Distributed SQL in TiDB

dongxu

PingCAP

About me

- Dongxu Huang, Cofounder & CTO @ PingCAP
- Geek / Engineer / Opensource enthusiast / Entrepreneur
- Go / Rust / Python / Clojure
- Build TiDB with <3

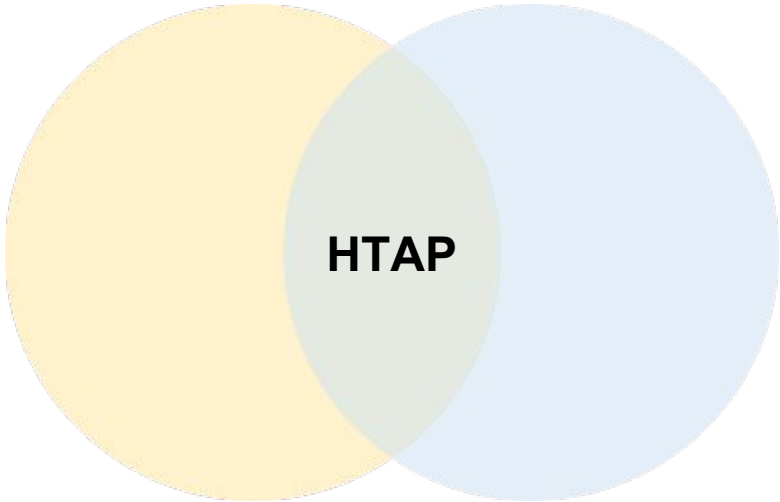
Rethinking SQL

- Data is growing at fast rate than ever before
  - The trending of AI / Data mining
  - Distributed systems become mainstream
- Traditional RDBMSs are no longer sufficient for many companies' needs
  - Scalability
- OLTP and OLAP are separate to each other
  - ETL is a pain in the ...
- **But SQL never dies**

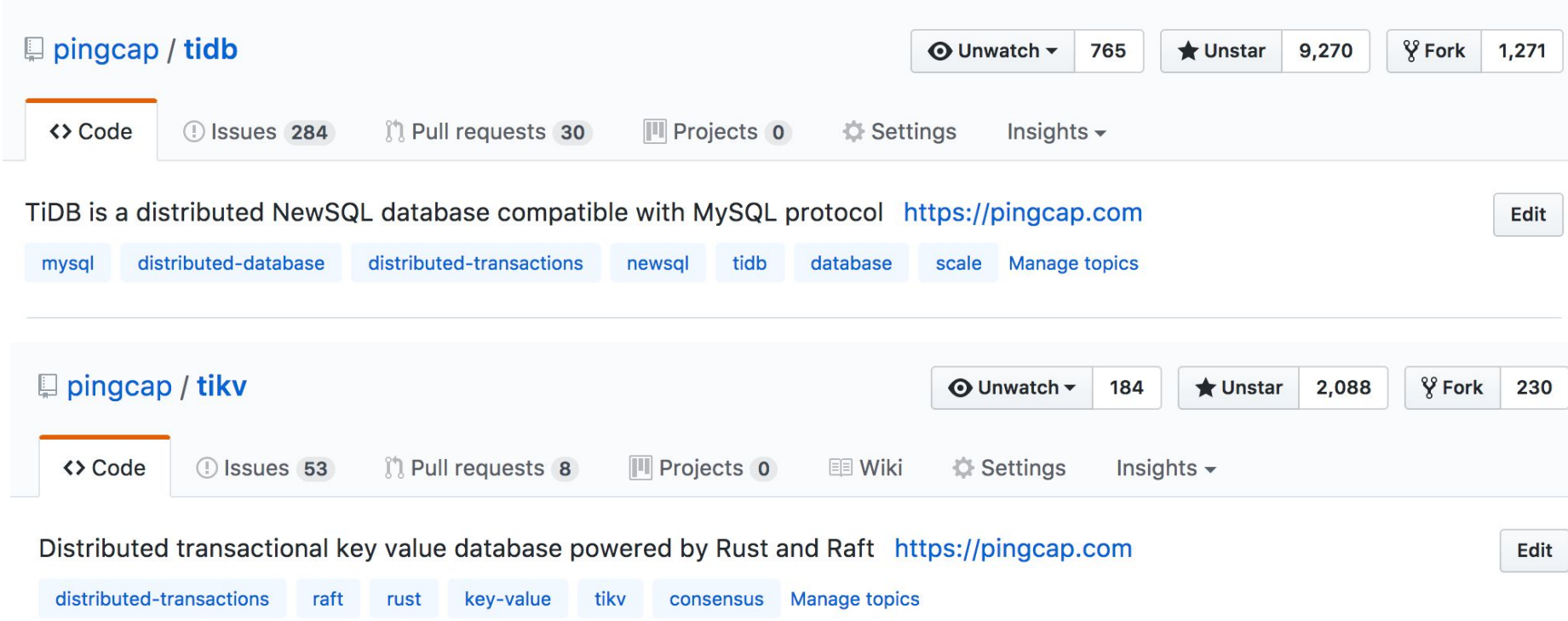
OLAP + OLTP = HTAP

Hybrid **T**ransactional / **A**nalytical **P**rocessing

- ACID Transaction
- Real-time analysis
- SQL



What's TiDB


pingcap / tidb
Unwatch 765
Unstar 9,270
Fork 1,271
Code
Issues 284
Pull requests 30
Projects 0
Settings
Insights
TiDB is a distributed NewSQL database compatible with MySQL protocol https://pingcap.com
Edit
mysql
distributed-database
distributed-transactions
newsql
tidb
database
scale
Manage topics
pingcap / tikv
Unwatch 184
Unstar 2,088
Fork 230
Code
Issues 53
Pull requests 8
Projects 0
Wiki
Settings
Insights
Distributed transactional key value database powered by Rust and Raft https://pingcap.com
Edit
distributed-transactions
raft
rust
key-value
tikv
consensus
Manage topics


盖娅互娱


心动网络
XD.COM


YOUZU.COM
游族网络


mobike


猿辅导


PANDA.TV


同程旅游


七牛云
QINIU

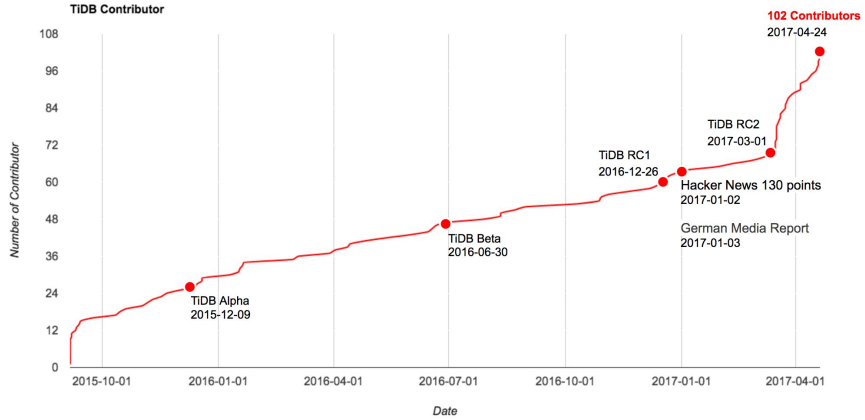

中国电信
CHINA TELECOM

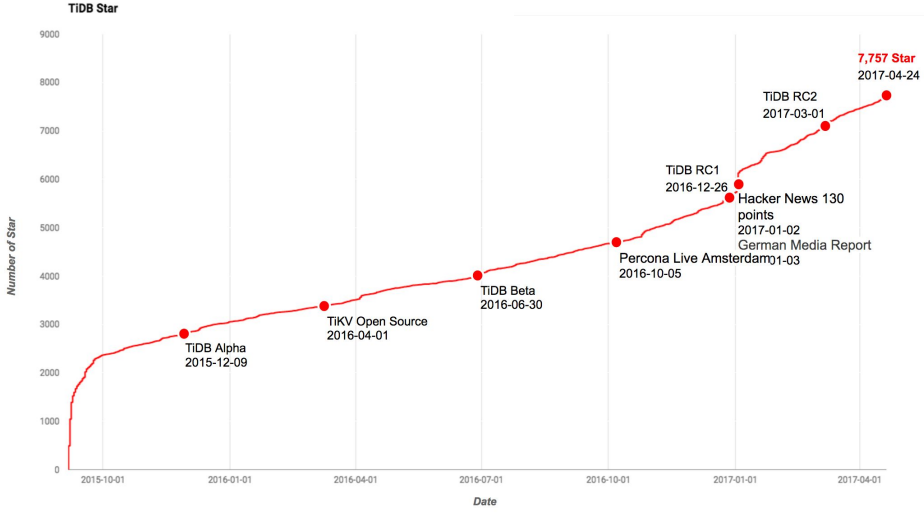

360金融
理财 | 贷款 | 保险

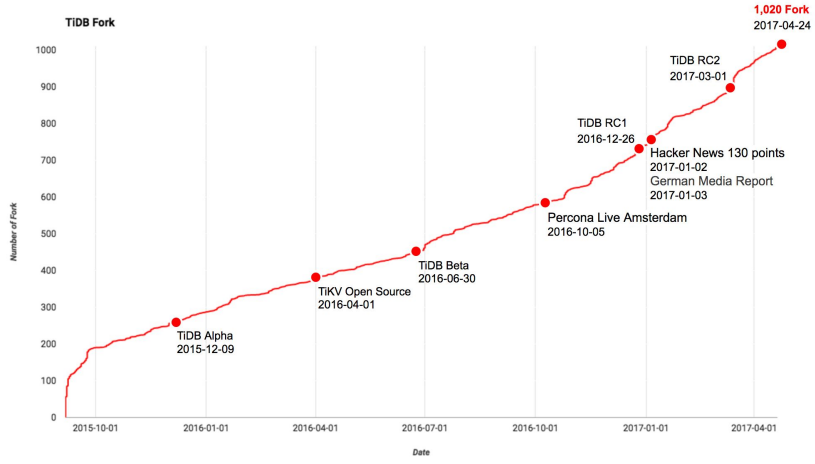

国联证券
GUOLIAN SECURITIES


元素征信
ELEMENTS CREDIT

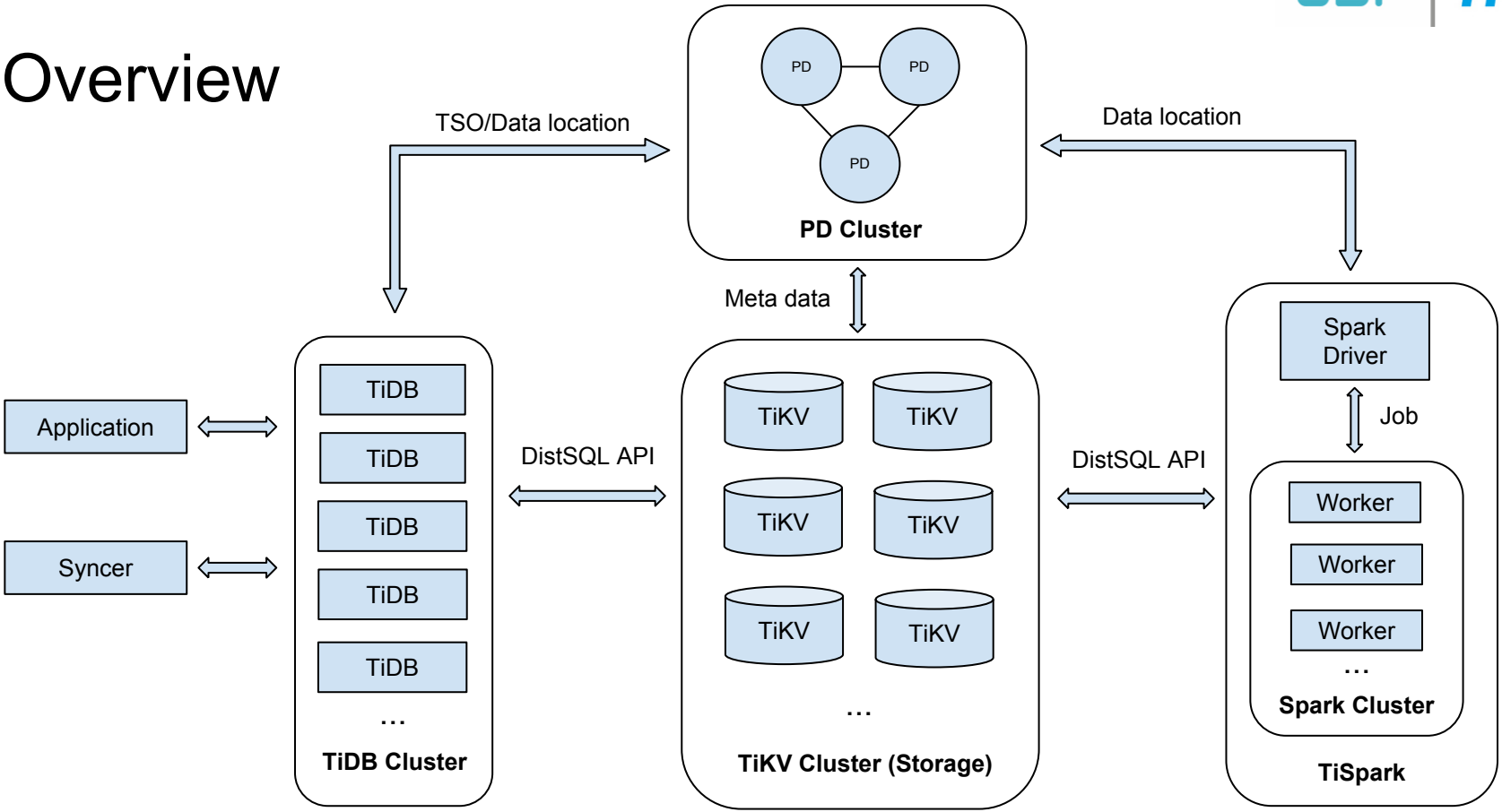
.....



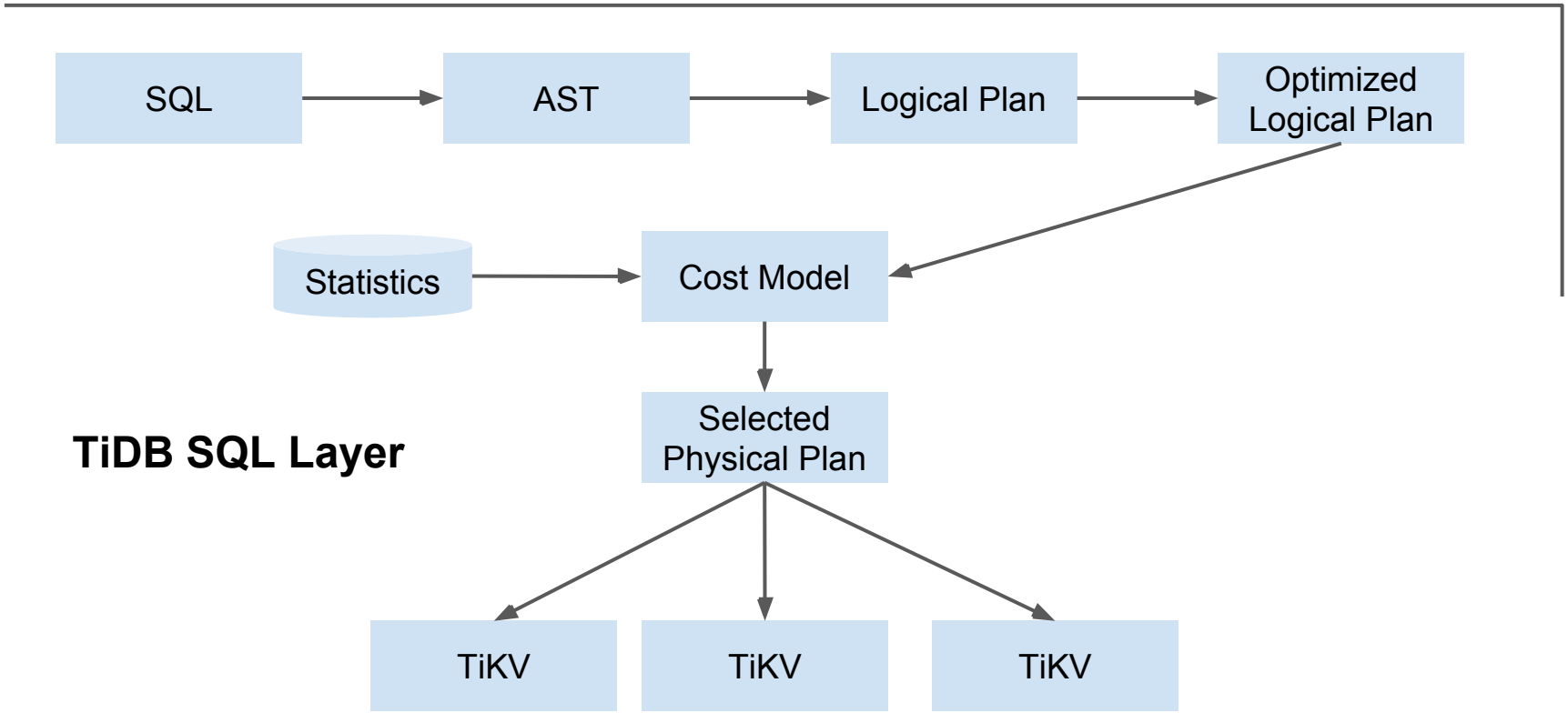




Overview


PD
PD
PD
PD Cluster
TSO/Data location
Data location
Meta data
Application
Syncer
TiDB
TiDB
TiDB
TiDB
TiDB
…
TiDB Cluster
DistSQL API
TiKV
TiKV
TiKV
TiKV
TiKV
TiKV
…
TiKV Cluster (Storage)
DistSQL API
Spark Driver
Job
Worker
Worker
Worker
…
Spark Cluster
TiSpark

Overview


SQL
AST
Logical Plan
Optimized Logical Plan
Statistics
Cost Model
Selected Physical Plan
TiDB SQL Layer
TiKV
TiKV
TiKV

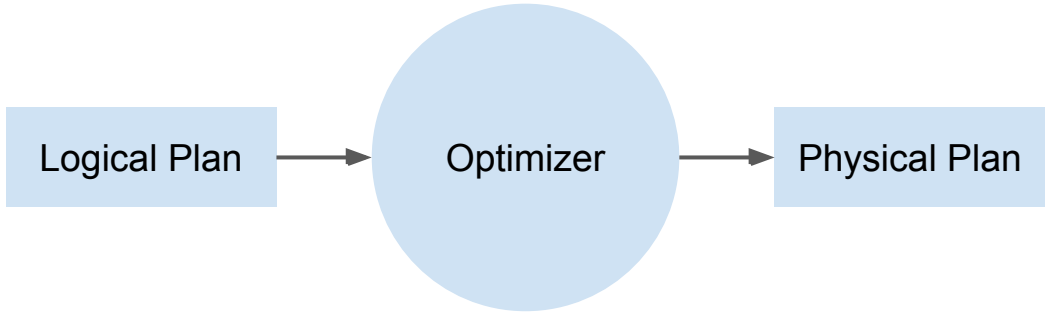
Lexer / Parser

- Yacc
  - goyacc
  - golex
- 100% homemade
  - Why not use MySQL's yacc file?
  - Pros and Cons?
- Nothing fancy...

```
SelectStmt:
    "SELECT" SelectStmtOpts SelectStmtFieldList Se
    {
        st := &ast.SelectStmt {
            SelectStmtOpts: $2.(*ast.SelectStmtOpts),
            Distinct:       $2.(*ast.SelectStmtOpts).Distinct,
            Fields:         $3.(*ast.FieldList),
            LockTp:         $5.(ast.SelectLockType),
        }
        lastField := st.Fields.Fields[len(st.Fields.Fields)-1]
        if lastField.Expr != nil && lastField.AsName.O == "" {
            src := parser.src
            var lastEnd int
            if $4 != nil {
                lastEnd = yyS[yypt-1].offset-1
            } else if $5 != ast.SelectLockNone {
                lastEnd = yyS[yypt].offset-1
            } else {
                lastEnd = len(src)
                if src[lastEnd-1] == ';' {
                    lastEnd--
                }
            }
            lastField.SetText(src[lastField.Offset:lastEnd])
        }
        if $4 != nil {
            st.Limit = $4.(*ast.Limit)
        }
        $$ = st
    }
```

Optimizer

- Logical plan
  - Predicate pushdown
  - Column pruning
  - Constant folding
  - DNF -> CNF
- Physical plan
  - Index selection
  - Join re-order



Optimizer

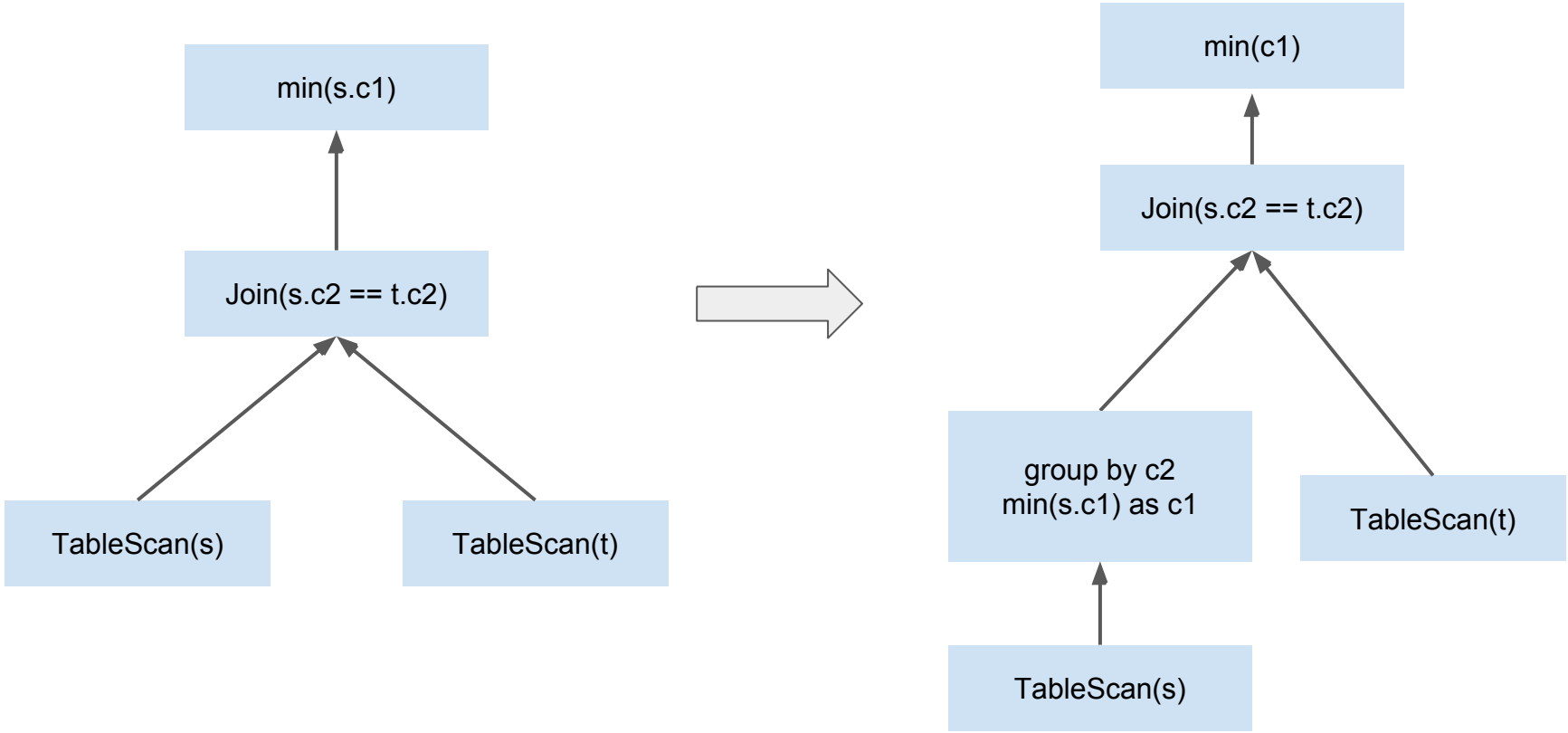
- Predicate Pushdown
- Column Pruning
- Eager Aggregate
- Convert Subquery to Join
- Statistics framework
- CBO Framework
  - Index Selection
  - Join Operator Selection
    - Hash join
    - Index lookup join
    - Sort-merge join
  - Stream Operators VS Hash Operators

Eager Aggregation

- Eager Aggregation and Lazy Aggregation - VLDB Endowment
- Example:

```
SELECT MIN(s.c1) FROM s JOIN t ON s.c2 = t.c2
```

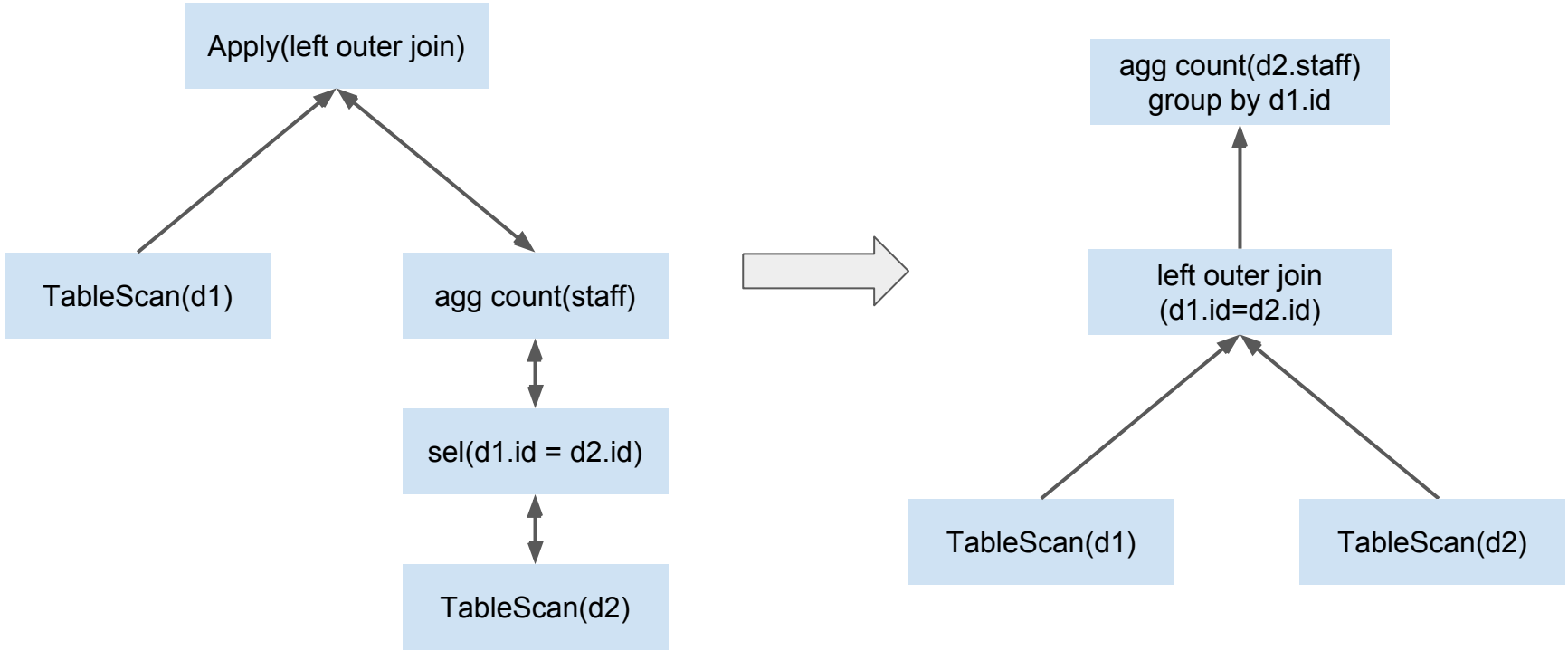
SELECT MIN(s.c1) FROM s JOIN t ON s.c2 = t.c2



Convert Subquery to Join

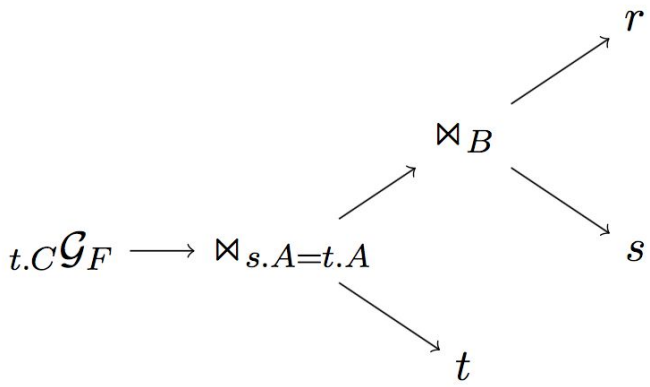
- Paper: Orthogonal Optimization of Subqueries and Aggregation (SIGMOD 2001)
- Most queries contain subquery could be converted to join
- Example:

```
SELECT * FROM depart as d1 WHERE 3 = (SELECT count(people) FROM depart as d2 where d1.id = d2.id);
```

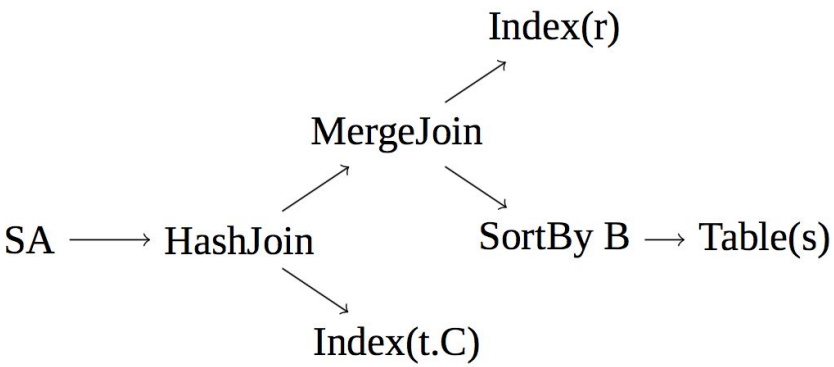

Apply(left outer join)
TableScan(d1)
agg count(staff)
sel(d1.id = d2.id)
TableScan(d2)
agg count(d2.staff)
group by d1.id
left outer join
(d1.id=d2.id)
TableScan(d1)
TableScan(d2)

CBO Framework

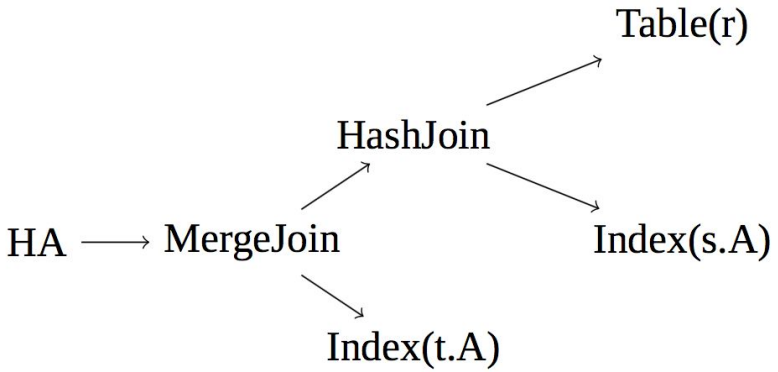
Imagine we got a logical plan:



its physical plan could be:



or:



Cost estimation

$$Cost(p) = N(p) * F_N + M(p) * F_M + C(p) * F_C$$

Network cost **Memory cost** **CPU cost**

In TiDB, default memory factor is 5 and cpu factor is 0.8.
For example: Operator Sort(r), its cost would be:

$$0 + n_r * 5.0 + n_r * log_2(n_r) * 0.8$$

CBO Framework

- Access path selection in a relational database management system - 1979 IBM
- DP(Dynamic Programming) on tree based on statistic infomation

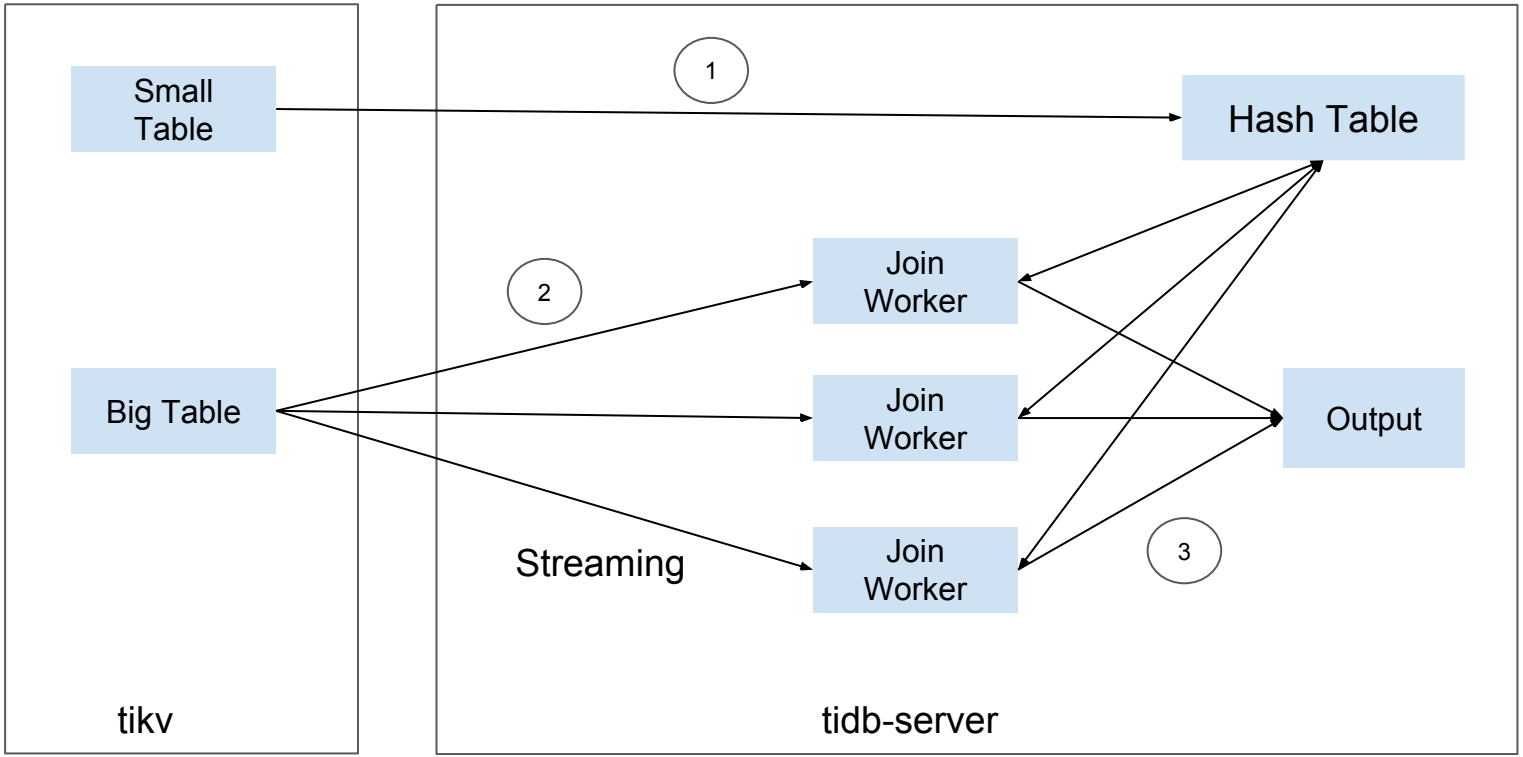
$$\text{let } S_0 = \{c_0, c_1 \dots c_n\}$$

$$\mathrm{C}(\mathrm{lp}_0, S_0) = min \begin{cases} C(lp_1, S_0) + cost(pp_j) & pp_j \text{ won't change anything} \\ \dots & \\ C(lp_1, S_1) + cost(pp_i) + cost(Sort) & pp_i \text{ return order } S_1 \end{cases}$$

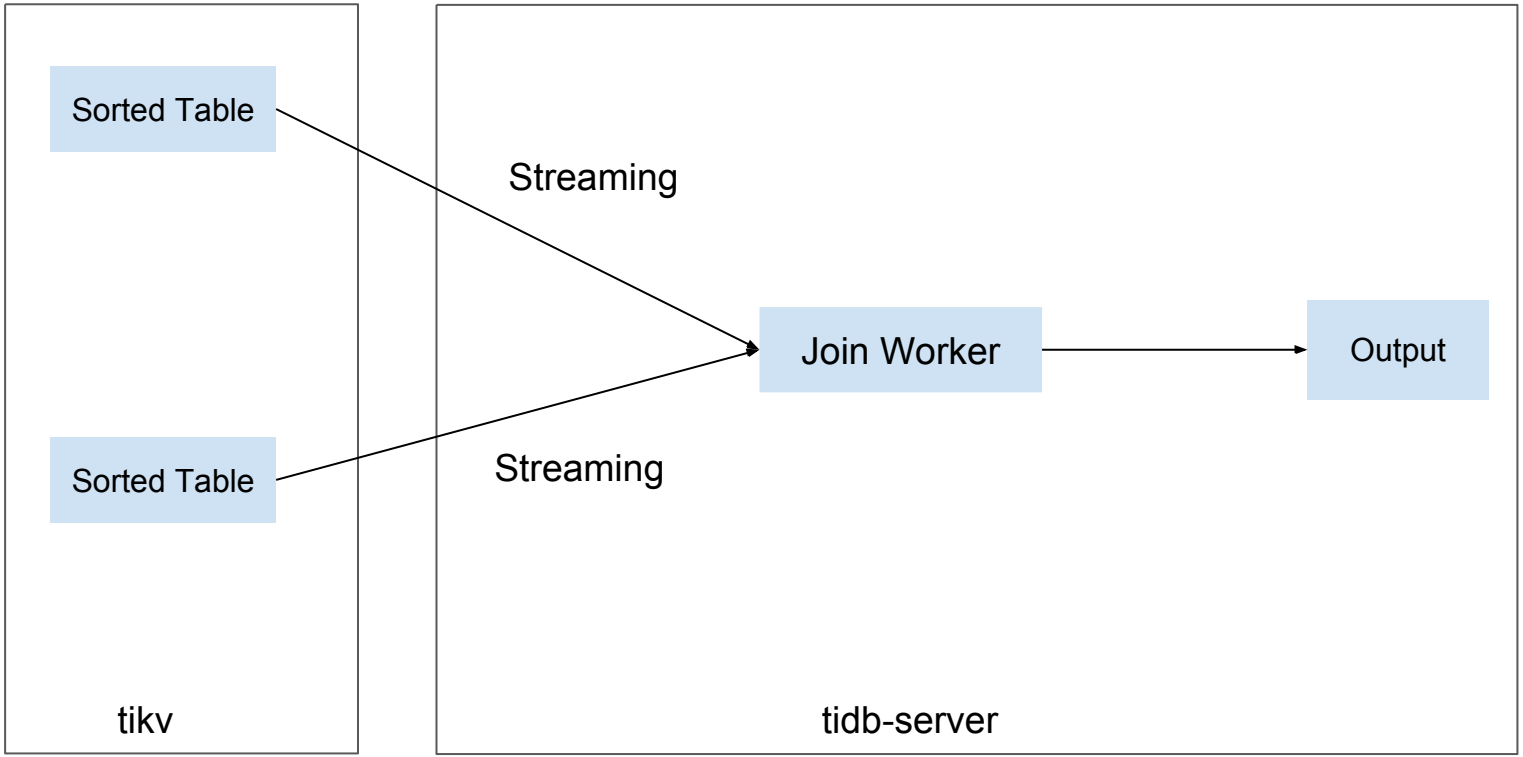
Parallel Join

- Hash join
  - Fastest, joined tables are not very large, <= 50M rows, works with/without index.
- Sort merged join
  - Memory-free, must join on indexed column (or ordered data source)
- Index lookup join
  - must join on indexed column with high selectivity (filtered result set should be less than 10000 rows)

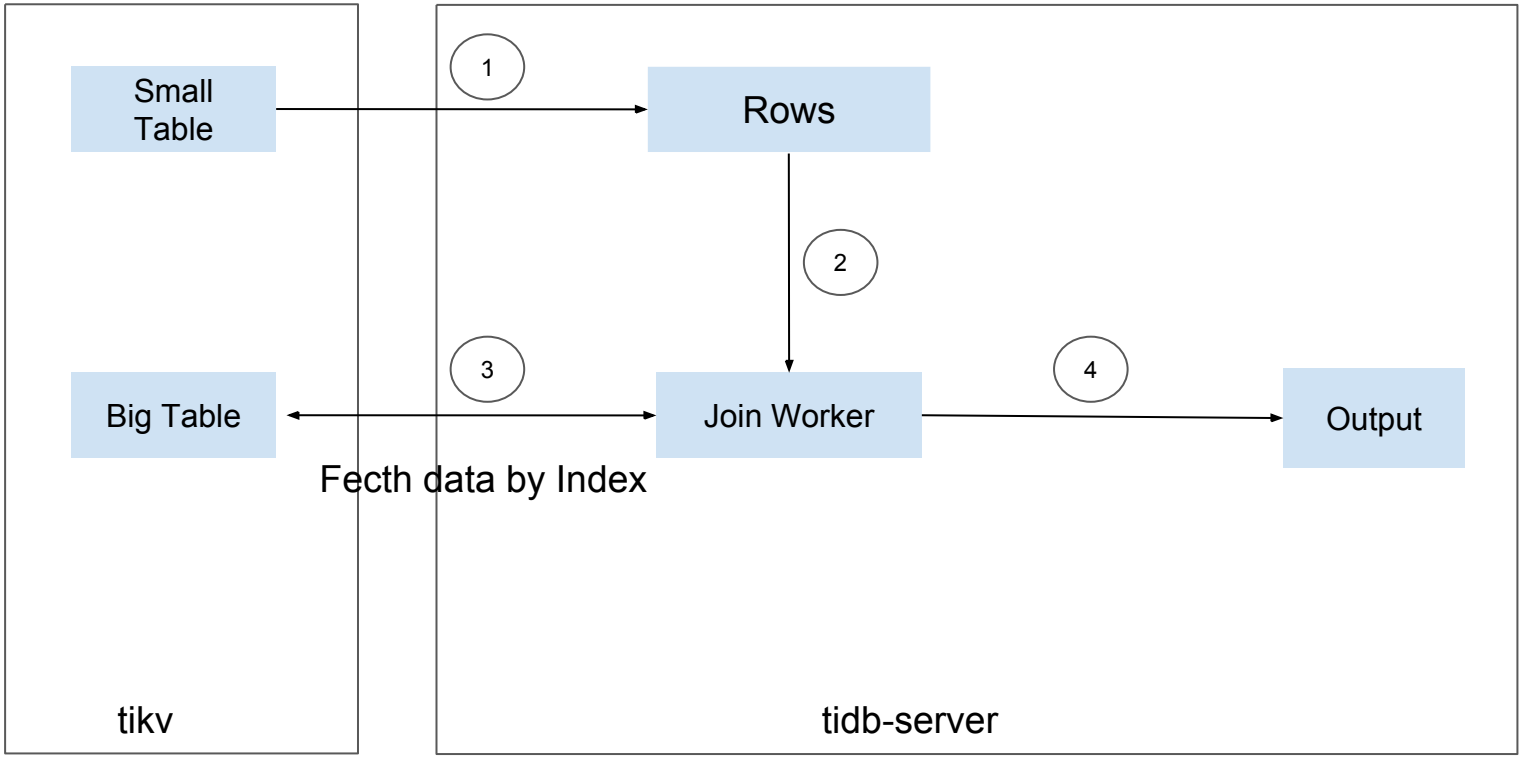
Hash join


Small Table
1
Hash Table
2
Join Worker
Big Table
Join Worker
Output
Streaming
Join Worker
3
tikv
tidb-server

Sort-merge Join



Index Lookup Join



Statistics

- Equi-depth Histograms
- Max 256 Buckets
- NDV
  - Efficient and Scalable Statistics Gathering for Large Databases in Oracle 11g
- Full table analyze
  - Data sampling
  - Row && Index
- Incrementally analyze
- Distributed analyze (TODO)
  - Pushdown analyze jobs to TiKV coprocessor
  - Real-time analyze

Statistics

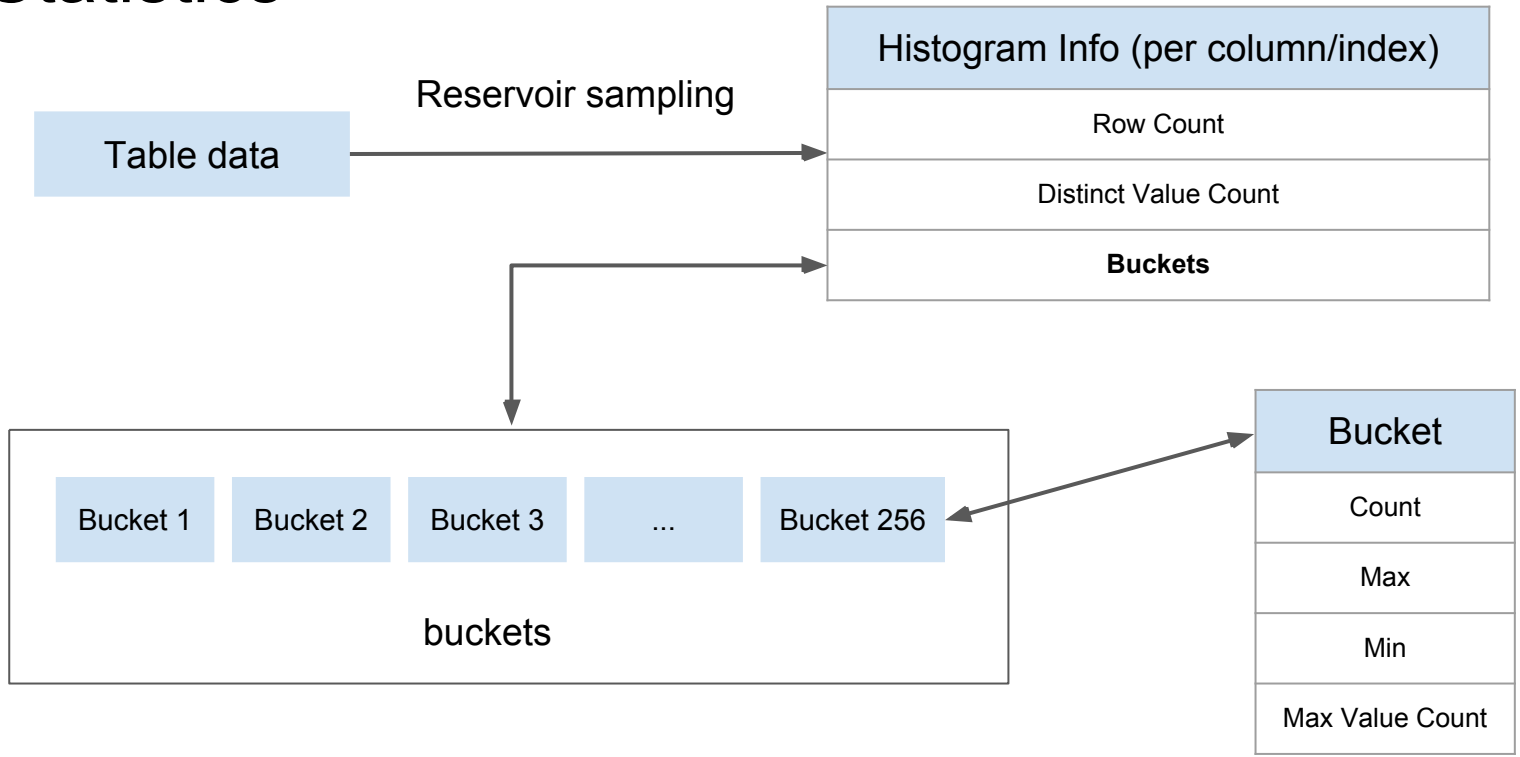
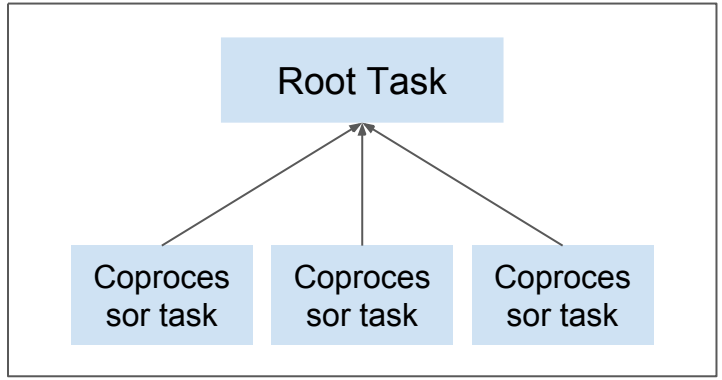
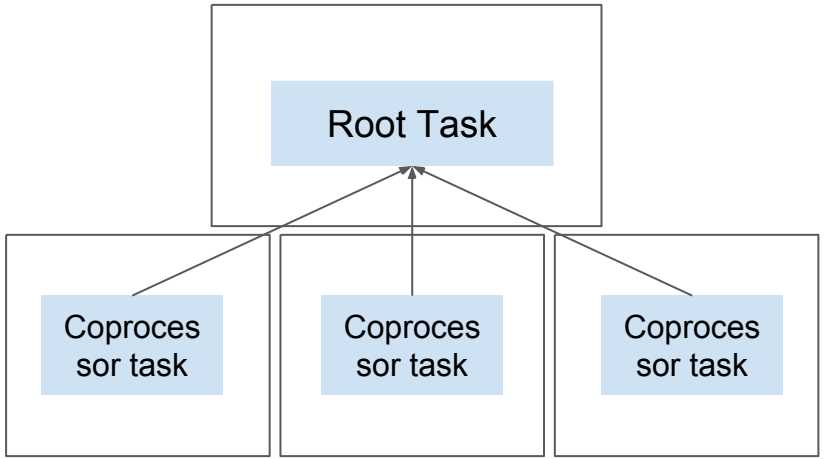


Table data
Reservoir sampling
Histogram Info (per column/index)
Row Count
Distinct Value Count
Buckets
Bucket 1
Bucket 2
Bucket 3
...
Bucket 256
buckets
Bucket
Count
Max
Min
Max Value Count

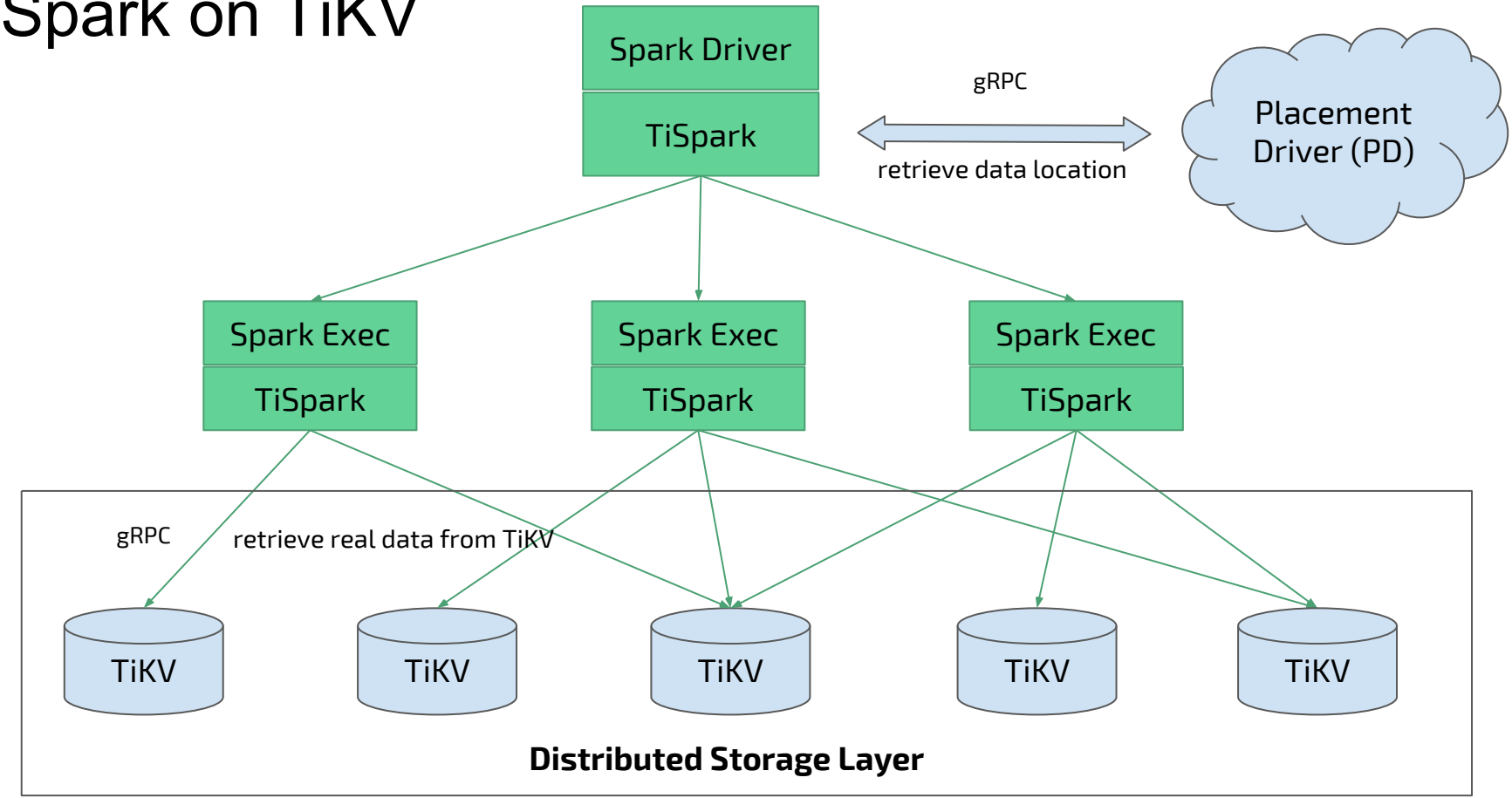
Executor

- Coprocessor TiDB and TiKV
- Task
  - Root Task
  - Coprocessor Task

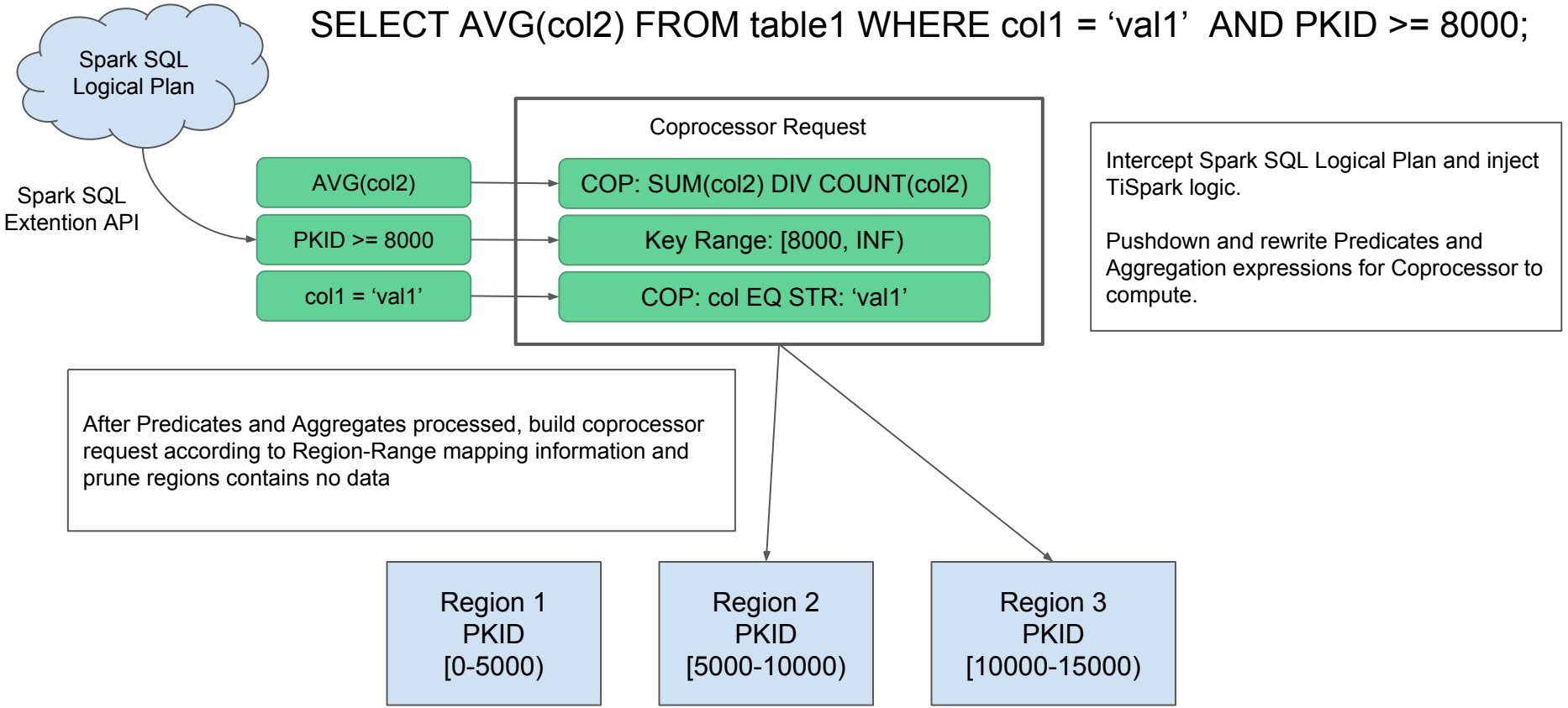




Spark on TiKV

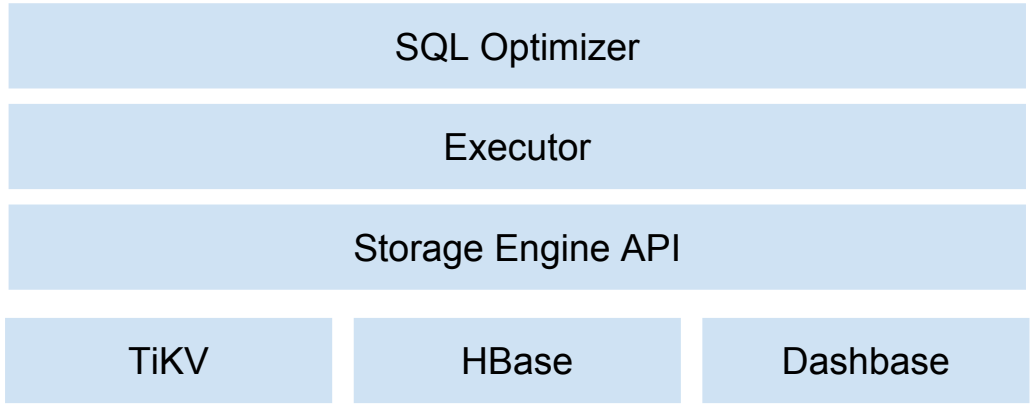


Spark on TiKV



Pluggable Storage Engine

- Standalone
  - goleveldb
  - boltdb
  - mem
- TiKV
- 3rd party
  - hbase
  - dashbase



Future works

- Code Generation
- MPP Engine
- Mixed storage engine (Columnar / Row-based)
- ...

Q&A

Thank You