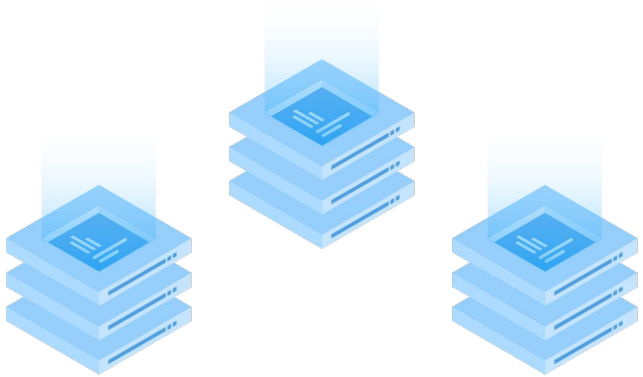
TiDB Compiler 源码阅读

wangcong@pingcap.com

PingCAP TiDB



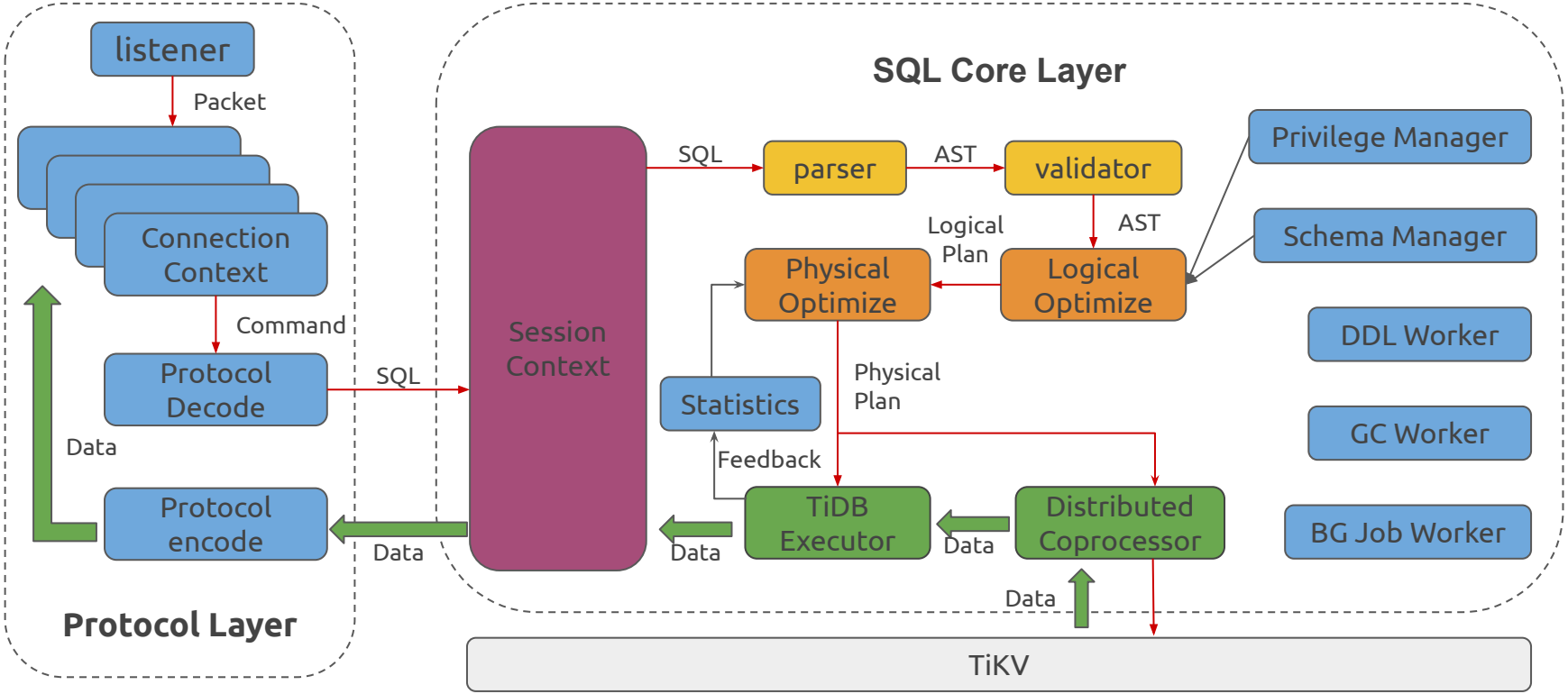
Agenda

1. Overview
2. Preprocess
3. Optimizer
4. Cascades Planner
5. Summary

PingCAP

Part I - Overview

TiDB SQL Layer



Compiler Overview

```
// Compiler compiles an ast.StmtNode to a physical plan.
type Compiler struct {
    Ctx sessionctx.Context
}
```

- (From parser) AST -> LogicalPlan -> PhysicalPlan (To executor)
- Validation
- Optimize

PingCAP

Part II - Preprocess

Preprocess

- Resolve semantics
- ast.Node interface
  - Accept
- Visitor interface
  - Enter
  - Leave

PingCAP

Part III - Optimizer

Optmizer

- PlanBuilder
- Logical Optimize
- Physical Optimize
- (Post Optimize)

Optmizer - PlanBuilder

- Build logical plan tree bottom-up
- Add flags for later logical optimization
- An important Visitor: expressionRewriter
  - Convert AST node to Expression (e.g, resolve column names)
  - Convert correlated subquery expression to LogicalApply
  - Unfold incorrelate subquery
  - FoldConstant
  - Eliminate ifnull(pk, 0, 1)
  - Convert `IN (incorrelate subquery)` to inner join on Distinct
  - ...

Optmizer - LogicalOptimize

- Apply applicable rules in order:
  - Prune columns
  - Build key info
  - Decorrelate subquery
  - Eliminate aggregation
  - Eliminate projection
  - Eliminate Max / Min
  - Predicate push down
  - Eliminate outer join
  - Join reorder
  - ...

Optmizer - PhysicalOptimize

- task interface
  - copTask / rootTask
- TableScanReader / IndexScanReader / IndexLookUpReader
- PhysicalProperty
  - order property
  - task type property
  - expected count property

Optmizer - PhysicalOptimize

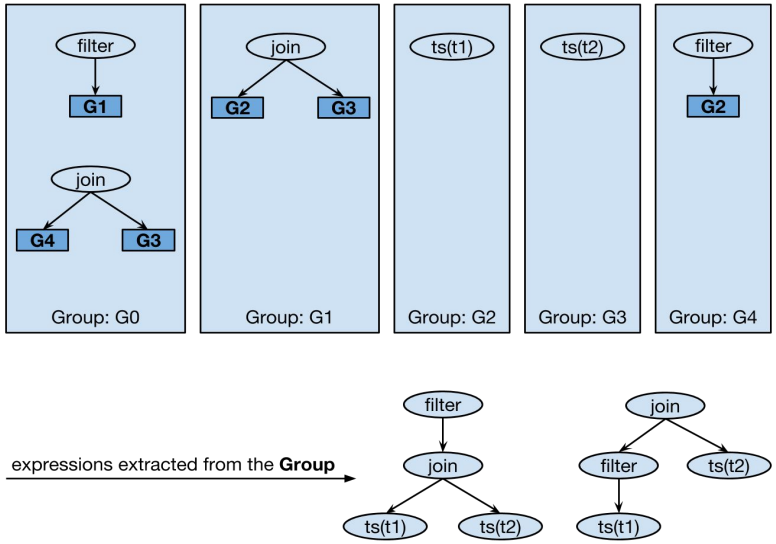
- Pull up statistics
  - ranger
- Pull up possible order properties
- findBestTask
  - cost model
  - memorization
  - required child physical property
  - enforced merge join by TIDB_SMJ hint
  - aggregation / limit / top-n pushdown
  - boundary of cop / root task
  - impose expected count
  - one special case: index join

PingCAP

Part IV - Cascades Planner

Group/Group Expression

- A group is a set of logically equivalent logical and physical expressions that produce the same output.
- Group expression represents an expression in the expression group. The child of group expression is expression group.



Optimize

- Split into two phases:
  - Exploration
  - Implementation

Exploration - Explore Group

- For each equivalent group expression
  - explore the children groups
  - transform the current group expression
  - delete the current group expression if we can

- For each possible tranformation rule
  - For each possible matched expression
    - transform and insert into group

Implementation - Impl Group

- For each group expr
  - for each possible implementations
    - impl it’s children according to the req
    - caclualate the cost

- For all the possible implementation rules
  - Check if match the physical property
  - Implement it if matched

Implementation - Enforce

- For each possible enforcer rules
  - get the new property
  - implement the group
  - add the force operator

PingCAP

Thanks!

Wang Cong | wangcong@pingcap.com