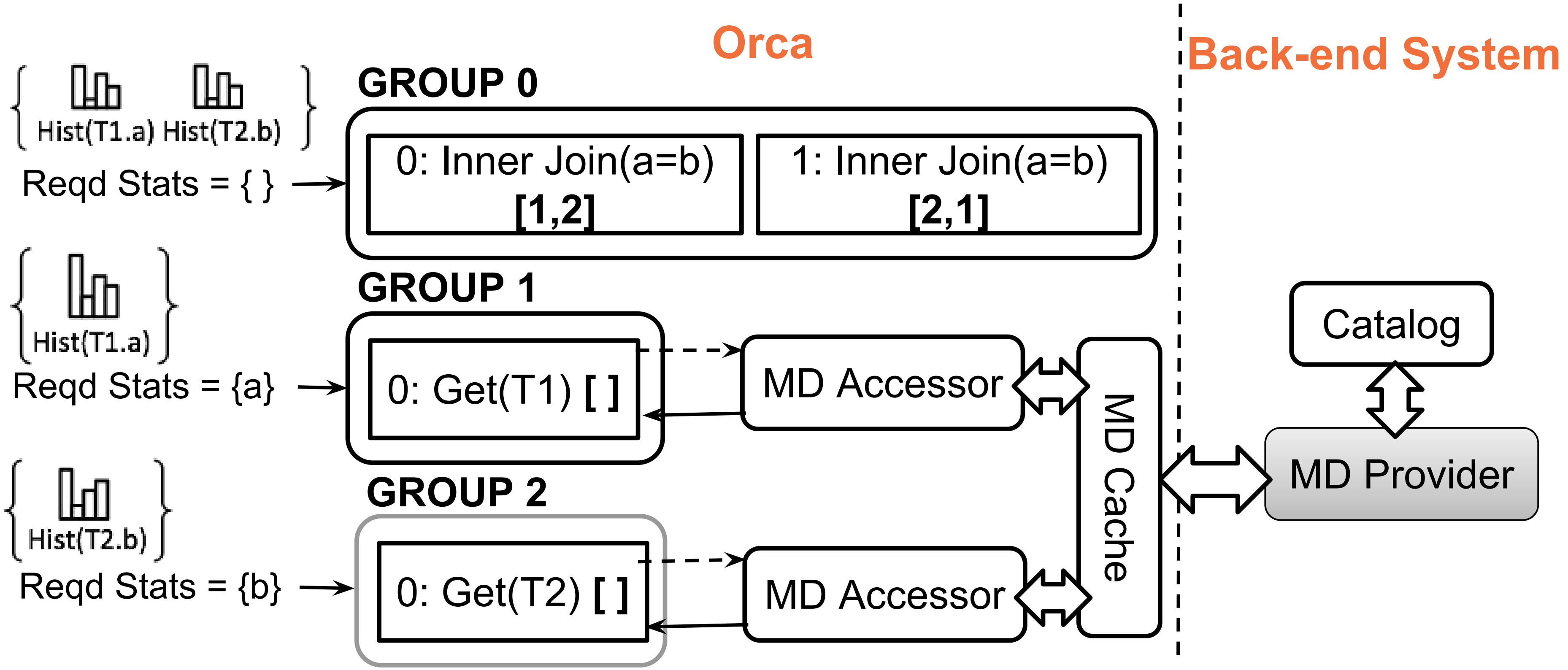
Step 1 Exploration 2/2: Transformation



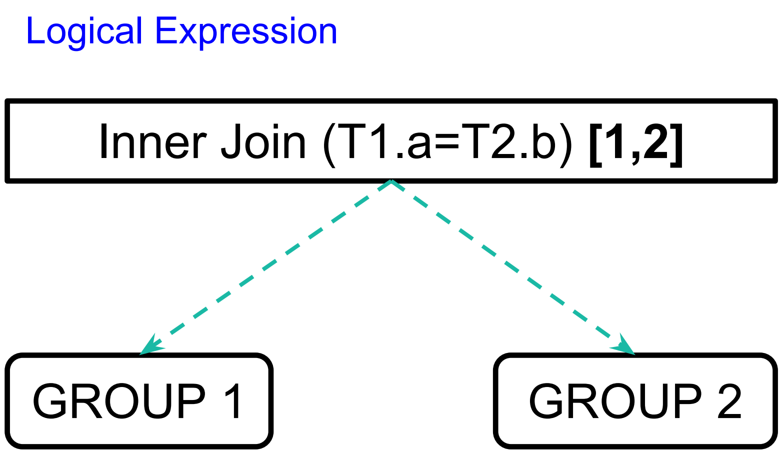
Step 2 Statistics Derivation 1/2

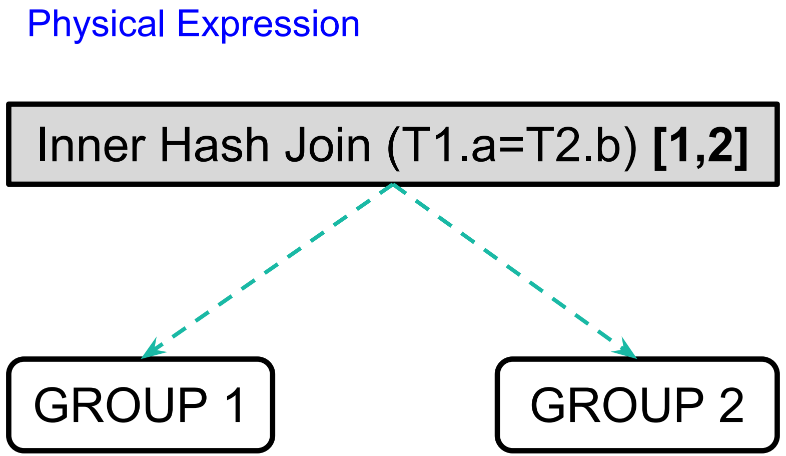


Step 2 Statistics Derivation 2/2

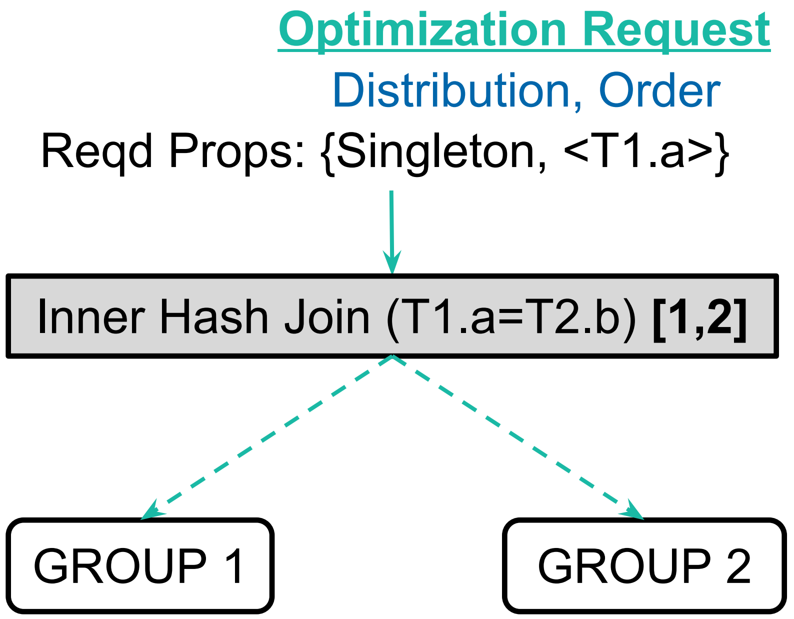


Step 3 Implementation

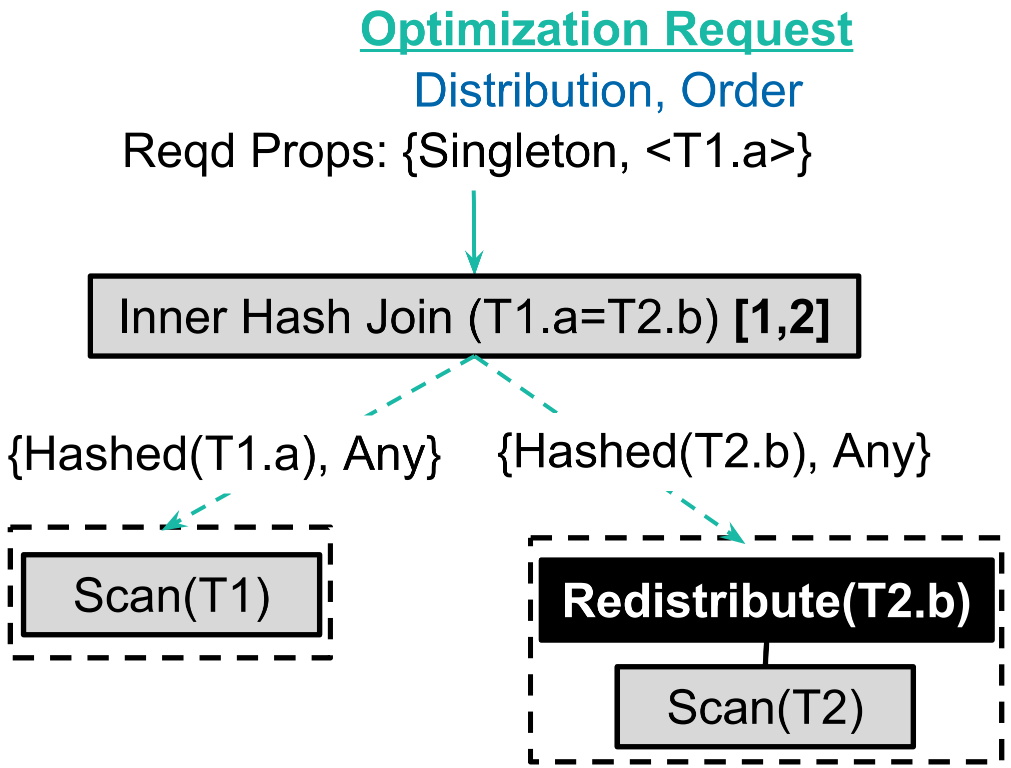




Step 4 MPP Optimization 1/3 Requirement

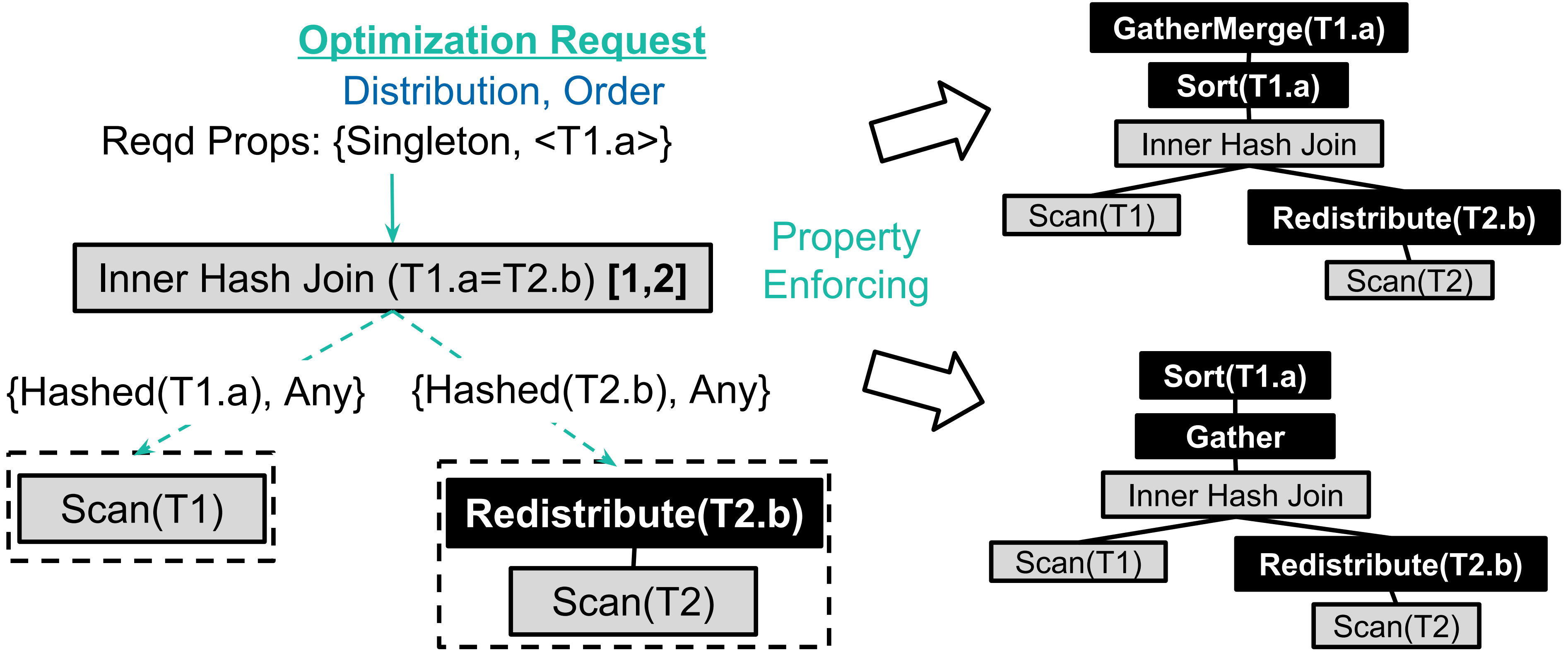

Optimization Request
Distribution, Order
Reqd Props: {Singleton, <T1.a>}
Inner Hash Join (T1.a=T2.b) [1,2]
GROUP 1
GROUP 2

Step 4 MPP Optimization 2/3 Costing



Pivotal

Step 4 MPP Optimization 3/3 Enforcement



GPORCA

DEVELOP

Pivotal

Walkthrough

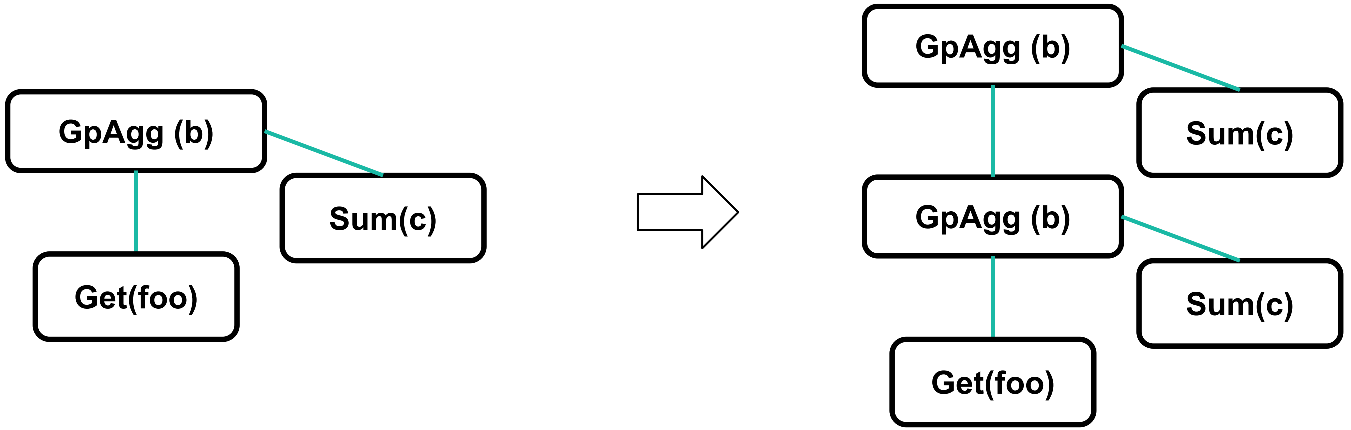
*Split an aggregate into a pair of **local** and **global** aggregate.*

```
CREATE TABLE foo (a int, b int, c int)
DISTRIBUTED BY (a);

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

Do **local** aggregation on segments
The **global** aggregation on master

Split Groupby Aggregate



CXformSplitGbAgg

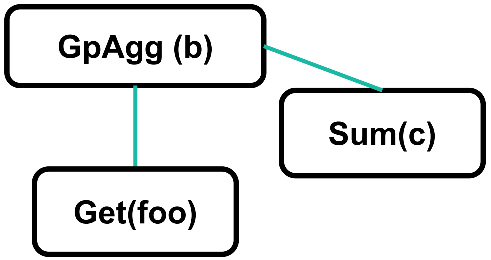
```
// HEADER FILES
~/orca/libgpopt/include/gpopt/xforms


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

Transformation Trigger

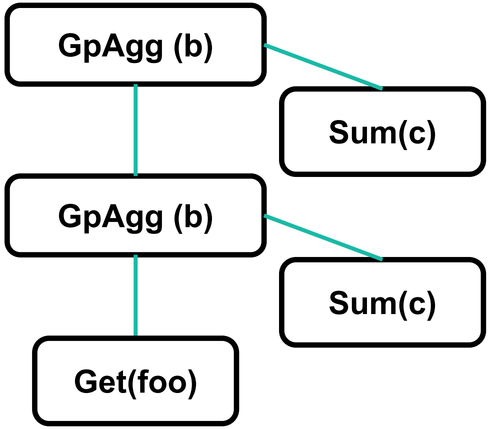
- Pattern
- Pre-Condition Check

Pattern



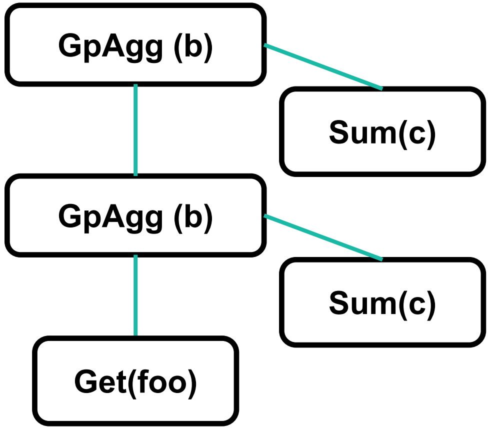
```
GPOS_NEW(pmp)
CExpression
(
    pmp,
    // logical aggregate operator
    GPOS_NEW(pmp) CLogicalGbAgg(pmp),
    // relational child
    GPOS_NEW(pmp) CExpression(pmp, GPOS_NEW(pmp) CPatternLeaf(pmp)),
    // scalar project list
    GPOS_NEW(pmp) CExpression(pmp, GPOS_NEW(pmp) CPatternTree(pmp))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

What's WRONG of this pattern?



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

Pre-Condition Check



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

Do not fire this rule on a logical operator produced by the same rule.
(Avoid Infinite Recursion)

The Actual Transformation

```
void Transform
         (
             CXformContext *pxfctxt, // update
             CXformResult *pxfres, // output
             CExpression *pexpr // input
         )
         const;
```

details: `libgpopt/src/xforms/CXformSplitGbAgg.cpp`

Pivotal

Register Transformation Rule

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

Pivotal

GPORCA

BUILD

Dependencies

GP-XERCES:

https://github.com/greenplum-db/gp-xerces

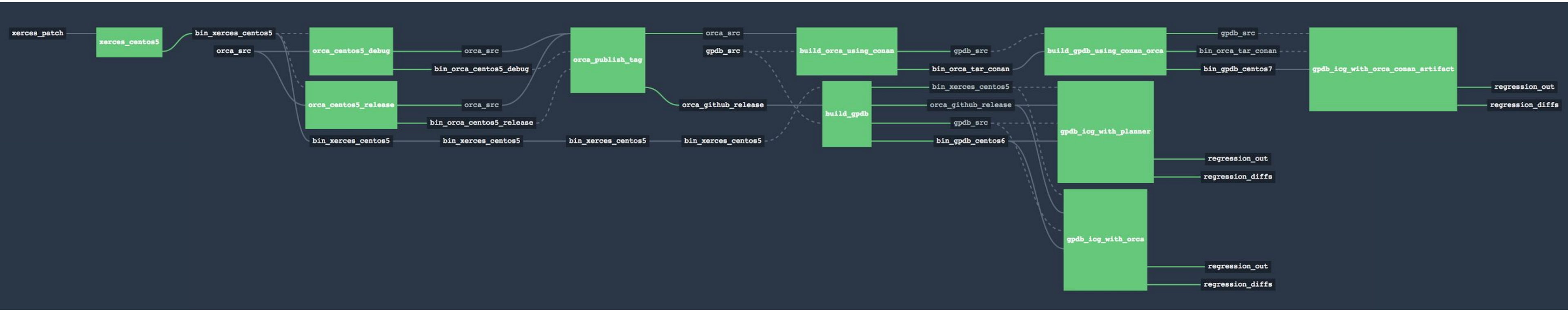
CMake 3.0+

CMake Build

```
mkdir build
cd build
cmake ../
make && make install
```

CI: Concourse Pipeline

https://ci.orca.pivotalci.info/teams/main/pipelines/gporca



GPORCA

TEST

Pivotal

Test GPORCA (<1min)

```
# run all unit tests
ctest

# run all unit tests in parallel with 7 threads
ctest -j7

# run only one unit test called CAggTest
./server/gporca_test -U CAggTest
```

Test with GPDB OSS in Docker

Follow instructions from:

https://github.com/d/bug-free-fortnight

It's very useful to verify installcheck-good locally with latest GPORCA changes.

GPORCA

CONTRIBUTE

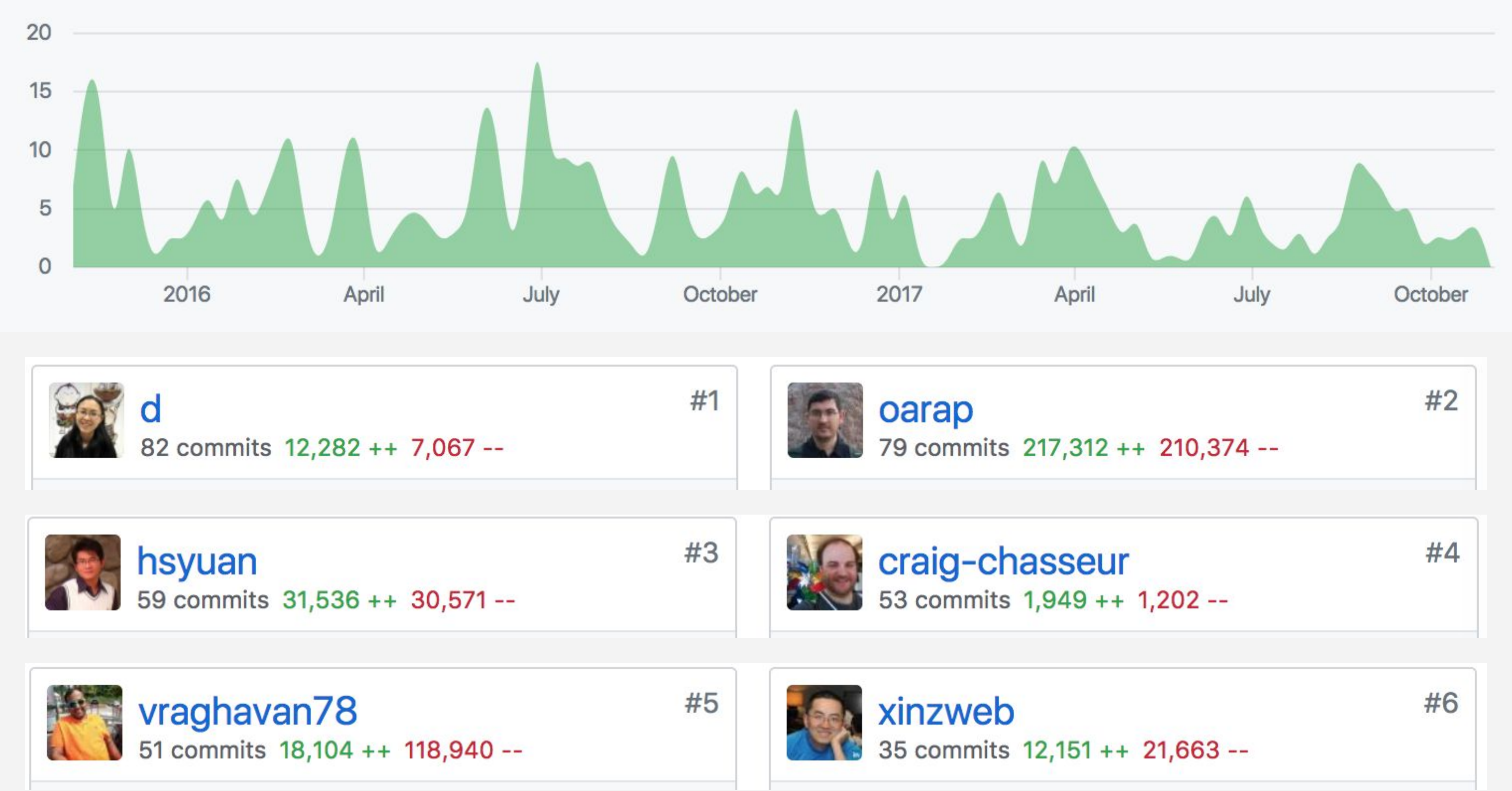
Pivotal

1-2-3

1. Fork GPORCA at
https://github.com/greenplum-db/gporca
2. Pick an issue
https://github.com/greenplum-db/gporca/issues
3. Send a Pull Request (PR)

Thanks to all these contributors

http://www.ceotodaymagazine.com/issue/issue-09-2017/#32



d #1
82 commits 12,282 ++ 7,067 --

oarap #2
79 commits 217,312 ++ 210,374 --

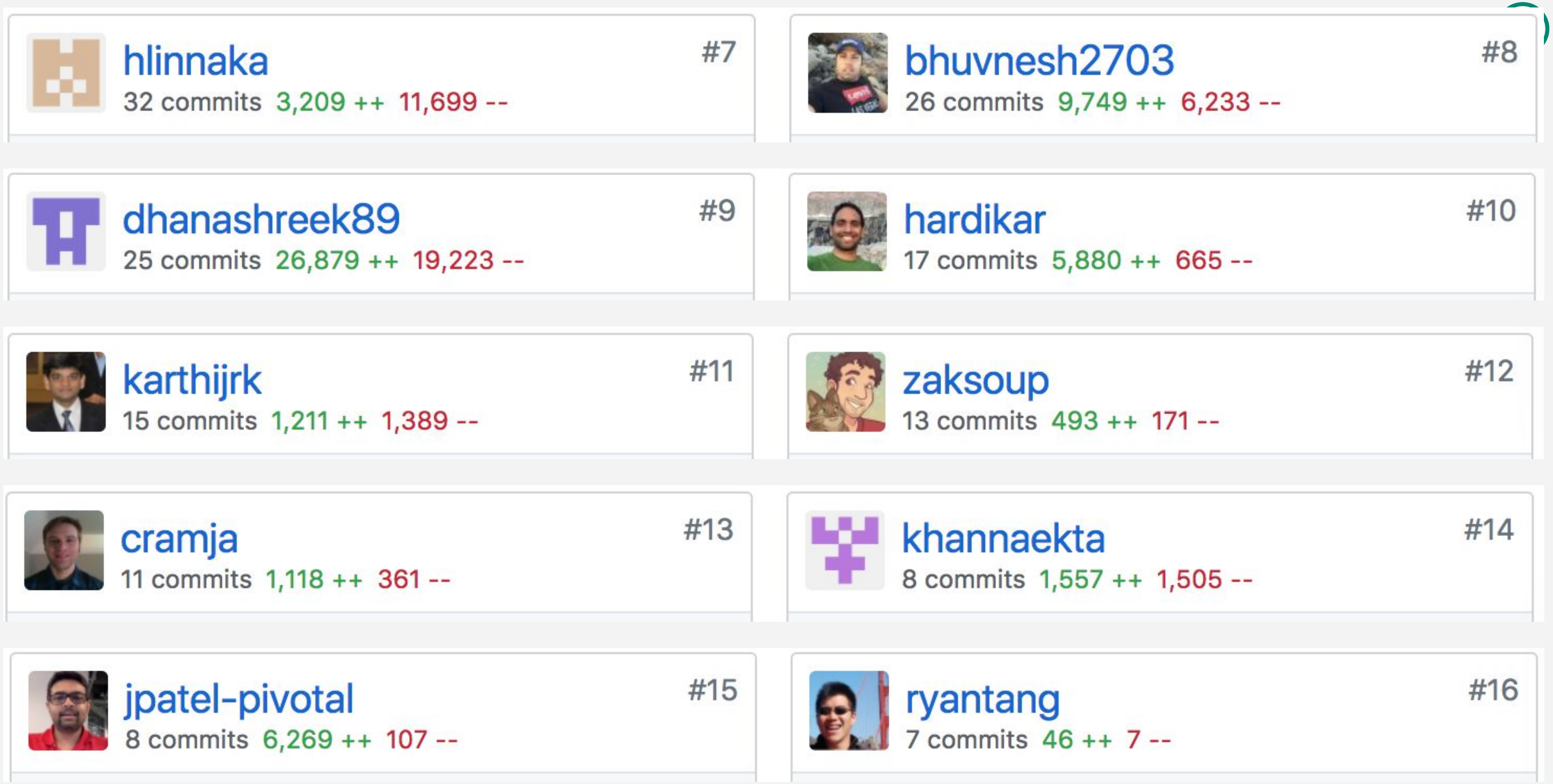
hsyuan #3
59 commits 31,536 ++ 30,571 --

craig-chasseur #4
53 commits 1,949 ++ 1,202 --

vraghavan78 #5
51 commits 18,104 ++ 118,940 --

xinzweb #6
35 commits 12,151 ++ 21,663 --

Pivotal



hlinnaka #7
32 commits 3,209 ++ 11,699 --

bhuvnesh2703 #8
26 commits 9,749 ++ 6,233 --

dhanashreek89 #9
25 commits 26,879 ++ 19,223 --

hardikar #10
17 commits 5,880 ++ 665 --

karthijrk #11
15 commits 1,211 ++ 1,389 --

zaksoup #12
13 commits 493 ++ 171 --

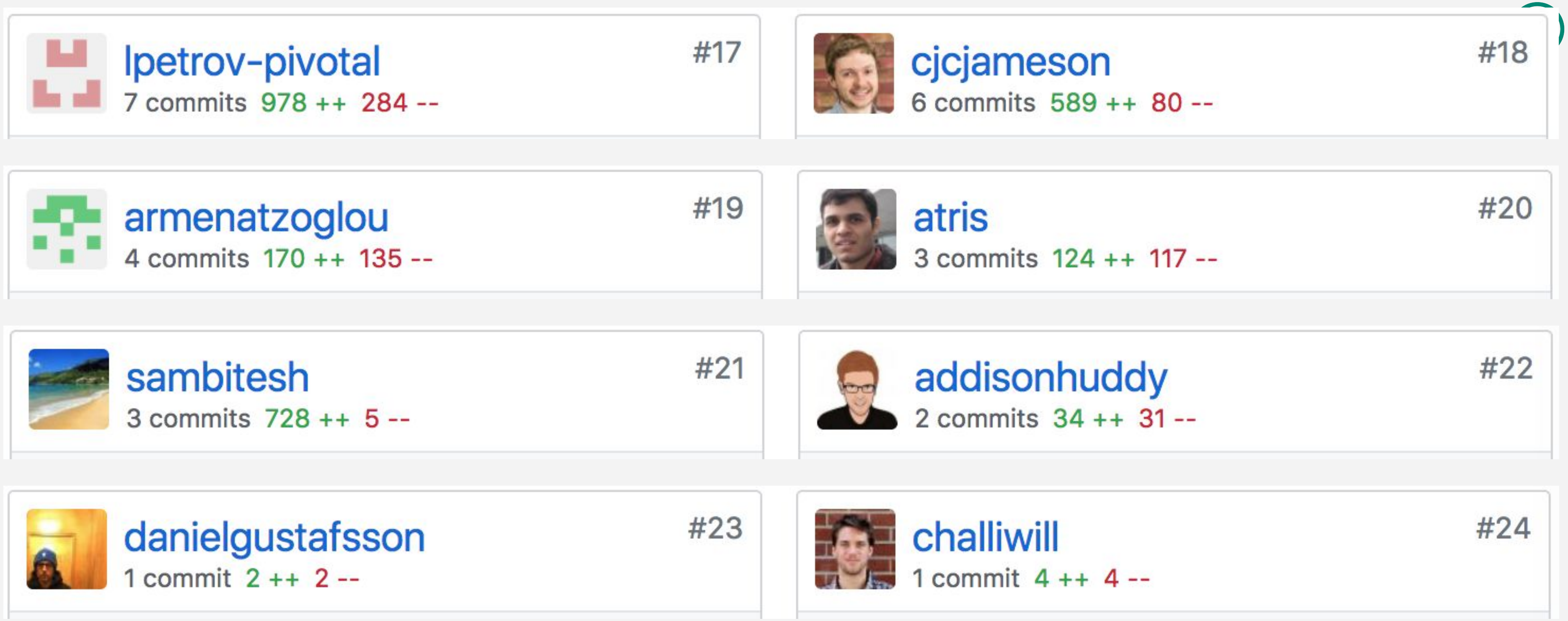
cramja #13
11 commits 1,118 ++ 361 --

khannaekta #14
8 commits 1,557 ++ 1,505 --

jpatel-pivotal #15
8 commits 6,269 ++ 107 --

ryantang #16
7 commits 46 ++ 7 --

Pivotal



lpetrov-pivotal #17
7 commits 978 ++ 284 --

cjcjameson #18
6 commits 589 ++ 80 --

armenatzoglou #19
4 commits 170 ++ 135 --

atris #20
3 commits 124 ++ 117 --

sambitesh #21
3 commits 728 ++ 5 --

addisonhuddy #22
2 commits 34 ++ 31 --

danielgustafsson #23
1 commit 2 ++ 2 --

challiwill #24
1 commit 4 ++ 4 --

Pivotal

You are invited!

Pivotal

THANK YOU

GPDB:

http://greenplum.org/

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

GPORCA Github:

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

mailing lists (400+ member):

gpdb-users@greenplum.org

Greenplum YouTube:

https://www.youtube.com/GreenplumDatabase

Pivotal

Pivotal®

Transforming How The World Builds Software

FAQ

Q: What happen to SORT generated in Enforcement?
A: There is only ONE implementation of SORT, so, that's a physical operator and won't be pushed anywhere after enforcement.

Q: Why CXformResult has more than one alternatives?
A: Some rule (e.g. CXformExpandNAryJoinDP) can produce multiple choices after transformation

Q: How hard to make GPORCA adapt to a new host?
A: The hard part is on the MD translation. So far, GPORCA is very PostgreSQL friendly.

Q: Why there is no separate SQL parser included in GPORCA?
A: GPORCA focused on relational algebra and let host handle the binding, view expansions, and permissions.

Q: How to add a new property like 'reliability' or 'dollar cost' to this optimizer?
A: That can be added to Property Enforcement as the Order/Distribution/Partition/Rewindability. For example, if people want to favor a more 'reliable' data source, they can add a CEnfdReliability class to cost that choice. It's an interesting combination of 'reliability' and 'dollar cost', usually, when it's more reliable is more expensive. It's an interesting balance to achieve.

Q: How long does the `ctest` run?
A: Around 5min on 2.8Ghz Intel i7. Running with `ctest -j7` finished in < 2min.

Q: Is GPORCA multi-threaded?
A: It's multi-thread READY, but currently, we run with single thread. There are still few caveats (thread safe issues) to iron out before we can fully turn it on.

Publications

Orca: A Modular Query Optimizer Architecture for Big Data, SIGMOD 2014
Mohamed A. Soliman, Lyublena Antova, Venkatesh Raghavan, Amr El-Helw, Zhongxian Gu, Entong Shen, George C. Caragea, Carlos Garcia-Alvarado, Foyzur Rahman, Michalis Petropoulos, Florian Waas, Sivaramakrishnan Narayanan, Konstantinos Krikellas, Rhonda Baldwin

Optimization of Common Table Expressions in MPP Database Systems, VLDB 2015
Amr El-Helw, Venkatesh Raghavan, Mohamed A. Soliman, George C. Caragea, Zhongxian Gu, Michalis Petropoulos.

Optimizing Queries over Partitioned Tables in MPP Systems, SIGMOD 2014
Lyublena Antova, Amr El-Helw, Mohamed Soliman, Zhongxian Gu, Michalis Petropoulos, Florian Waas

Reversing Statistics for Scalable Test Databases Generation, DBTest 2013
Entong Shen, Lyublena Antova

Total Operator State Recall - Cost-Effective Reuse of Results in Greenplum Database, ICDE Workshops 2013
George C. Caragea, Carlos Garcia-Alvarado, Michalis Petropoulos, Florian M. Waas

Testing the Accuracy of Query Optimizers, DBTest 2012
Zhongxian Gu, Mohamed A. Soliman, Florian M. Waas

Automatic Capture of Minimal, Portable, and Executable Bug Repros using AMPERe, DBTest 2012
Lyublena Antova, Konstantinos Krikellas, Florian M. Waas

Automatic Data Placement in MPP Databases, ICDE Workshops 2012
Carlos Garcia-Alvarado, Venkatesh Raghavan, Sivaramakrishnan Narayanan, Florian M. Waas

GPORCA

DEBUG

Pivotal

GPORCA Traceflags and GPDB GUC

GPORCA relies on Traceflags to change runtime behavior: **Traceflag.h**

Exposed in GPDB as GUC (Grand Unified Configuration): **guc_gp.c**

```
-- turn on GPORCA

set optimizer=on;

-- print input query (GPORCA TF 101000)

set optimizer_print_query=on;
```

Minidump: DXL document

Turn on the minidump

```
set client_min_messages='log';
set optimizer=on;
set optimizer_enable_constant_expression_evaluation=off;
set optimizer_enumerate_plans=on;
set optimizer_minidump=always;
```

Run a query

GPORCA creates a *.mdp file in the **$MASTER_DATA_DIRECTORY/minidump**

```
# run only one minidump directly
./server/gporca_test -d ../data/dxl/minidump/TVFRandom.mdp
```

Input Query, Output Plan

Debug the plans

```
set client_min_messages='log';

set optimizer=on;

set optimizer_print_query=on; -- input query, and
preprocessed query

set optimizer_print_plan=on; -- output final physical plan
```

Plan Enumeration

Turn on the plan enumerations

```
set client_min_messages='log';

set optimizer=on;

set optimizer_enumerate_plans=on;
```

Pick a plan out of search space

```
set optimizer=on;

set client_min_messages='log';

set optimizer_enumerate_plans=on;

set optimizer_plan_id=1;
```

Pivotal

Optimization Stats and Xform Rules

Debug optimizer stages

```
set client_min_messages='log';

set optimizer=on;

set optimizer_print_optimization_stats=on;
```

Debug the transformation rules details

```
set client_min_messages='log';

set optimizer=on;

set optimizer_print_xform=on;
```

MEMO Groups

```
set optimizer_print_memo_after_exploration=on;

set optimizer_print_memo_after_implementation=on;

set optimizer_print_memo_after_optimization=on;
```

ROOT group is indicated as `ROOT`

Way to Disable Xform Rules

```
select disable_xform('CXformJoinAssociativity');

select enable_xform('CXformJoinAssociativity');
```

All the xform rules can be found from the class names under **`libgpopt/include/gpopt/xforms`**

`CXformFactory::Instantiate` lists all the activated xform rules (~130 rules)

Useful Breakpoints

```
# Entry point of optimizer
         COptimizer::PdxlnOptimize
# DXL: Translate DXL into Query
         CTranslatorDXLToExpr::PexprTranslateQuery
# Step 1: Pre-processor
         CExpressionPreprocessor::PexprPreprocess
# Step 2-3-4: Optimization
         COptimizer::PexprOptimize
# Individual rule transformation, all CXform* classes
         CXformSplitGbAgg::Transform
# Enforceable Property
         CEngine::FCheckEnfdProps
         CPartitionPropagationSpec::AppendEnforcers
# DXL: Translate Plan back in DXL
         CTranslatorExprToDXL::PdxlnTranslate
```

GPORCA

CODE BASE

Pivotal

Top Level

```
.
├── cmake
├── concourse
├── data
├── libgpdbcost
├── libgpopt
├── libgpos
├── libnaucrates
├── patches
├── scripts
└── server
```

Pivotal

libgpos: memory management, task scheduler, exception handling, unit-test framework

```
libgpos
├── include
└── src
    ├── common
    ├── error
    ├── io
    ├── memory
    ├── net
    ├── string
    ├── sync
    ├── task
    └── test
```

```
├── server
│   ├── include
│   └── src
│       ├── startup
│       └── unittest
│           └── gpos
│               ├── common
│               ├── error
│               ├── io
│               ├── memory
│               ├── string
│               ├── sync
│               ├── task
│               └── test
│
```

libnaucrates: DXL, metadata, statistics, traceflags

```
libnaucrates
├── include
│   └── naucrates
│       ├── base
│       ├── dxl
│       │   ├── operators
│       │   ├── parser
│       │   └── xml
│       ├── md
│       ├── statistics
│       └── traceflags
└── src
    ├── base
    ├── md
    ├── operators
    ├── parser
    ├── statistics
    └── xml
```

libgpopt: engine, metadata cache, minidump, operators, memo, xform rules

```
libgpopt
├── include
│   └── gpopt
└── src
    ├── base
    ├── engine
    ├── eval
    ├── mdcache
    ├── metadata
    ├── minidump
    ├── operators
    ├── optimizer
    ├── search
    ├── translate
    └── xforms
```

libgpdbcost: cost model

```
libgpdbcost
├── CMakeLists.txt
├── include
│   └── gpdbcost
└── src
    ├── CCostModelGPDB.cpp
    ├── CCostModelGPDBLegacy.cpp
    ├── CCostModelParamsGPDB.cpp
    ├── CCostModelParamsGPDBLegacy.cpp
    └── ICostModel.cpp
```

server: unit tests

```
server
├── include
└── src
    ├── startup
    └── unittest
        ├── dxl
        │   ├── base
        │   └── statistics
        └── gpopt
            ├── base
            ├── cost
            ├── csq
            ├── engine
            ├── eval
            ├── mdcache
            ├── metadata
            ├── minidump
            ├── operators
            ├── search
            ├── translate
            └── xforms
```

Data: all the test data

```
data
├── dxl
│   ├── cost
│   ├── csq_tests
│   ├── expressiontests
│   ├── indexjoin
│   ├── metadata
│   ├── minidump
│   │   ├── CArrayExpansionTest
│   │   ├── CJoinOrderDPTest
│   │   ├── CPhysicalParallelUnionAllTest
│   │   ├── CPruneColumnsTest
│   │   └── sql
│   ├── multilevel-partitioning
│   ├── parse_tests
│   ├── plstmt
│   ├── query
│   ├── search
│   ├── statistics
│   ├── tpcds
│   ├── tpcds-partitioned
│   ├── tpch
│   └── tpch-partitioned
│
```

Pivotal