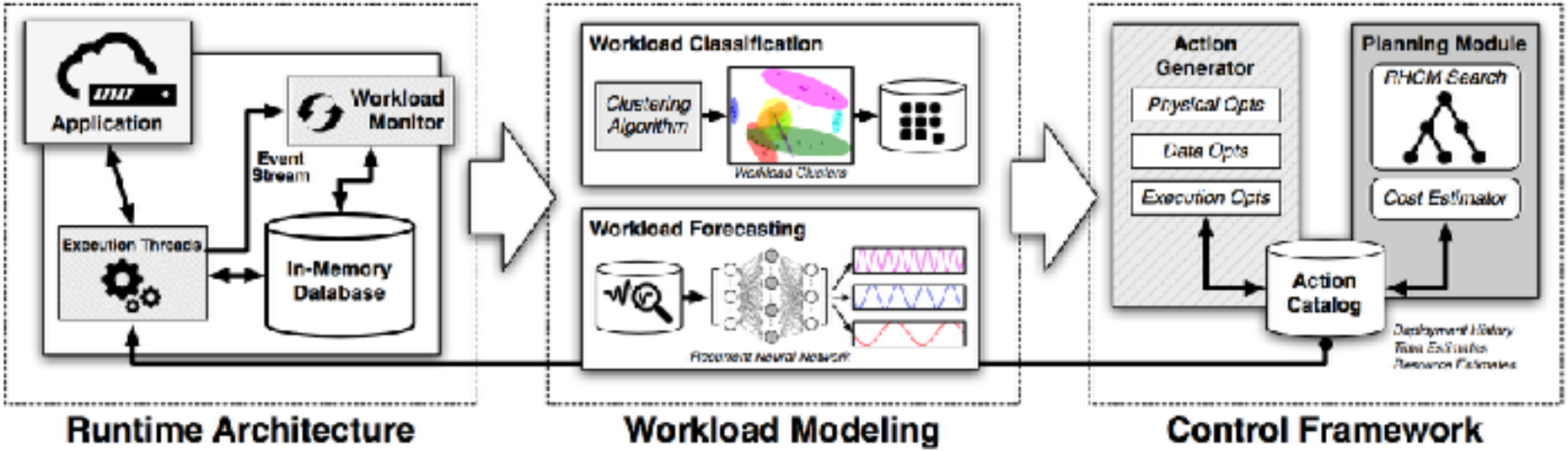
How to build a *Self-Driving* database
An overview

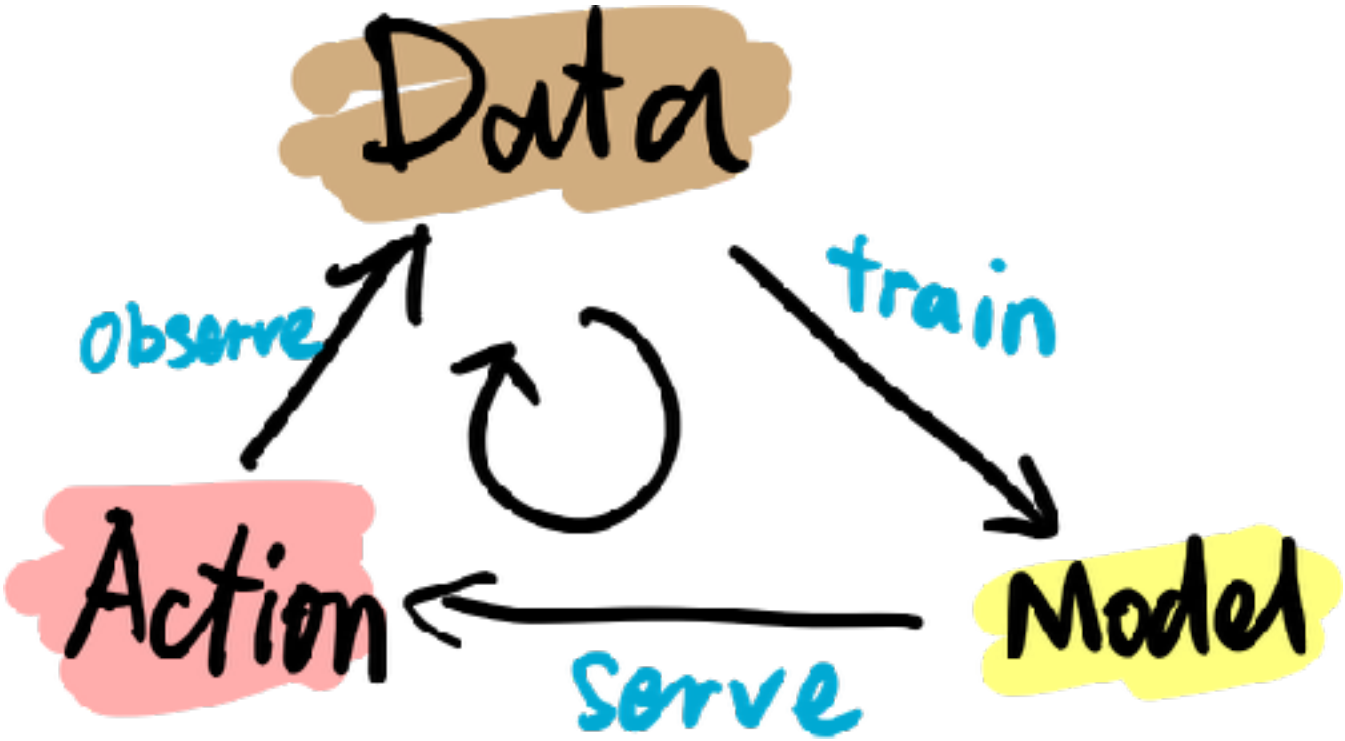
Dongxu

Who am I

- Dongxu Huang
- CTO & Cofounder, PingCAP
- Distributed system engineer / Open source advocator
- h@pingcap.com
- TiDB / TiKV / TiSpark

Self-Driving?


Application
Workload Monitor
Event Stream
Execution Threads
In-Memory Database
Workload Classification
Clustering Algorithm
Workload Clusters
Workload Forecasting
Recurrent Neural Network
Action Generator
Physical Opts
Data Opts
Execution Opts
Planning Module
RHCM Search
Cost Estimator
Action Catalog
Deployment History
Runtime Architecture
Workload Modeling
Control Framework

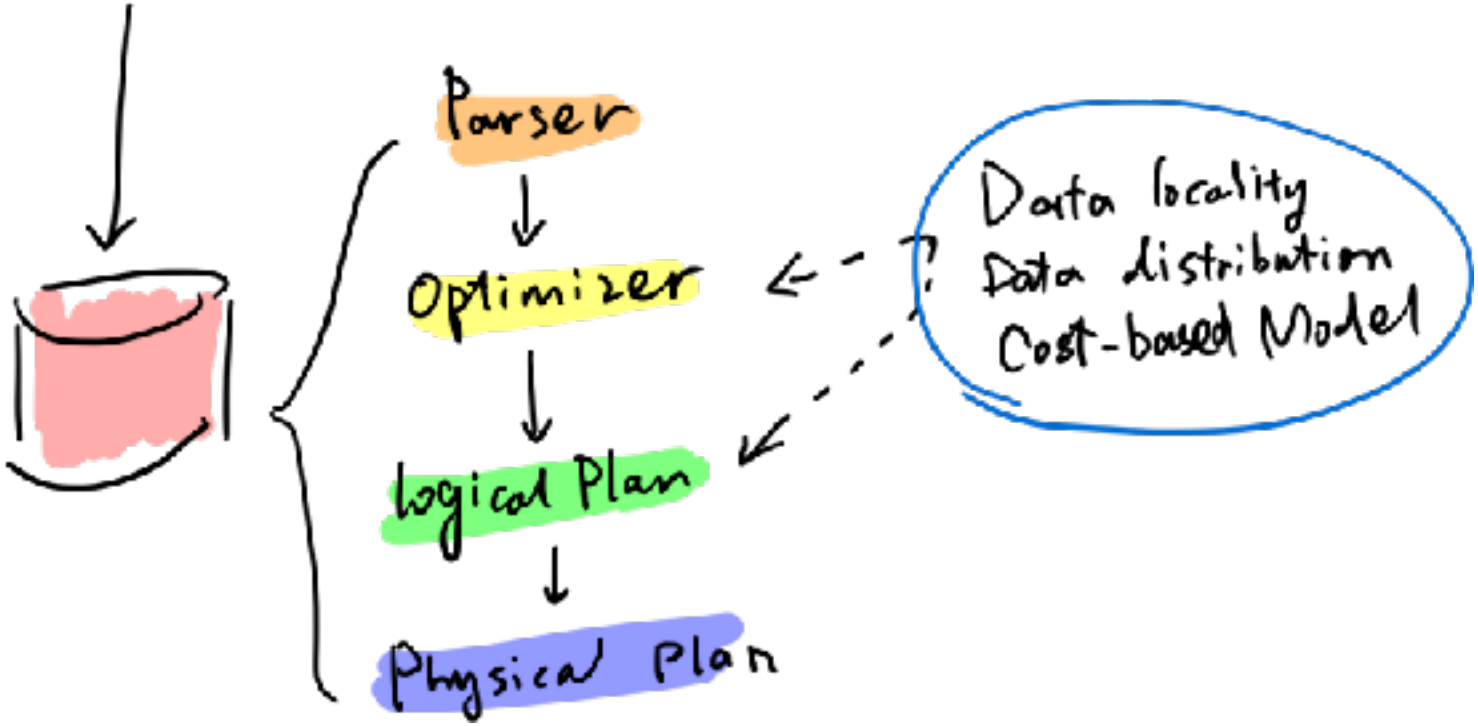

Data
train
Model
Serve
Action
Observe

Why now?

- Better hardware
- Better tools
- Workload is different

SQL Tuning

select * from t where id>10 and id<100 and name = 'tom';

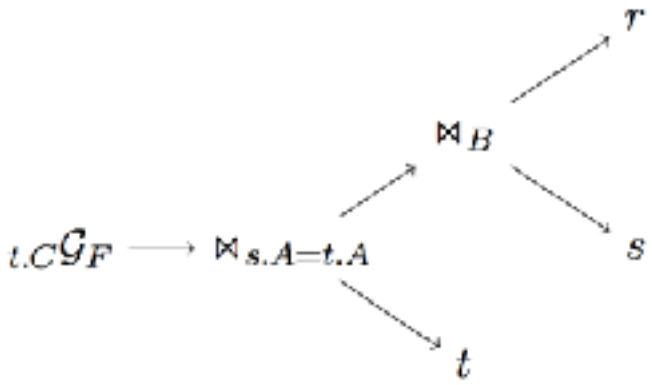


Data : Rows / Indexs distribution

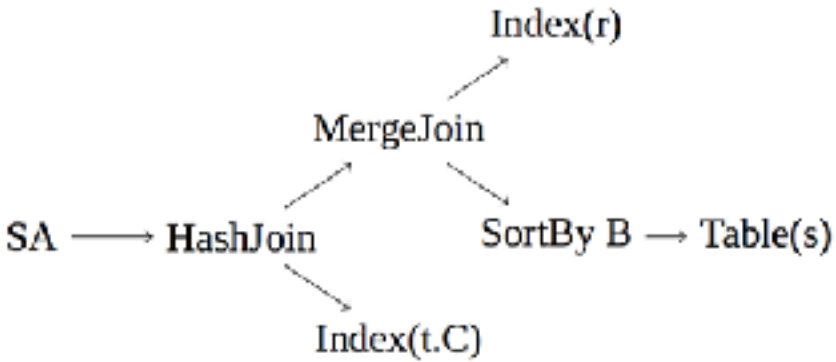
Model : Cost model

Action : Choose a better Physical Plan

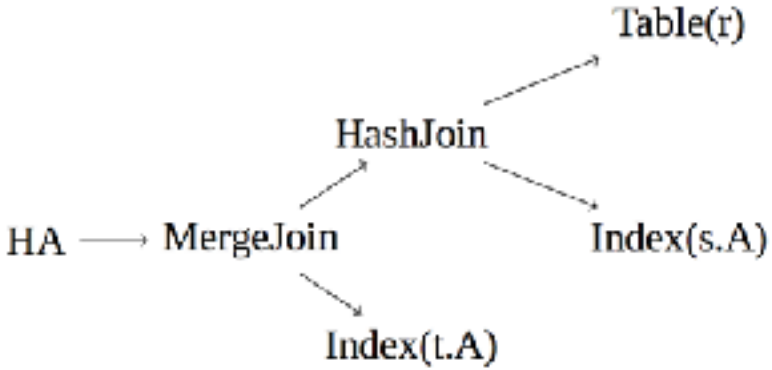
Imagine we got a logical plan:



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**

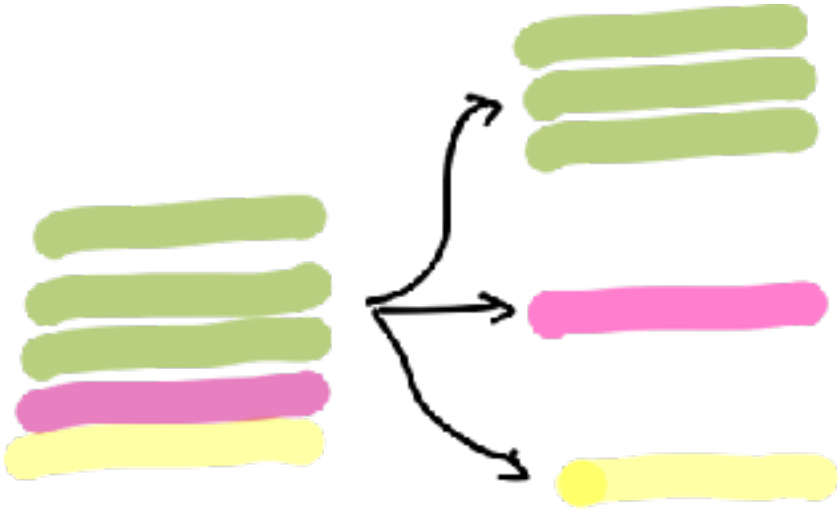
In TiDB, the default memory factor is 5 and CPU factor is 0.8.
For example: Operator Sort(r), its cost would be:

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

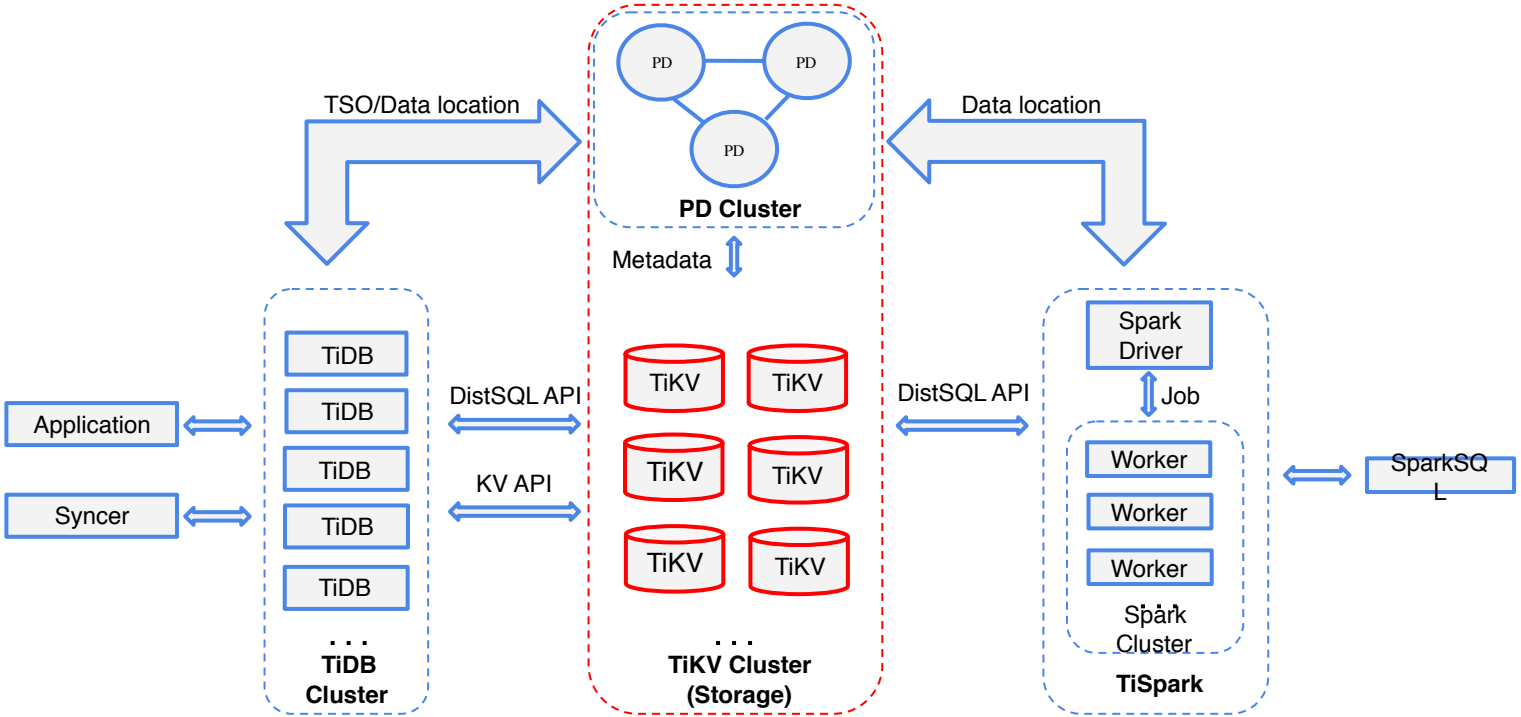
Data placement

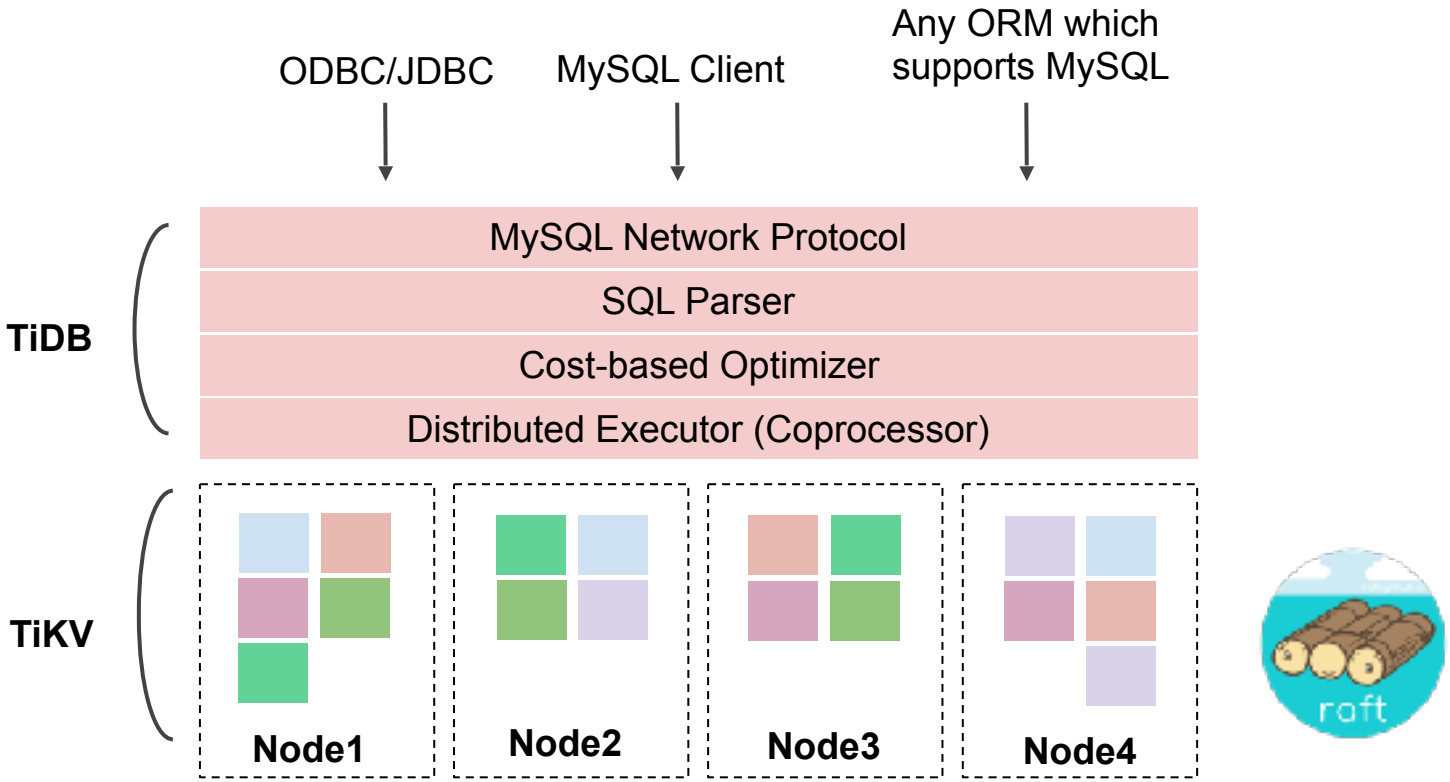
Problem: Uneven Data Distribution

- Dealing with hotspot
  - Choose a wrong sharding key
- Inefficient usage
  - Some are busy
  - Some are idle
- Caused by the nature of RDBMS



A little bit about how data is organized in TiDB




ODBC/JDBC
MySQL Client
Any ORM which supports MySQL
TiDB
MySQL Network Protocol
SQL Parser
Cost-based Optimizer
Distributed Executor (Coprocessor)
TiKV
Node1
Node2
Node3
Node4
raft

Why Raft?

- Saft split/merge
- Self-healing
- Easy to implement

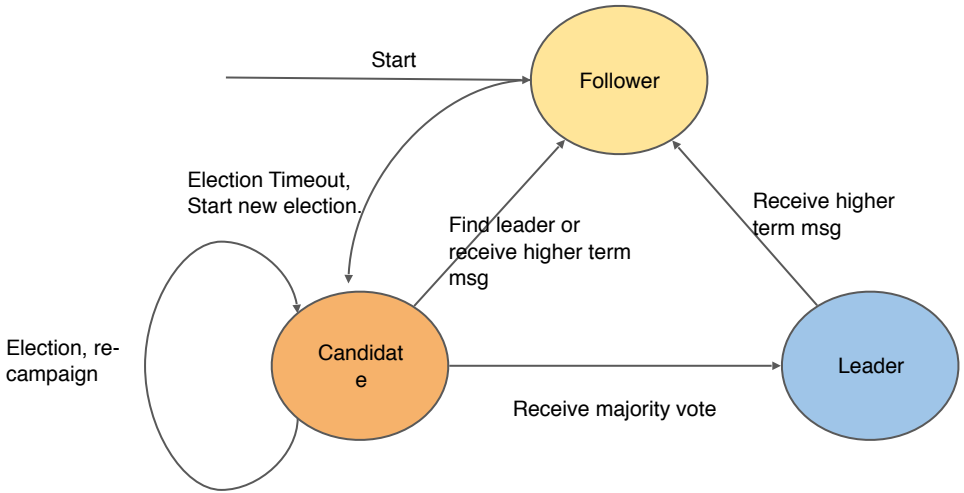
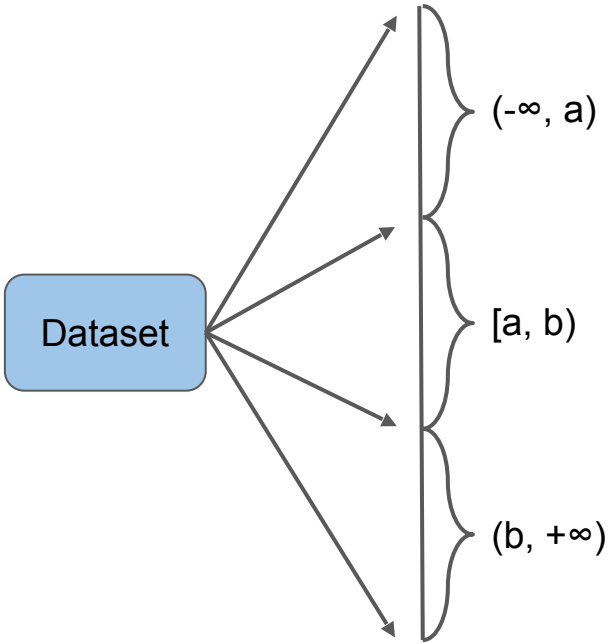
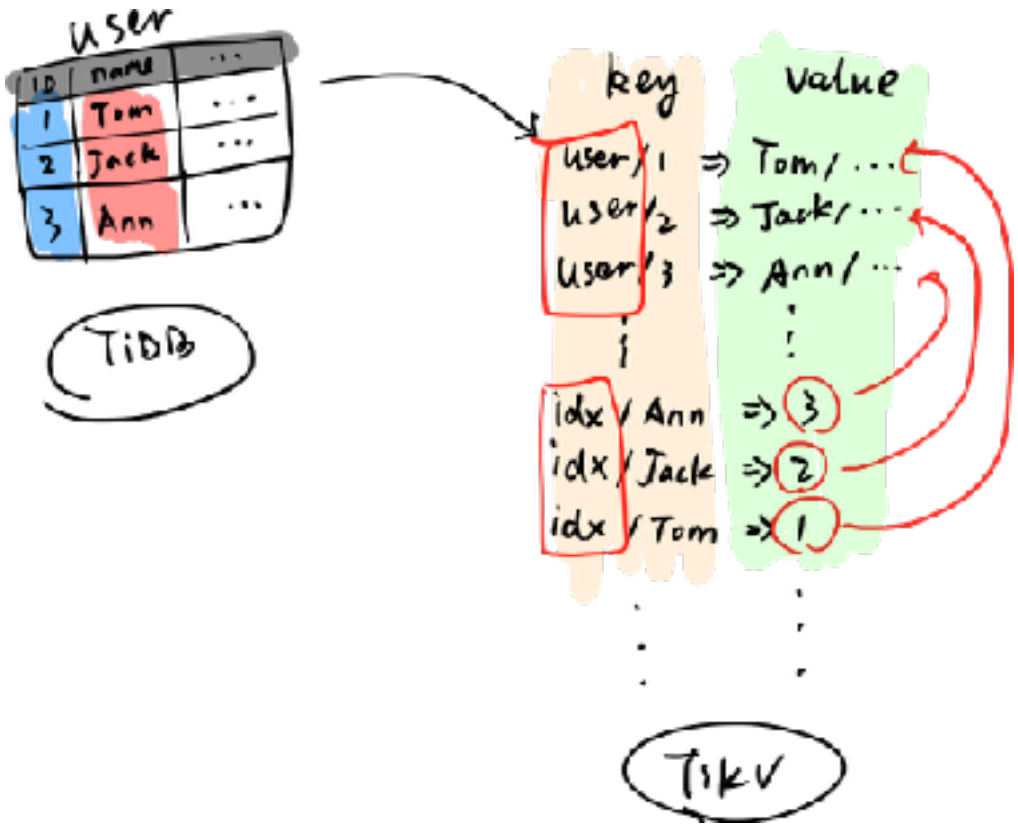
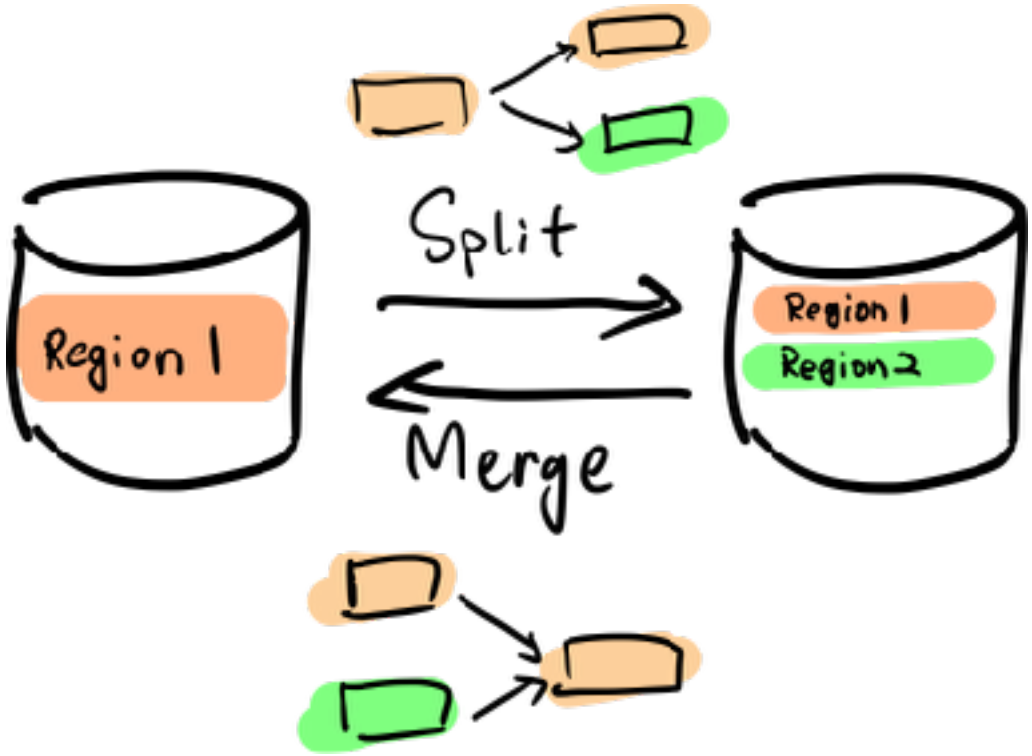


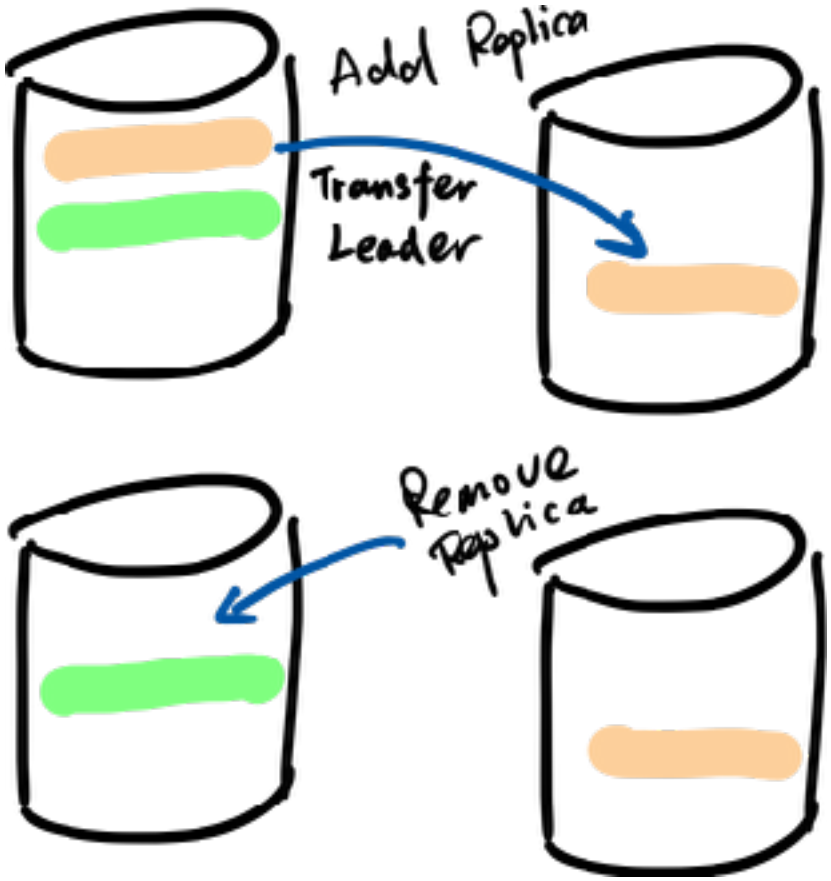
Table mapping: Rows => Key-Value pairs





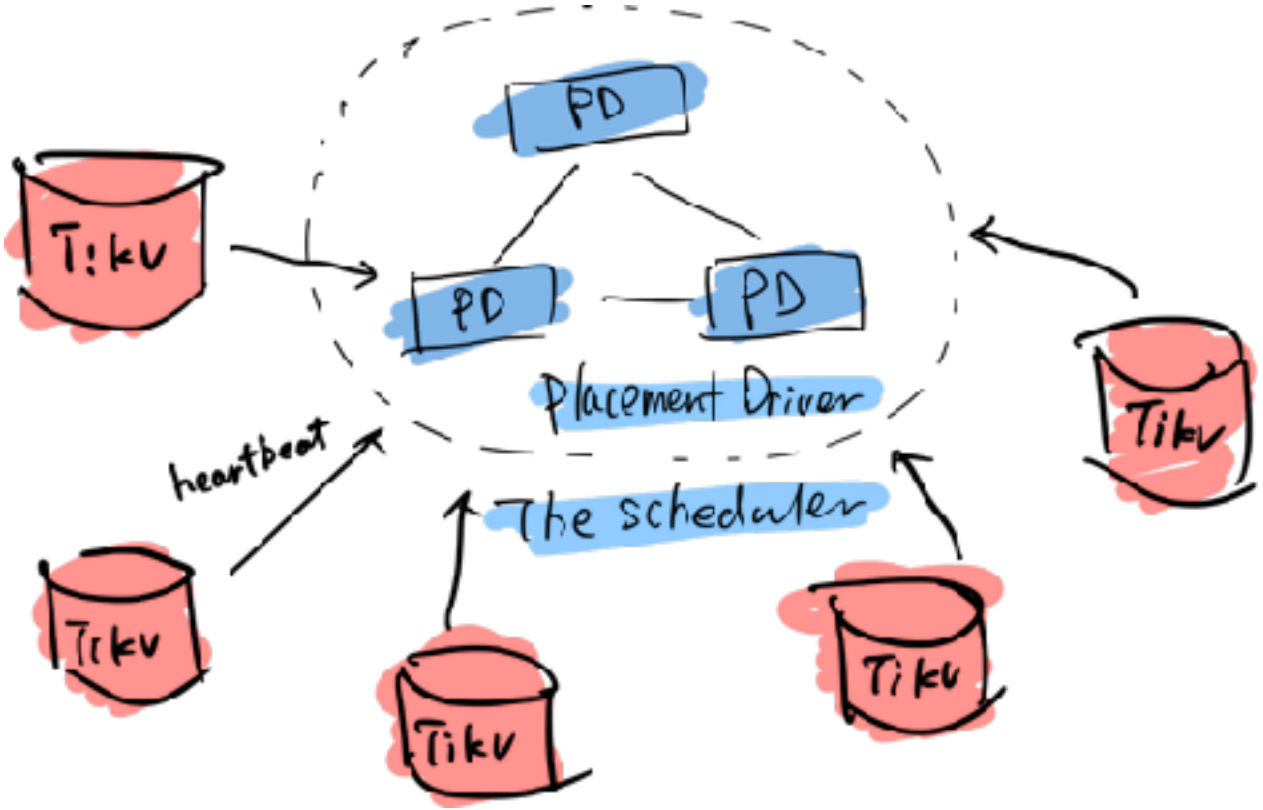
Data movement
Step 1:





Data movement
Step 2:

Placement Driver: The scheduler

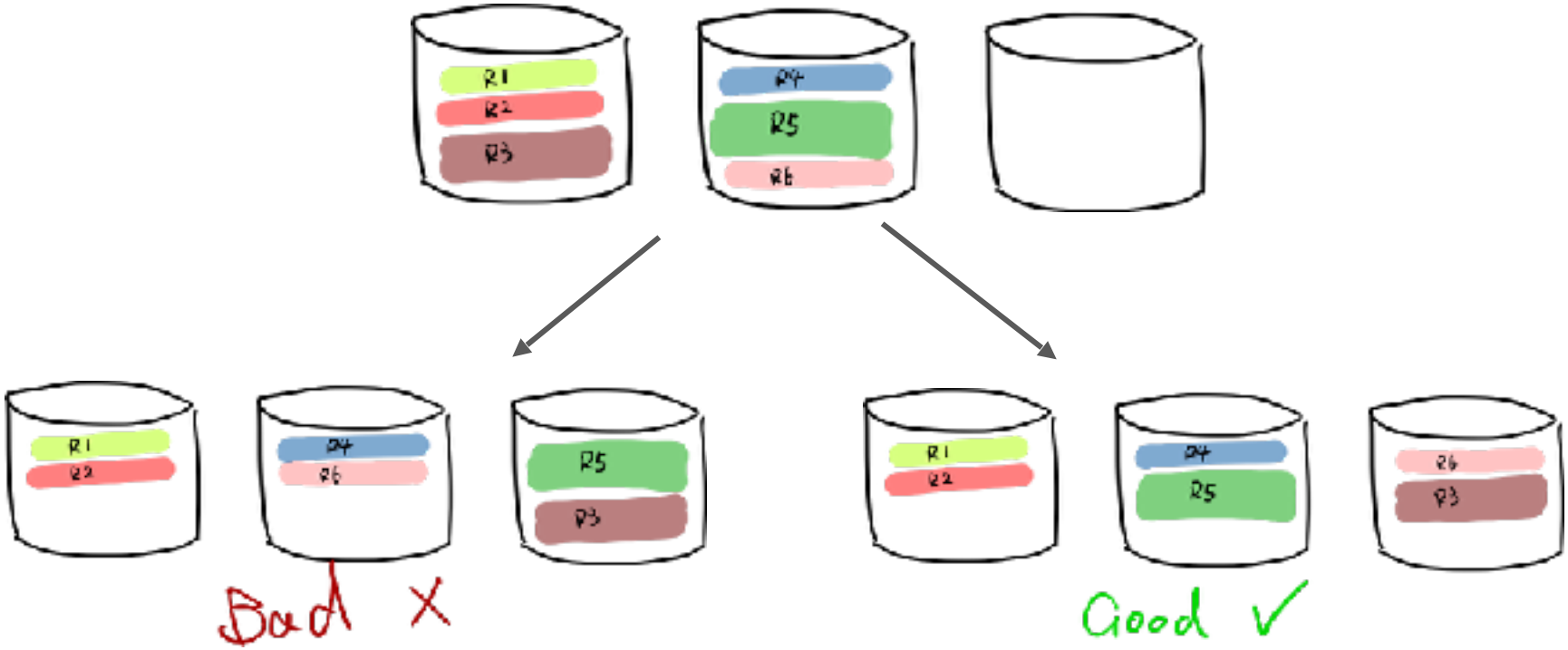


Data : Machine load Info / Disk capacity
Region Info / load on Region
... of each node via heartbeat

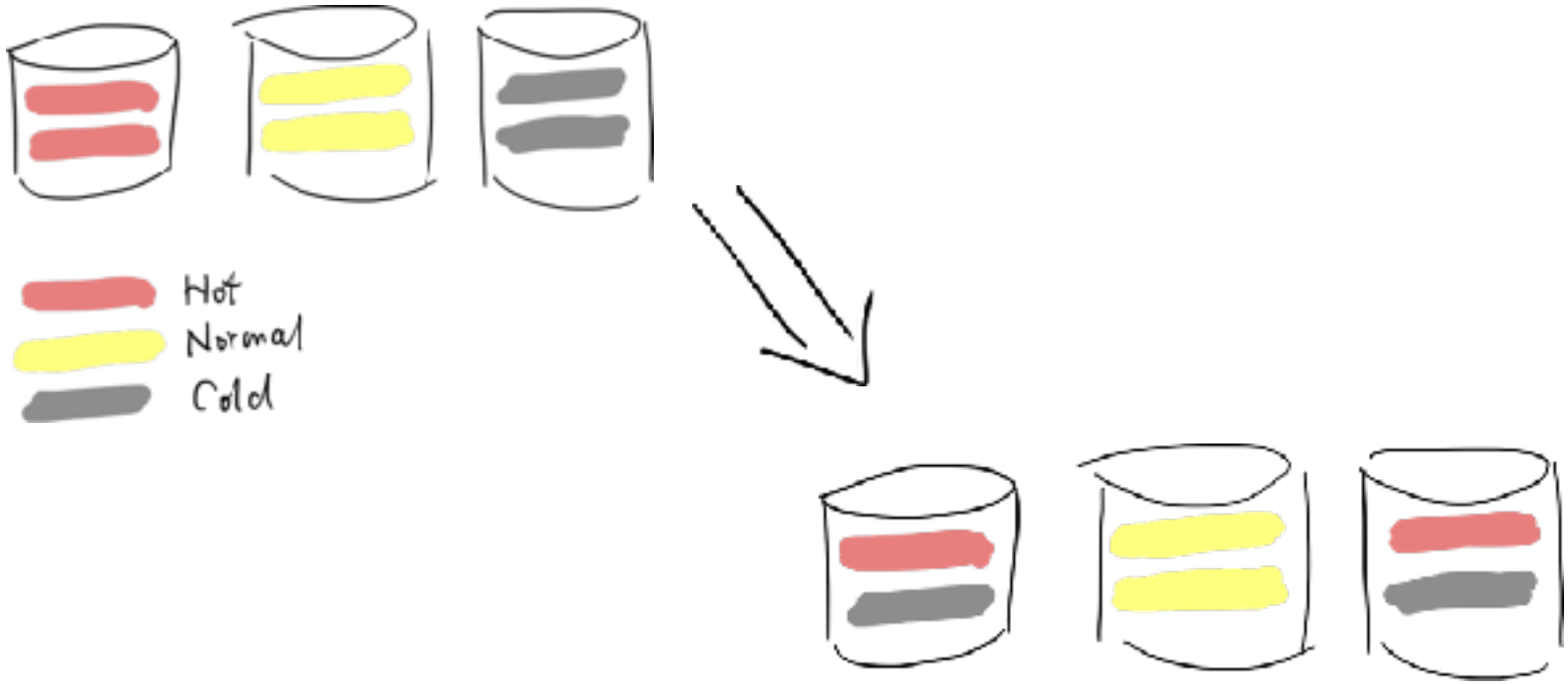
Model : Rules

Action : Add Replica
Remove Replica
Transfer leader
Force Split
⋮

Only by Region count? Size also matters.



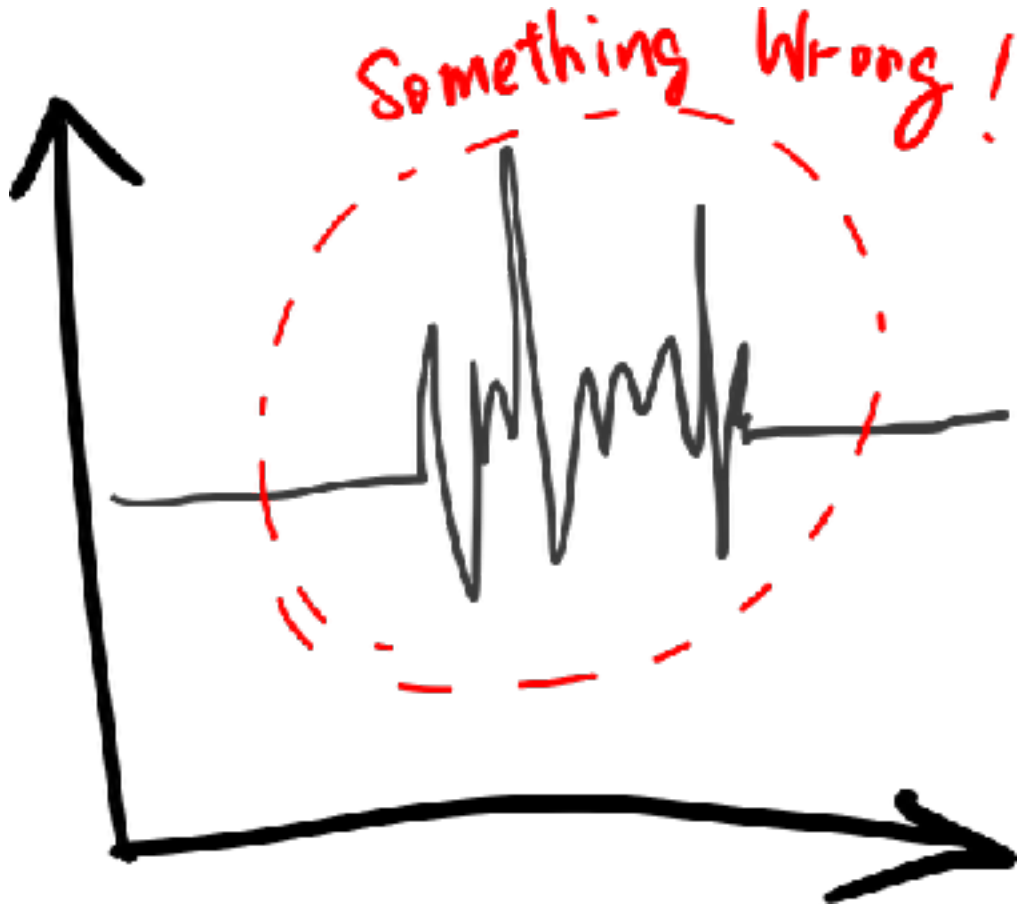
Hot Balance



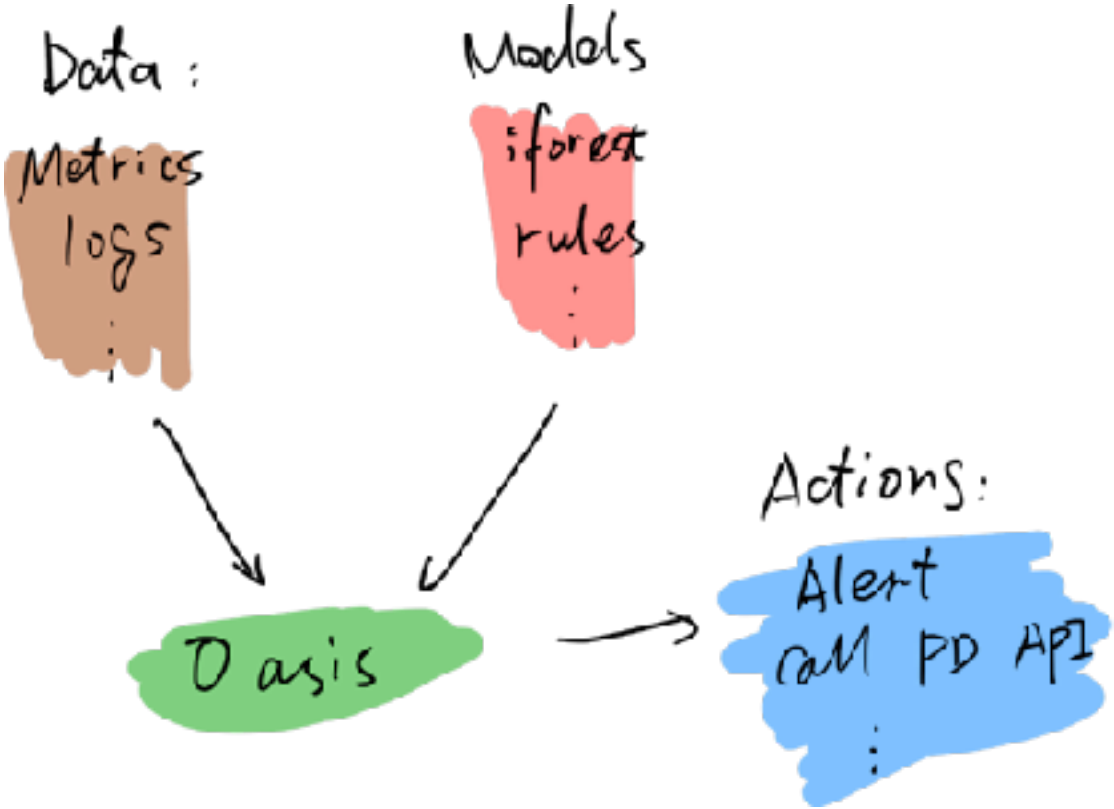
Scheduler - More

- More…
  - Weight Balance - High-weight TiKV will save more data
  - Evict Leader Balance - Some TiKV node can’t have any Raft leader
- OpInfluence - Avoid over frequent balancing

Autonomy

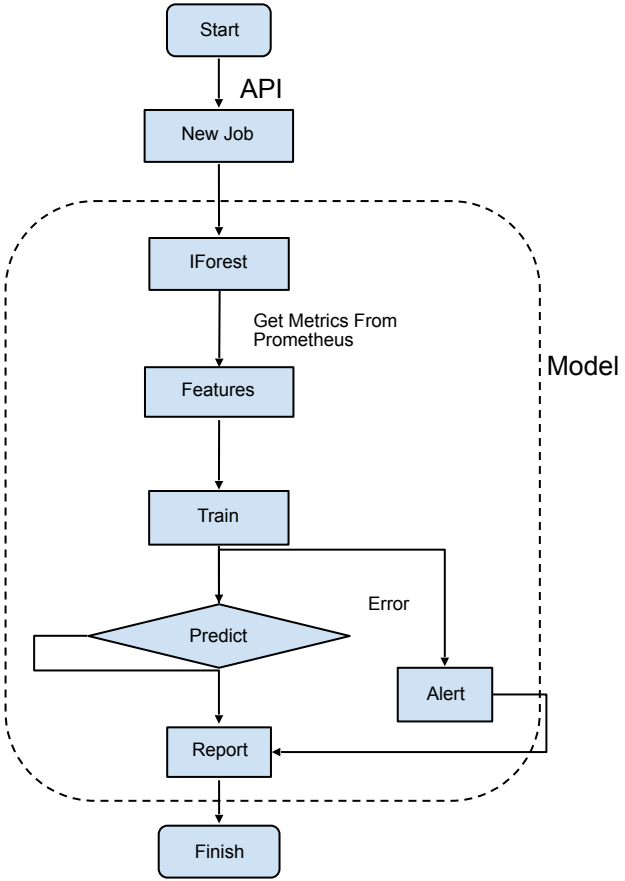

Something Wrong !

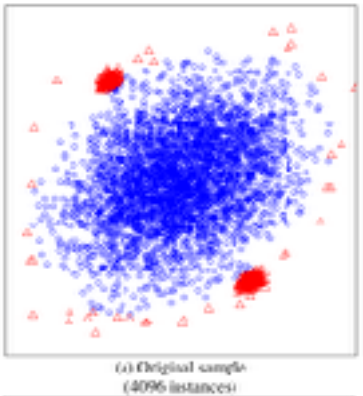
Our work: Oasis



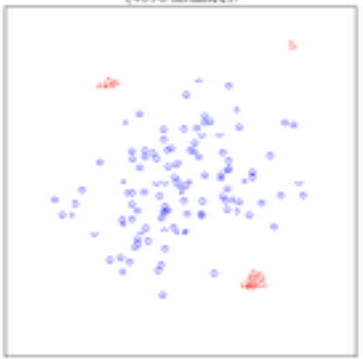
Isolation Forest

- Anomaly detection
- Easy to implement
- [Paper]





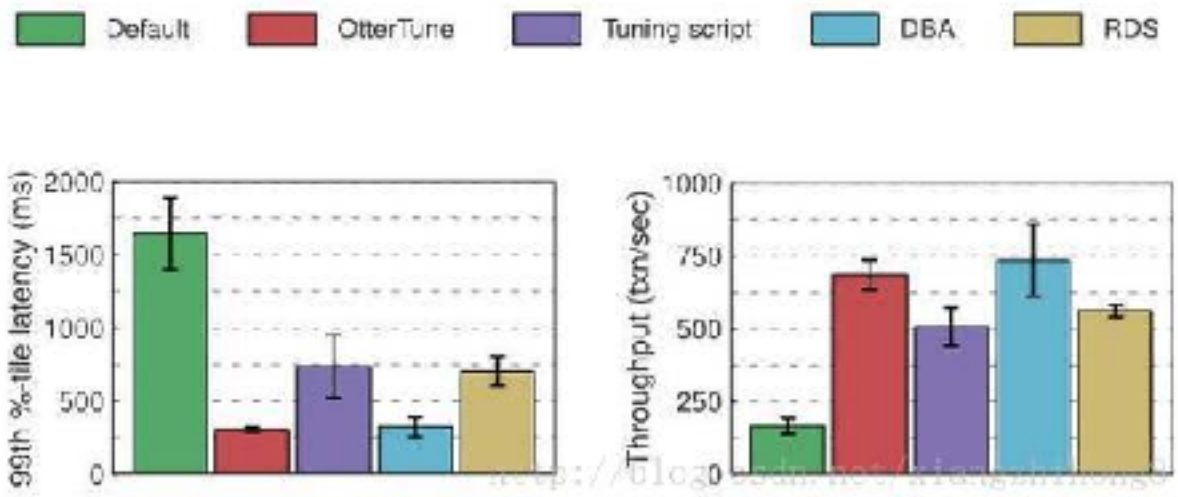
(a) Original sample (4096 instances)



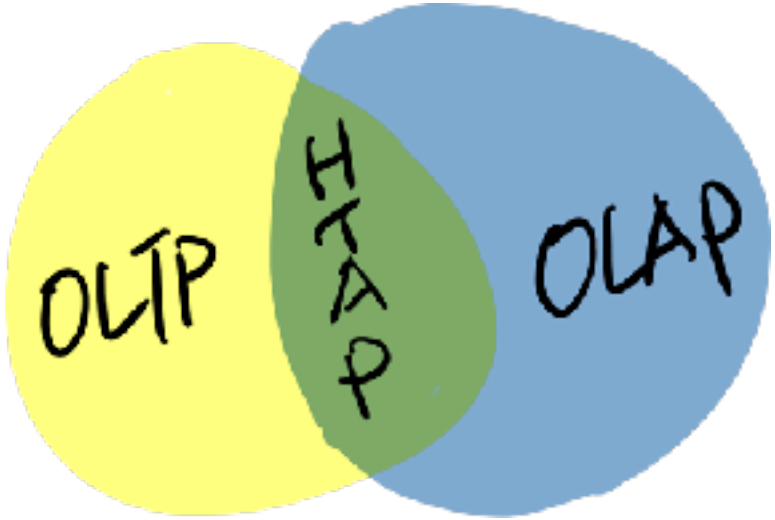
(b) Sub-sample (128 instances)

Related work - OtterTune

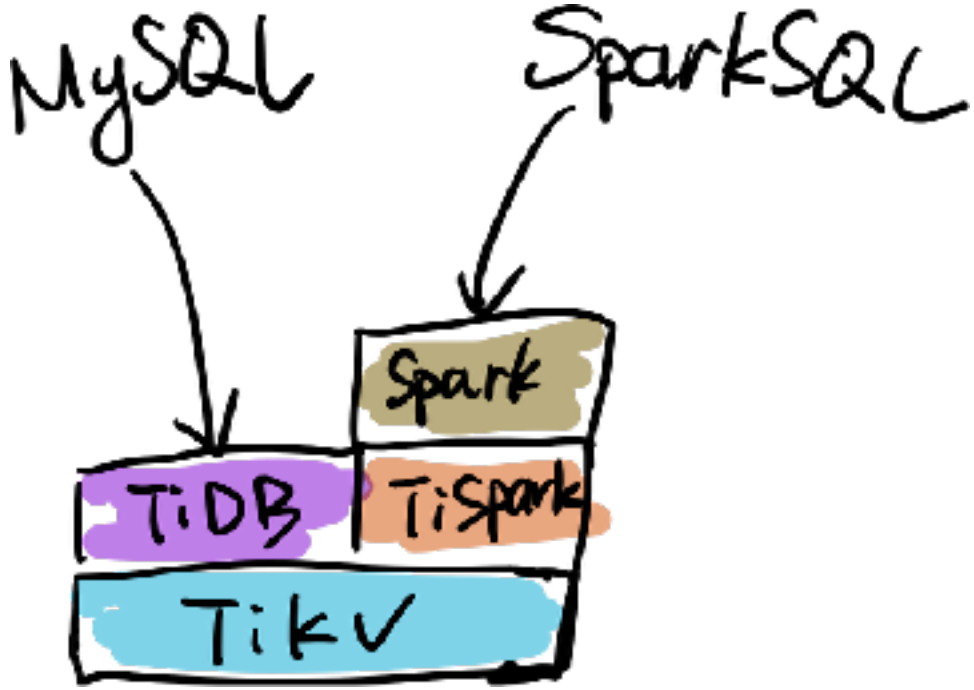
- Database Tuning-as-a-Service
  - Automatically generates DBMS knob configurations
  - Reuse data from previous tuning sessions
- Supported systems
  - PostgreSQL
  - MySQL
  - Greenplum
  - Vectorwise

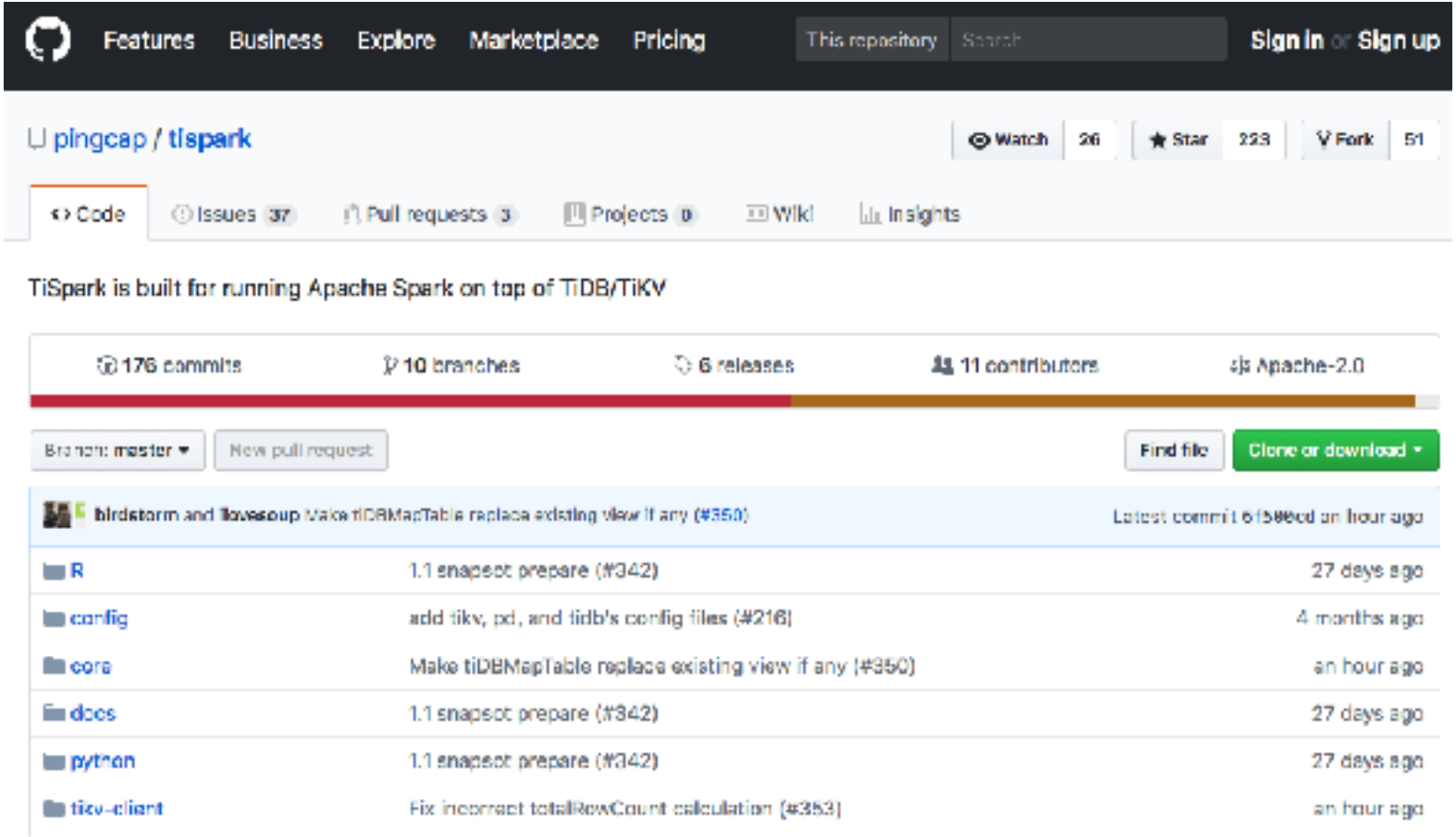


Serving different workloads at the same time

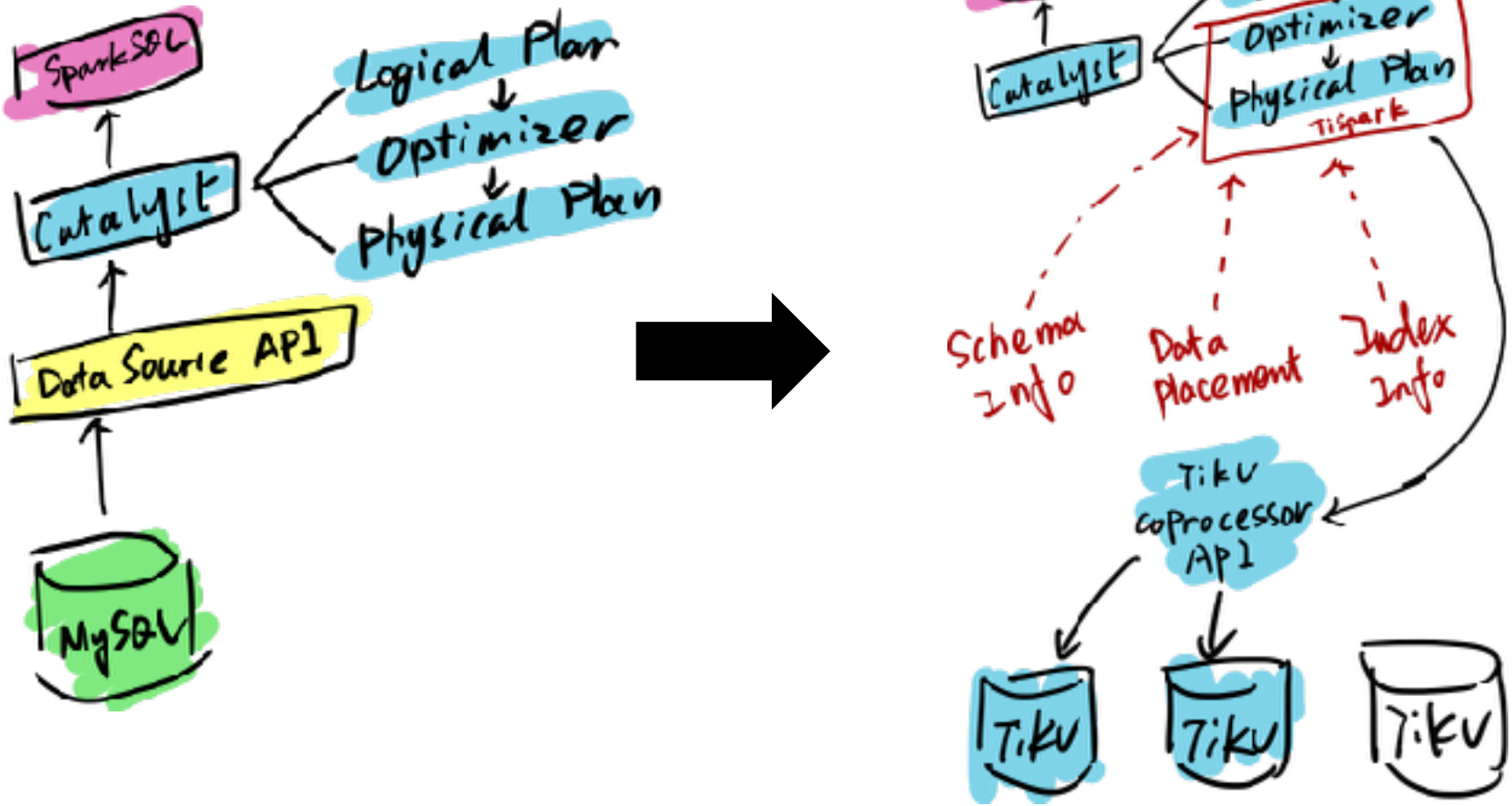

OLTP
HTAP
OLAP

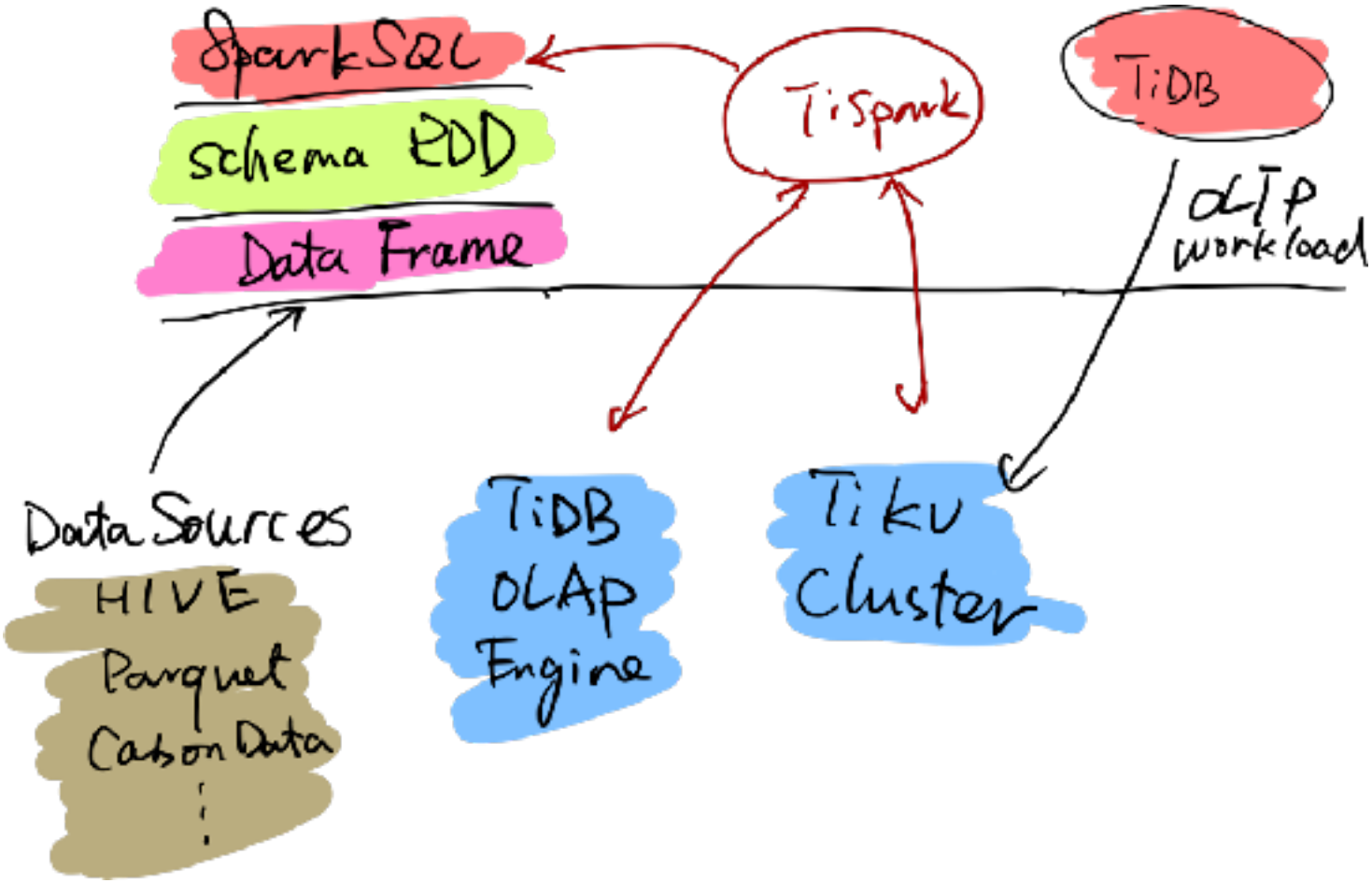

My Posts
Details
黄东旭 | PingCAP
大数据那些事(28)从Spark到Spark
Comment


MySQL
SparkSQL
Spark
TiDB
TiSpark
TiKV

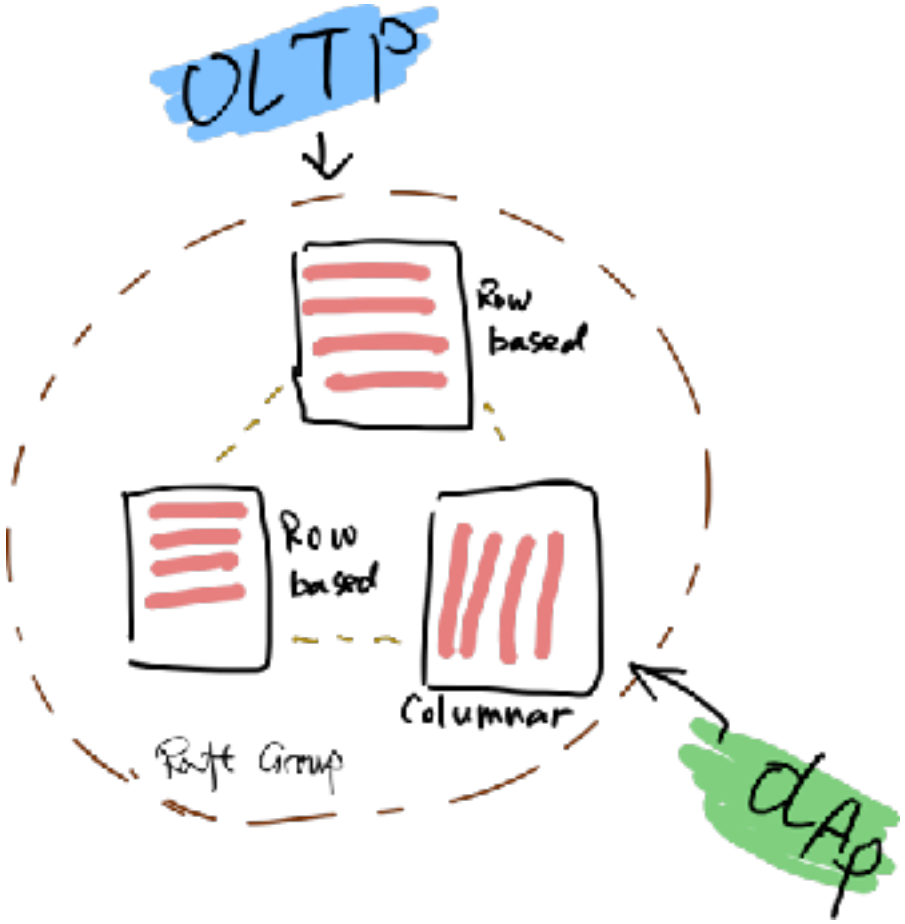


https://github.com/pingcap/tispark


SparkSQL
Catalyst
Logical Plan
Optimizer
Physical Plan
Data Source API
MySQL
SparkSQL
Catalyst
Logical Plan
Optimizer
Physical Plan
TiSpark
Schema Info
Data Placement
Index Info
TiKV
coProcessor API
TiKV
TiKV
TiKV


SparkSQL
Schema RDD
Data Frame
TiSpark
TiDB
OLTP workload
Data Sources
HIVE
Parquet
Carbon Data
TiDB OLAP Engine
Tikv Cluster


My Posts
Details
Comment


OLTP
Row based
Row based
Columnar
Raft Group
OLAP

How to test

- Unit Tests
- Simulator
- Jepsen Test
- 24/7 Chaos Test
- TLA+ (open sourced)

...

Thanks

Q&A