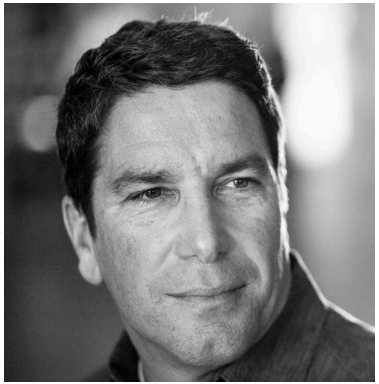
The Evolution of a Relational Database Layer over HBase

@ApachePhoenix
http://phoenix.apache.org/
James Taylor (@JamesPlusPlus)
Maryann Xue (@MaryannXue)

About James



- Architect at Salesforce.com
  - Part of the Big Data group
- Lead of Apache Phoenix project
- PMC member of Apache Calcite
- Engineer and Product Manager at BEA Systems
  - XQuery-based federated query engine
  - SQL-based complex event processing engine
- Various startups prior to that

APACHE PHOENIX

Agenda

- What is Apache Phoenix?
- State of the Union
- A Deeper Look
  - Joins and Subquery Support
- What's New?
- What's Next?
- Q&A

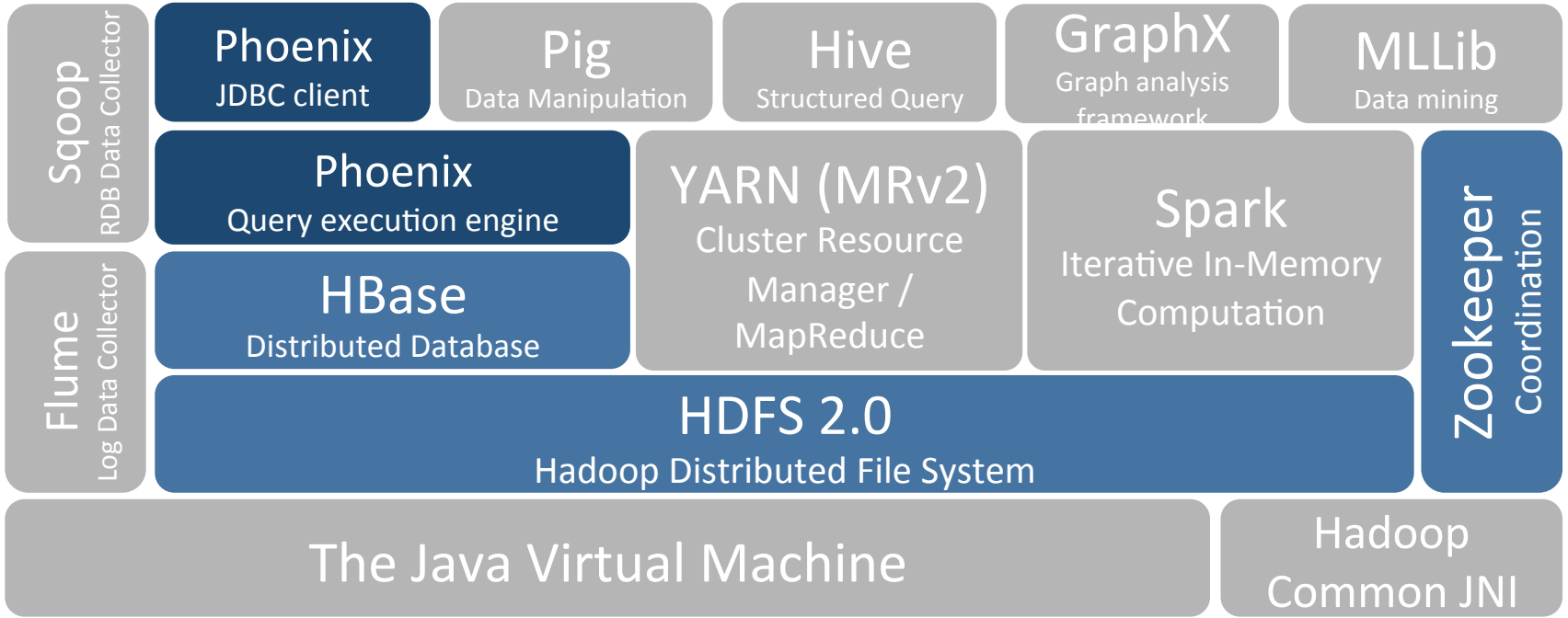
APACHE PHOENIX

What is Apache Phoenix?

- A relational database layer for Apache HBase
  - Query engine
    - Transforms SQL queries into native HBase API calls
    - Pushes as much work as possible onto the cluster for parallel execution
  - Metadata repository
    - Typed access to data stored in HBase tables
  - A JDBC driver
- A top level Apache Software Foundation project
  - Originally developed at Salesforce
  - Now a top-level project at the ASF (Happy Birthday!)
  - A growing community with momentum

APACHE PHOENIX

Where Does Phoenix Fit In?



APACHE PHOENIX

State of the Union

- Broad enough SQL support to run TPC queries
  - Joins, Sub-queries, Derived tables, etc.
- Three different secondary indexing strategies
  - Immutable for write-once/append only data
  - Global for read-heavy mutable data
  - Local for write-heavy mutable or immutable data
- Statistics driven parallel execution
- Tracing and metrics for Monitoring & Management

APACHE PHOENIX

About Maryann

- Software Engineer at Intel
- PMC Member of Apache Phoenix project
  - Joins, Subqueries, Phoenix+Calcite Integration, etc.
- IDH and Intel XML Libraries
  - The HBase part of Intel's Distribution of Hadoop
  - XSLT compiler of Intel XML Libraries

APACHE PHOENIX

Join and Subquery Support

- Grammar: inner join; left/right/full outer join; cross join
- Additional: semi join; anti join
- Algorithms: hash-join; sort-merge-join
- Optimizations:
  - Predicate push-down
  - FK-to-PK join optimization
  - Global index with missing data columns
  - Correlated subquery rewrite

TPC Example 1

Small-Quantity-Order Revenue Query (Q17)

```
select sum(l_extendedprice) / 7.0 as avg_yearly
from lineitem, part
where p_partkey = l_partkey
  and p_brand = '[B]'
  and p_container = '[C]'
  and l_quantity < (
    select 0.2 * avg(l_quantity)
    from lineitem
    where l_partkey = p_partkey
  );
```

```
CLIENT 4-WAY FULL SCAN OVER lineitem
    PARALLEL INNER JOIN TABLE 0
        CLIENT 1-WAY FULL SCAN OVER part
            SERVER FILTER BY p_partkey = '[B]' AND p_container = '[C]'
    PARALLEL INNER JOIN TABLE 1
        CLIENT 4-WAY FULL SCAN OVER lineitem
            SERVER AGGREGATE INTO DISTINCT ROWS BY l_partkey
    AFTER-JOIN SERVER FILTER BY l_quantity < $0
```

APACHE PHOENIX

TPC Example 2

Order Priority Checking Query (Q4)

```
select o_orderpriority, count(*) as order_count
from orders
where o_orderdate >= date '[D]'
    and o_orderdate < date '[D]' + interval '3' month
    and exists (
        select * from lineitem
        where l_orderkey = o_orderkey and l_commitdate < l_receiptdate
    )
group by o_orderpriority
order by o_orderpriority;
```

```
CLIENT 4-WAY FULL SCAN OVER orders
    SERVER FILTER o_orderdate >= '[D]' AND o_orderdate < '[D]' + 3(d)
    SERVER AGGREGATE INTO ORDERED DISTINCT ROWS BY o_orderpriority
CLIENT MERGE SORT
    SKIP-SCAN JOIN TABLE 0
        CLIENT 4-WAY FULL SCAN OVER lineitem
            SERVER FILTER BY l_commitdate < l_receiptdate
    DYNAMIC SERVER FILTER BY o_orderkey IN l_orderkey
```

APACHE PHOENIX

Join support - what can't we do?

- Nested Loop Join
- Statistics Guided Join Algorithm
  - Smartly choose the smaller table for the build side
  - Smartly switch between hash-join and sort-merge-join
  - Smartly turn on/off FK-to-PK join optimization

What's New?

- HBase 1.0 Support
- Functional Indexes

APACHE PHOENIX

Functional Indexes

- Creating an index on an expression as opposed to just a column value. For example, the following will be a full table scan:

```
SELECT AVG(response_time) FROM SERVER_METRICS
WHERE DAYOFMONTH(create_time) = 1
```

- Adding the following functional index will turn it into a range scan:

```
CREATE INDEX day_of_month_idx
ON SERVER_METRICS (DAYOFMONTH(create_time))
INCLUDE (response_time)
```

APACHE PHOENIX

What's New?

- HBase 1.0 Support
- Functional Indexes
- User Defined Functions

APACHE PHOENIX

User Defined Functions

- Extension points to Phoenix for domain-specific functions. For example, a geo-location application might load a set of UDFs like this:

```
CREATE FUNCTION WOEID_DISTANCE(INTEGER,INTEGER)
RETURNS INTEGER AS 'org.apache.geo.woeidDistance'
USING JAR '/lib/geo/geoloc.jar'
```

- Querying, functional indexing, etc. then possible:

```
SELECT * FROM woeid a JOIN woeid b ON a.country = b.country
WHERE woeid_distance(a.ID,b.ID) < 5
```

APACHE PHOENIX

What's New?

- HBase 1.0 Support
- Functional Indexes
- User Defined Functions
- Query Server with Thin Driver

APACHE PHOENIX

Query Server + Thin Driver

- Offloads query planning and execution to different server(s)
- Minimizes client dependencies
  - Enabler for ODBC driver (not available yet, though)
- Connect like this instead:

```
Connection conn = DriverManager.getConnection(
        "jdbc:phoenix:thin:url=http://localhost:8765");
```

- Still evolving, so **no backward compatibility guarantees yet**
- For more information, see http://phoenix.apache.org/server.html

APACHE PHOENIX

What's New?

- HBase 1.0 Support
- Functional Indexes
- User Defined Functions
- Query Server with Thin Driver
- Union All support
- Testing at scale with Pherf
- MR index build
- Spark integration
- Date built-in functions – WEEK, DAYOFMONTH, etc.
- Transactions (WIP - will be in next release)

APACHE PHOENIX

Transactions

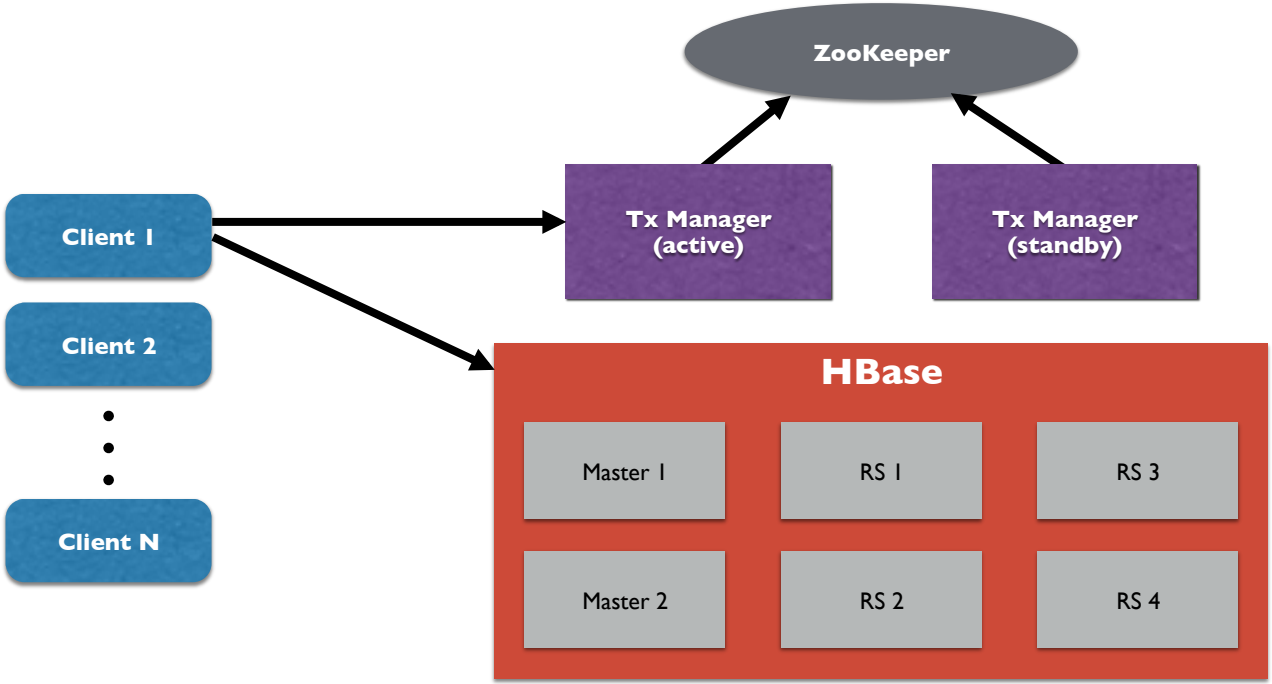
- Snapshot isolation model
  - Using Tephra (http://tephra.io/)
  - Supports REPEABLE_READ isolation level
  - Allows reading your own uncommitted data
- Optional
  - Enabled on a table by table basis
  - No performance penalty when not used
- Work in progress, but close to release
  - Come by the Phoenix booth for a demo
  - Try our txn branch
  - Will be available in next release

APACHE PHOENIX

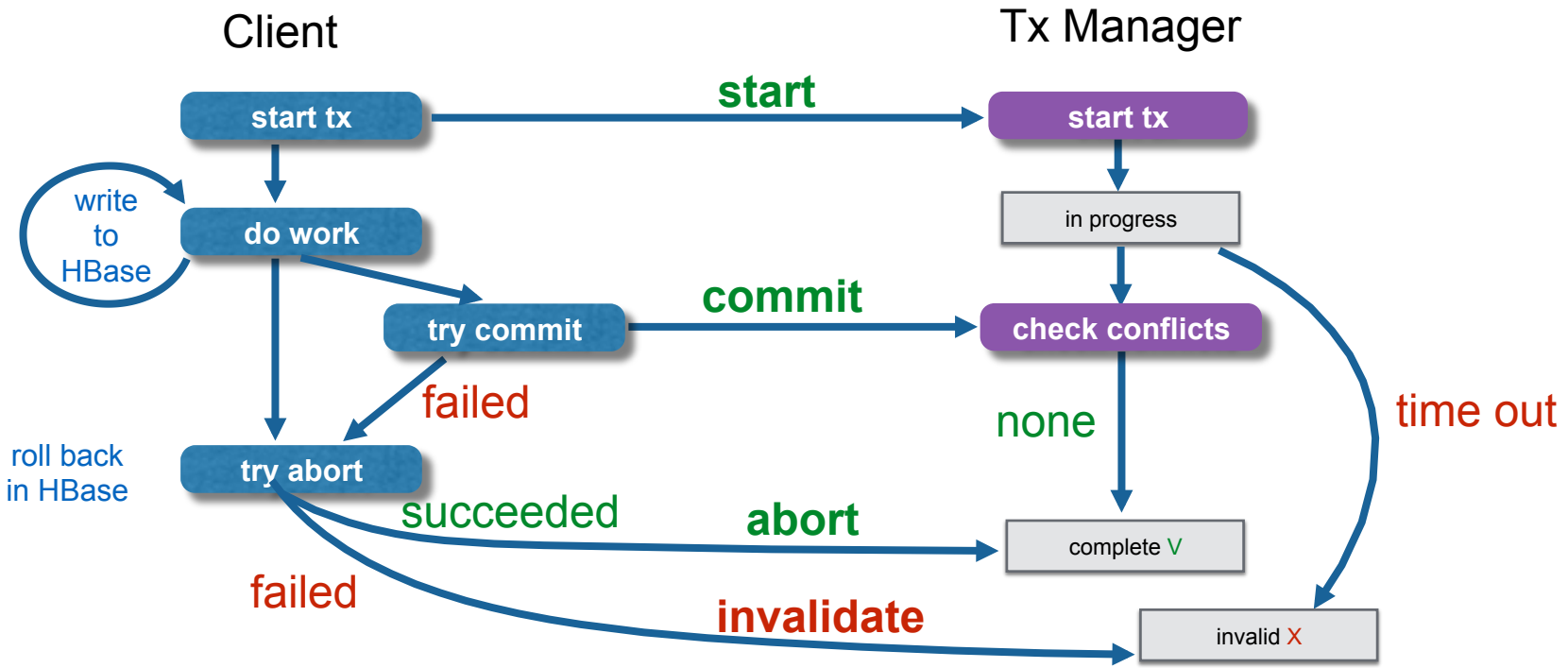
Optimistic Concurrency Control

- **Avoids cost of locking** rows and tables
- **No deadlocks** or lock escalations
- **Cost of conflict** detection and possible rollback is higher
- **Good if conflicts are rare**: short transaction, disjoint partitioning of work
- **Conflict detection not always necessary**: write-once/ append-only data

Tephra Architecture



Transaction Lifecycle



Tephra Architecture

- **TransactionAware client**
  - Coordinates transaction lifecycle with manager
  - Communicates directly with HBase for reads and writes
- **Transaction Manager**
  - Assigns transaction IDs
  - Maintains state on in-progress, committed and invalid transactions
- **Transaction Processor** coprocessor
  - Applies server-side filtering for reads
  - Cleans up data from failed transactions, and no longer visible versions

What's New?

- HBase 1.0 Support
- Functional Indexes
- User Defined Functions
- Query Server with Thin Driver
- Union All support
- Testing at scale with Pherf
- MR index build
- Spark integration
- Date built-in functions – WEEK, DAYOFMONTH, etc.
- Transactions (WIP - will be in next release)

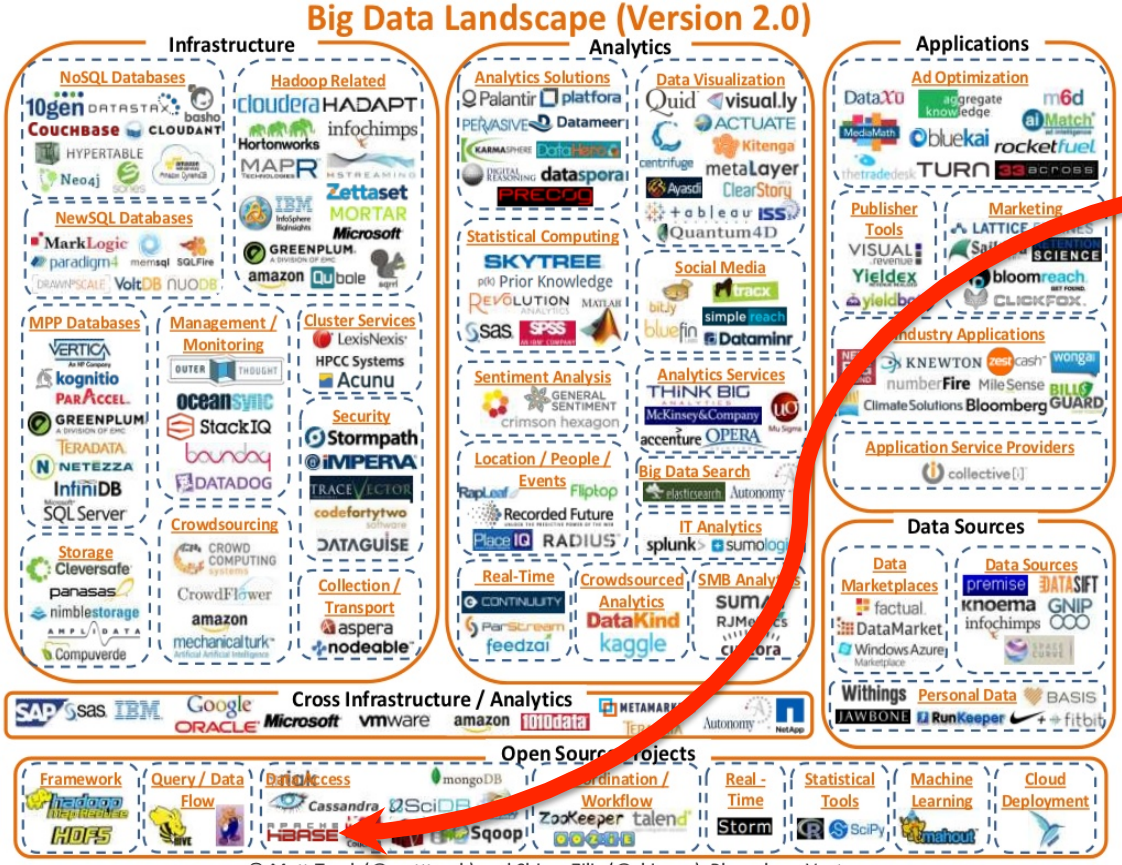
APACHE PHOENIX

What's Next?

- Is Phoenix done?
- What about the *Big Picture*?
  - How can Phoenix be leveraged in the larger ecosystem?
  - Hive, Pig, Spark, MR integration with Phoenix exists today, but not a great story

APACHE PHOENIX

What's Next?



You are here

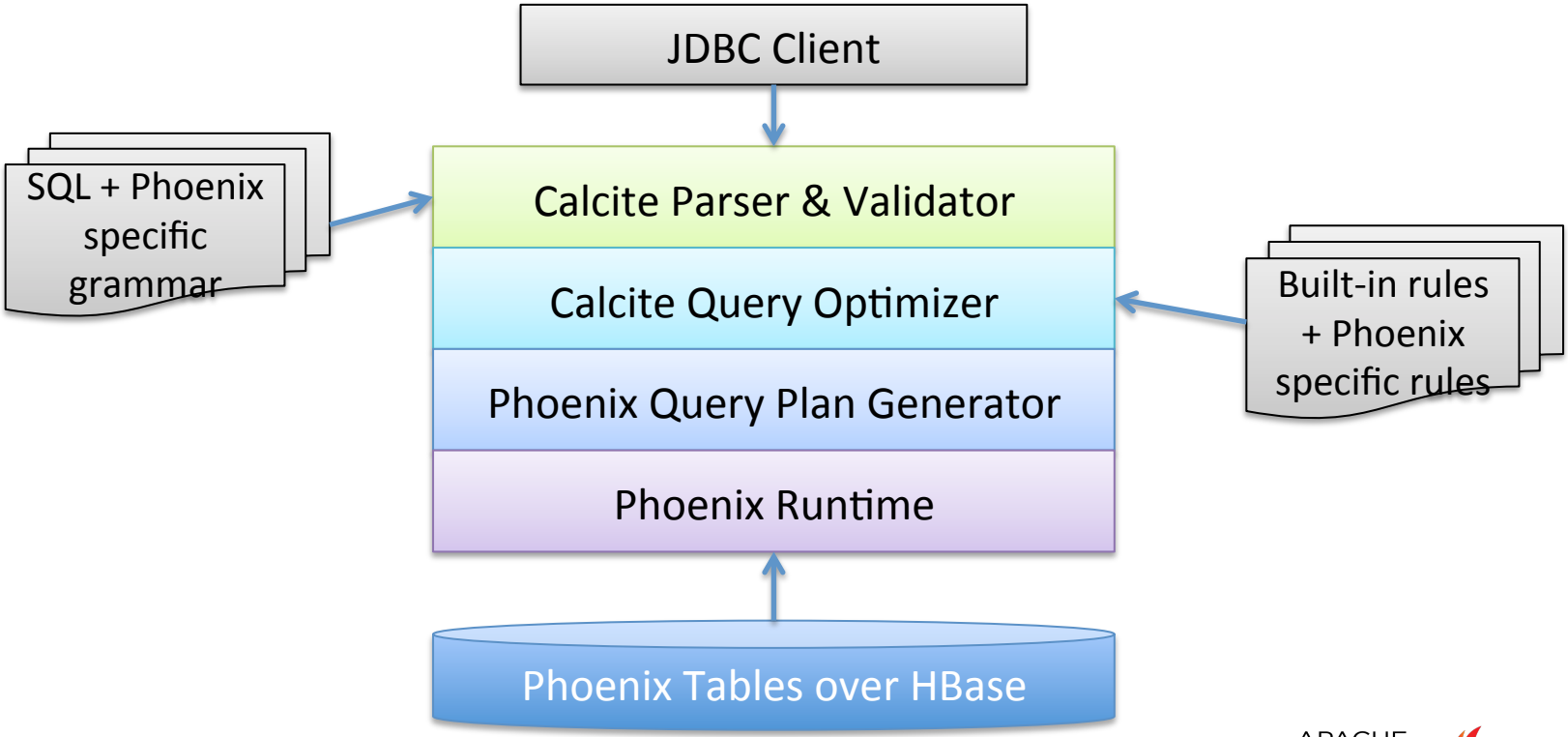
APACHE PHOENIX

Introducing Apache Calcite

- Query parser, compiler, and planner framework
  - SQL-92 compliant (ever argue SQL with Julian? :-))
  - Enables Phoenix to get missing SQL support
- Pluggable cost-based optimizer framework
  - Sane way to model **push down** through rules
- Interop with other Calcite adaptors
  - Not for free, but it becomes feasible
  - Already used by Drill, Hive, Kylin, Samza
  - Supports any JDBC source (i.e. RDBMS - remember them :-))
  - **One cost-model to rule them all**

APACHE PHOENIX

How does Phoenix plug in?



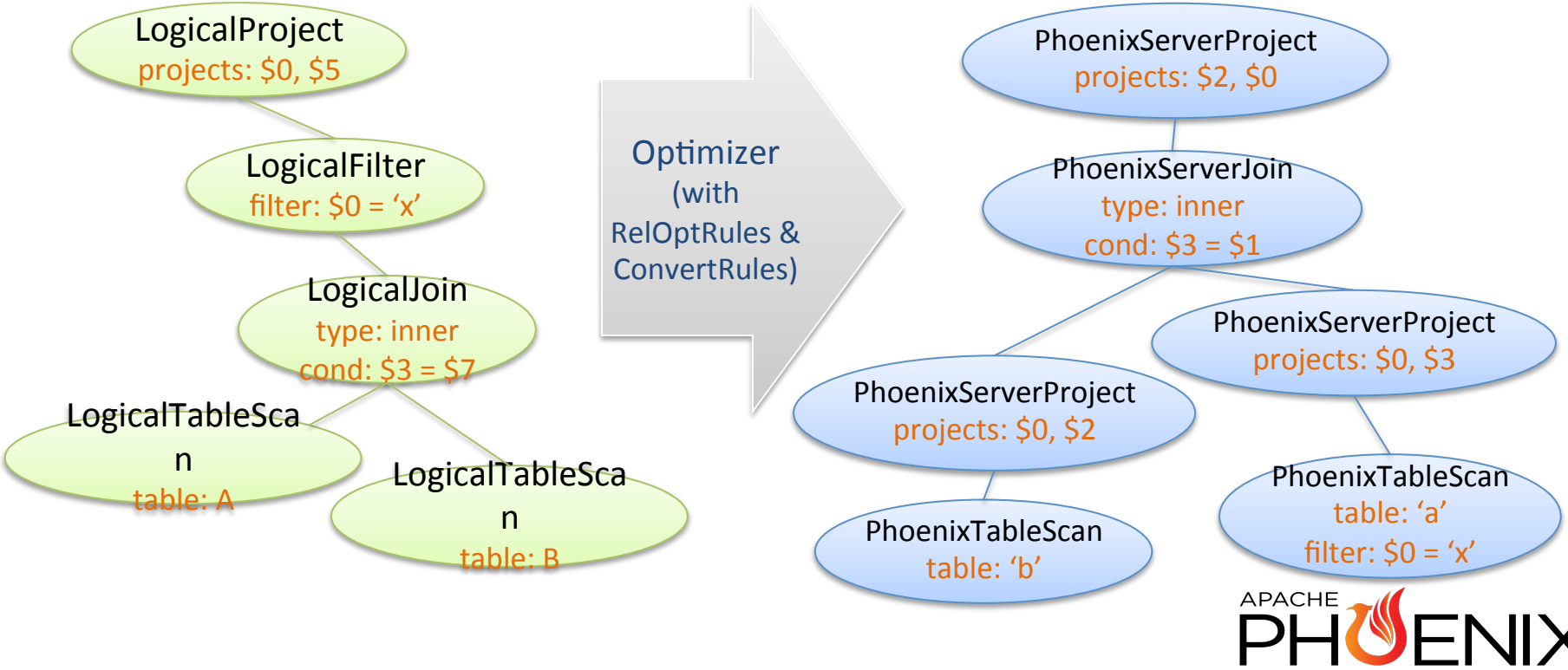
APACHE PHOENIX

Optimization Rules

- AggregateRemoveRule
- FilterAggregateTransposeRule
- FilterJoinRule
- FilterMergeRule
- JoinCommuteRule
- PhoenixFilterScanMergeRule
- PhoenixJoinSingleAggregateMergeRule
- …

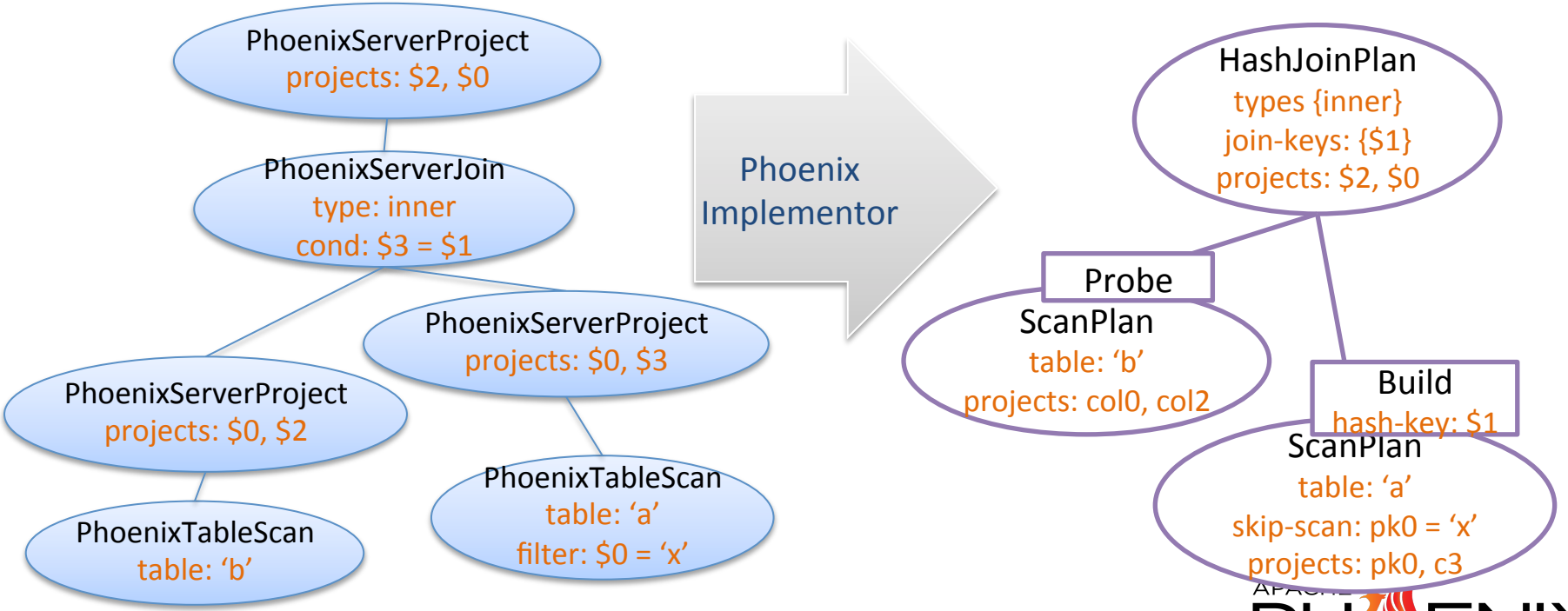
APACHE PHOENIX

Query Example
(filter push-down and smart join algorithm)



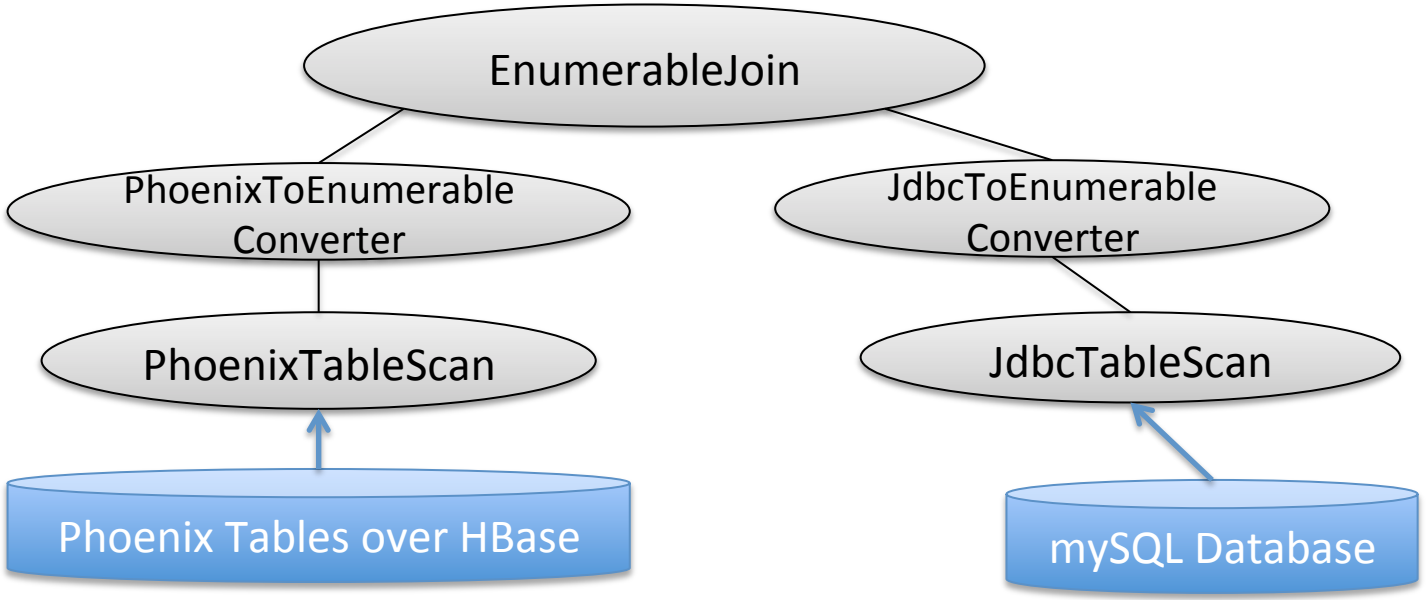
APACHE PHOENIX

Query Example
(filter push-down and smart join algorithm)



Interoperibility Example

- Joining data from Phoenix and mySQL



APACHE PHOENIX

Query Example 1

```
WITH m AS
   (SELECT *
    FROM dept_manager dm
    WHERE from_date =
       (SELECT max(from_date)
        FROM dept_manager dm2
        WHERE dm.dept_no = dm2.dept_no))
SELECT m.dept_no, d.dept_name, e.first_name, e.last_name
FROM employees e
JOIN m ON e.emp_no = m.emp_no
JOIN departments d ON d.dept_no = m.dept_no
ORDER BY d.dept_no;
```

APACHE PHOENIX

Query Example 2

```
SELECT dept_no, title, count(*)
FROM titles t
JOIN dept_emp de ON t.emp_no = de.emp_no
WHERE dept_no <= 'd006'
GROUP BY rollup(dept_no, title)
ORDER BY dept_no, title;
```

APACHE PHOENIX

Thank you!

Questions?

* who uses Phoenix

APACHE PHOENIX