Vitess解析

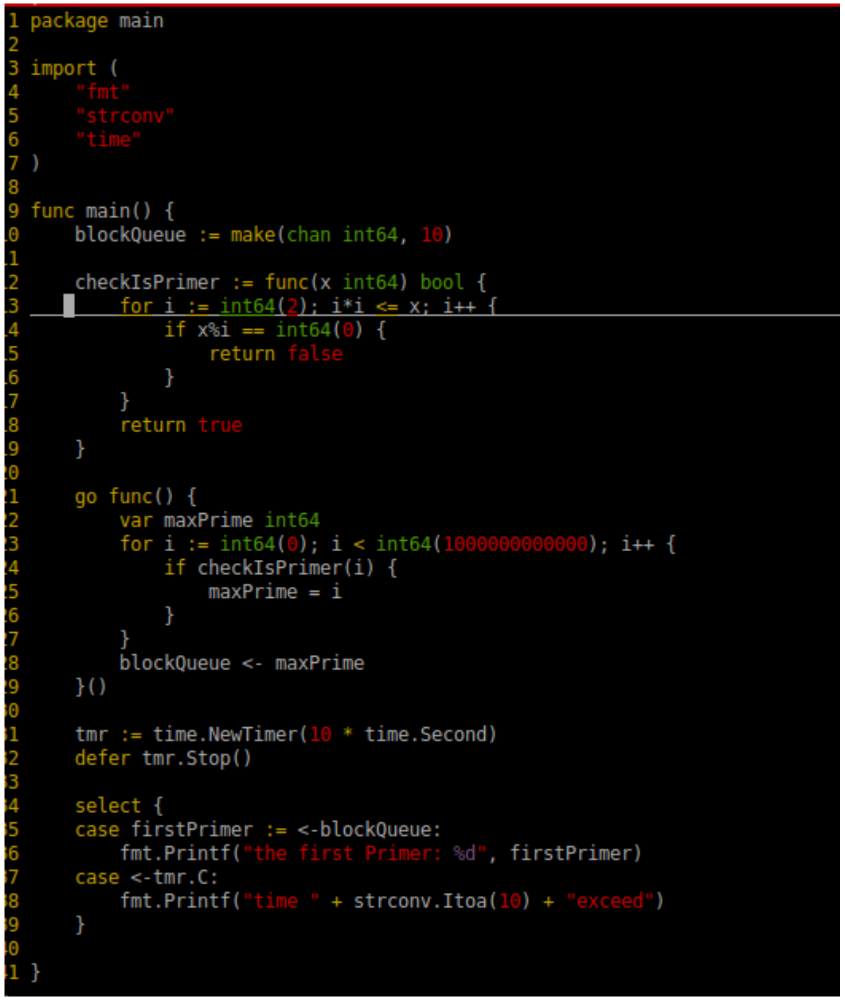
网易杭州研究院后台技术组---胡争

Vitess Introduction

- Golang , GTID
- youtube 2011~2015 , github 2000+ star
- vitess provides servers and tools which facilitate scaling of MySQL databases for large scale web services.
- about 8w line(test code exclude)
- client(bson SSL) java/python/go

Golang ?

- minimalist
- goroutines
- channels & selects
- closures
- defers
- generics
- GC
- map, slices
- performace
  - java < go < c/c++

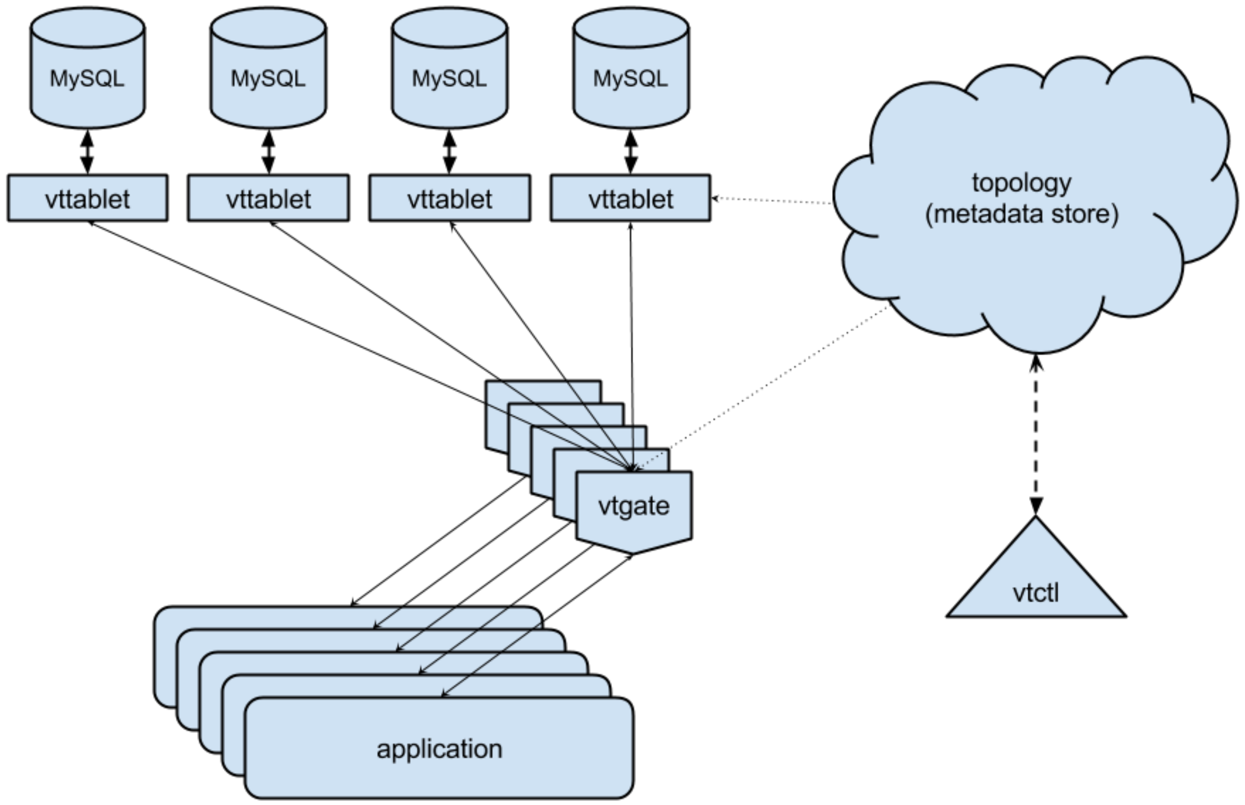


```go
package main

import (
	"fmt"
	"strconv"
	"time"
)

func main() {
	blockQueue := make(chan int64, 10)

	checkIsPrimer := func(x int64) bool {
		for i := int64(2); i*i <= x; i++ {
			if x%i == int64(0) {
				return false
			}
		}
		return true
	}

	go func() {
		var maxPrime int64
		for i := int64(0); i < int64(1000000000000); i++ {
			if checkIsPrimer(i) {
				maxPrime = i
			}
		}
		blockQueue <- maxPrime
	}()

	tmr := time.NewTimer(10 * time.Second)
	defer tmr.Stop()

	select {
	case firstPrimer := <-blockQueue:
		fmt.Printf("the first Primer: %d", firstPrimer)
	case <-tmr.C:
		fmt.Printf("time " + strconv.Itoa(10) + "exceed")
	}

}
```

Vitess Features

- dynamic resharding
- auto-failover
- row-cache
- limit inefficiency SQL
- more client connection
- replication lag optimization
- multi data center

Vitess Topo

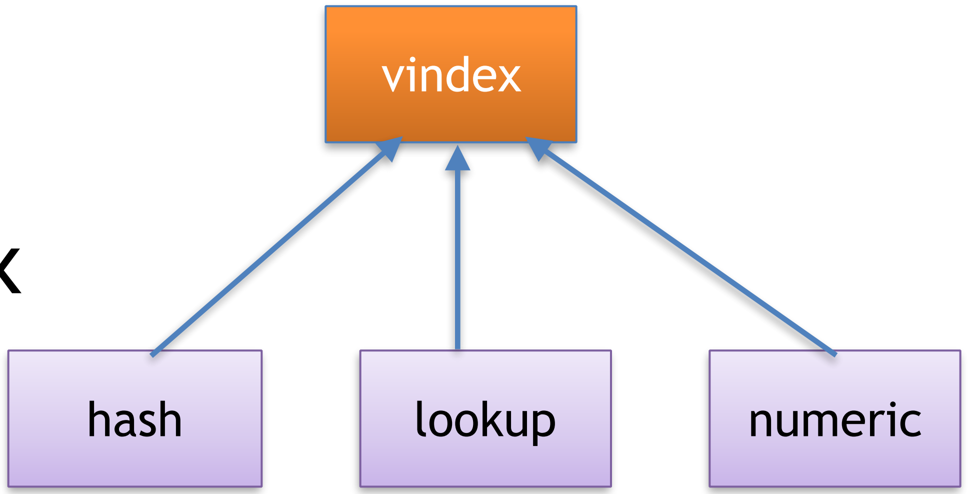

MySQL
MySQL
MySQL
MySQL
vttablet
vttablet
vttablet
vttablet
topology
(metadata store)
vtgate
vtctl
application

Vitess concepts

- keyspace/keyspaceId
- vtctl
- vttablet
- vtgate
- vtctld
- vtworker
- vtprimecache

Vitess concepts

- keyspaceId
- unique Index
- Non-unique Index



- second index support Select only
- How to find shards by keyspaceId ?

Configure Server

/zk/global/vt/keyspaces

/zk/global/vt/keyspaces/action

/zk/global/vt/keyspaces/actionlog

/zk/global/vt/keyspaces/shards

/zk/global/vt/keyspaces/<keyspace>/shards/<shard>

/zk/global/vt/keyspaces/<keyspace>/shards/<shard>/action

/zk/global/vt/keyspaces/<keyspace>/shards/<shard>/actionlog

/zk/global/vt/vschema

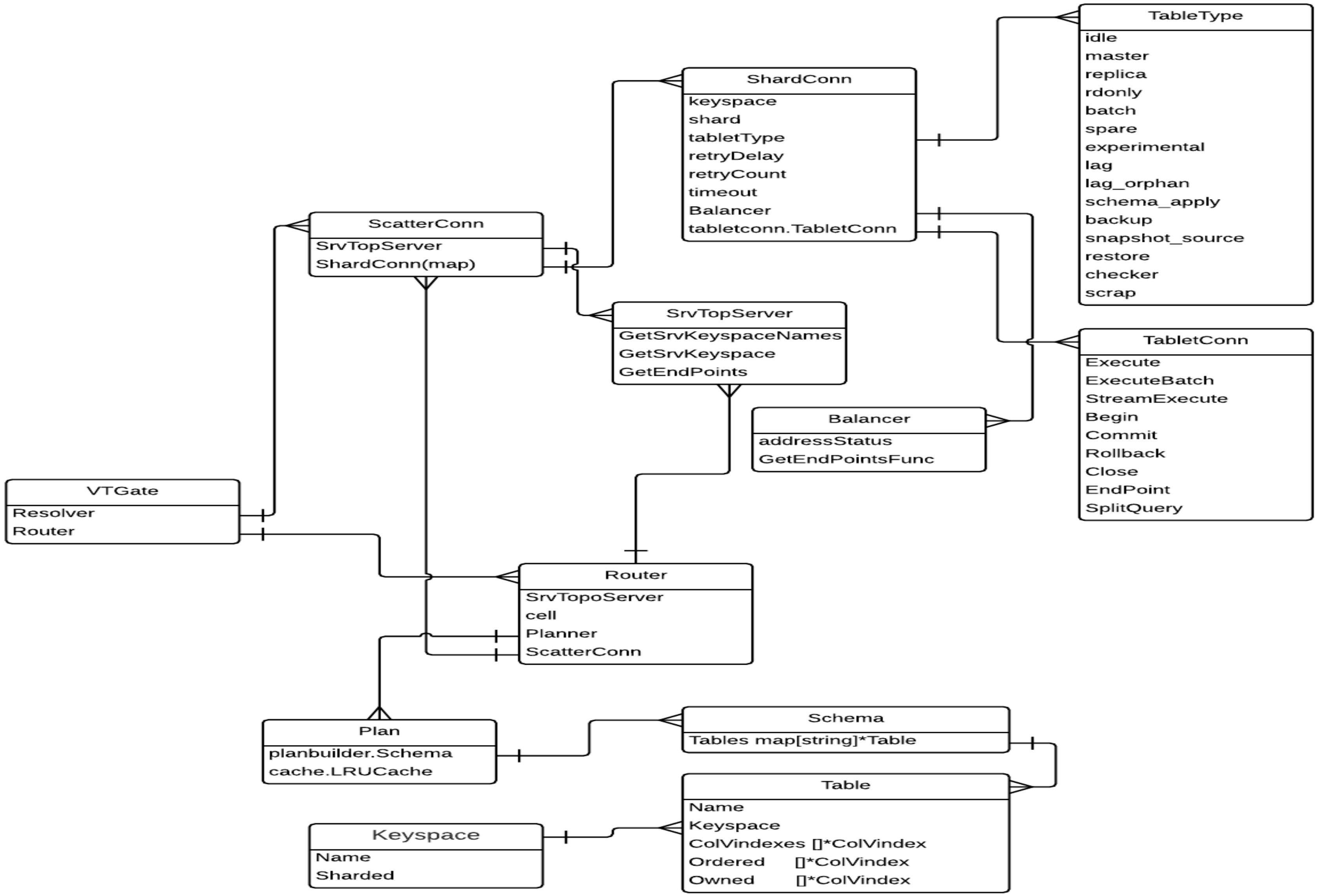
/zk/<cell>/vt/tablets

/zk/<cell>/vt/tablets/<table-uid>

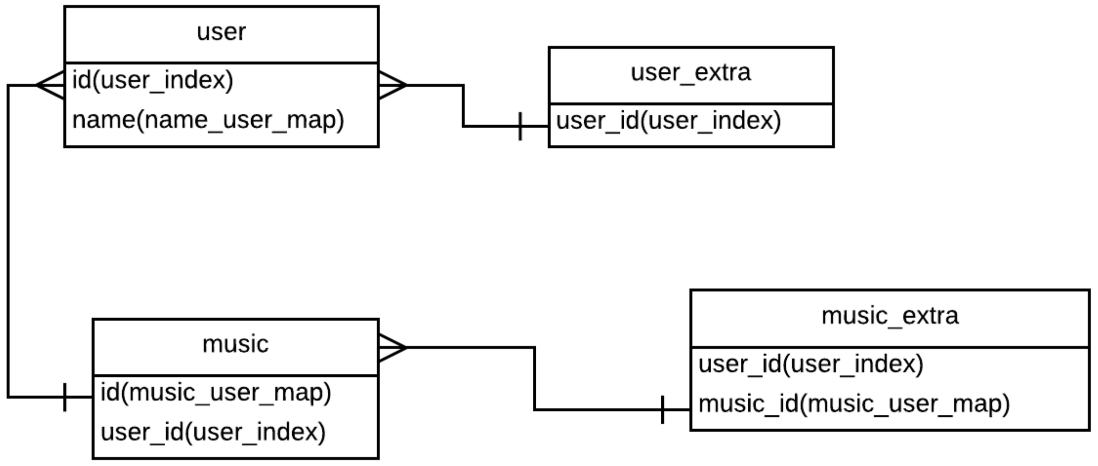
```
$ zk cat /zk/global/vt/keyspaces/ruser/shards/10-20
{
  "MasterAlias": {
    "Cell": "nyc",
    "Uid": 200278
  },
  "KeyRange": {
    "Start": "10",
    "End": "20"
  },
 "Cells": [
    "oe",
    "yh"
 ]
}
```

```
$ zk cat /zk/nyc/vt/tablets/0000200308
{
  "Alias": {
    "Cell": "nyc",
    "Uid": 200308,
  },
  "Parent": {
    "Cell": "",
    "Uid": 0
  },
  "Keyspace": "",
  "Shard": "",
  "Type": "idle",
  "State": "ReadOnly",
  "DbNameOverride": "",
  "KeyRange": {
    "Start": "",
    "End": ""
  }
}
```

Vtgate – Structure



Vtgate – SQL Example



NoPlan
SelectUnsharded
SelectEqual
SelectIN
SelectKeyrange
SelectScatter
UpdateUnsharded
UpdateEqual
DeleteUnsharded
DeleteEqual
InsertUnsharded
InsertSharded

Vtgate – SQL Example

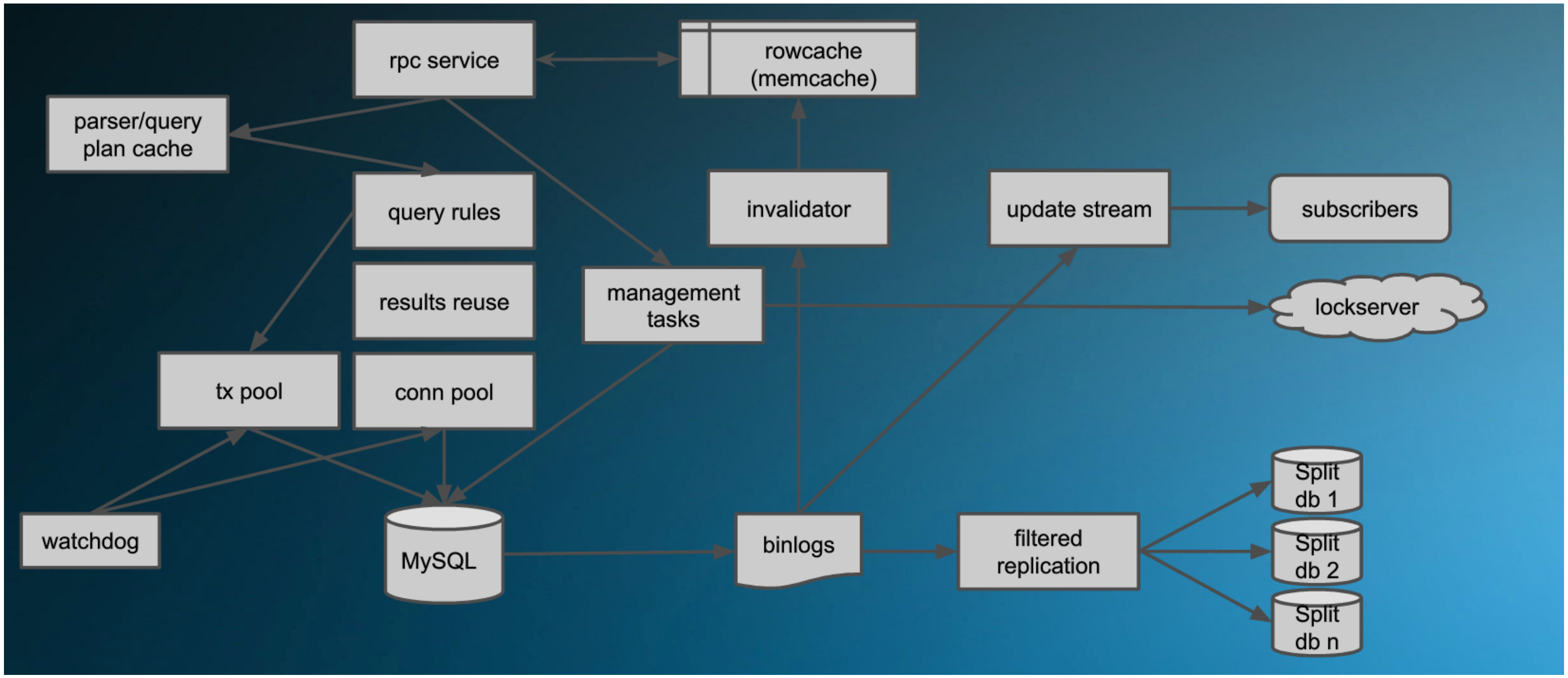
Plan	SQL语句	备注
SelectUnsharded	select * from main1	
SelectScatter	select * from user	
SelectScatter	select * from user where 1 = id	
SelectEqual	select * from user where id = 1	
SelectEqual	select * from user where name = 'foo'	索引不唯一
SelectIN	select * from user where id in (1, 2)	
SelectScatter	select * from user where id = 1 and var = 2 or var = 3	
SelectKeyrange	select * from user where keyrange(1, 2) and a = 1	重写之后，变成：select * from user where a = 1
SelectIN	select exists (select 1 from dual) from user where id in (1, 2)	
UpdateEqual	update user set val = 1 where id = 1	

Plan	SQL语句	原因
NoPlan	select * from user union select * from user	union
NoPlan	set a=1	set
NoPlan	create table a()	DDL
NoPlan	explain select * from user	explain
NoPlan	select * from music, user where id = 1	JOIN
NoPlan	select * from user where id in (select * from music)	subquery
NoPlan	select count(*) from user where name = 'foo'	多片聚合
NoPlan	insert into user(id) values (1), (2)	多行插入
NoPlan	update user set val = 1	更新多片
NoPlan	update user set val = 1 where keyrange(1, 2)	更新多片
NoPlan	update music set id = 1 where id = 1	更新索引
NoPlan	distinct, groupby, having, orderBy, Limit.	多片合并

Vttablet

- User Auth
- SQL Filter
- Binlog Filter
- Backup
- Restore
- Diffs
- Row Limit
- Kill Timeout SQL
- Query blacklisting
- row cache
- more stats & monitor

Vttablet Structure



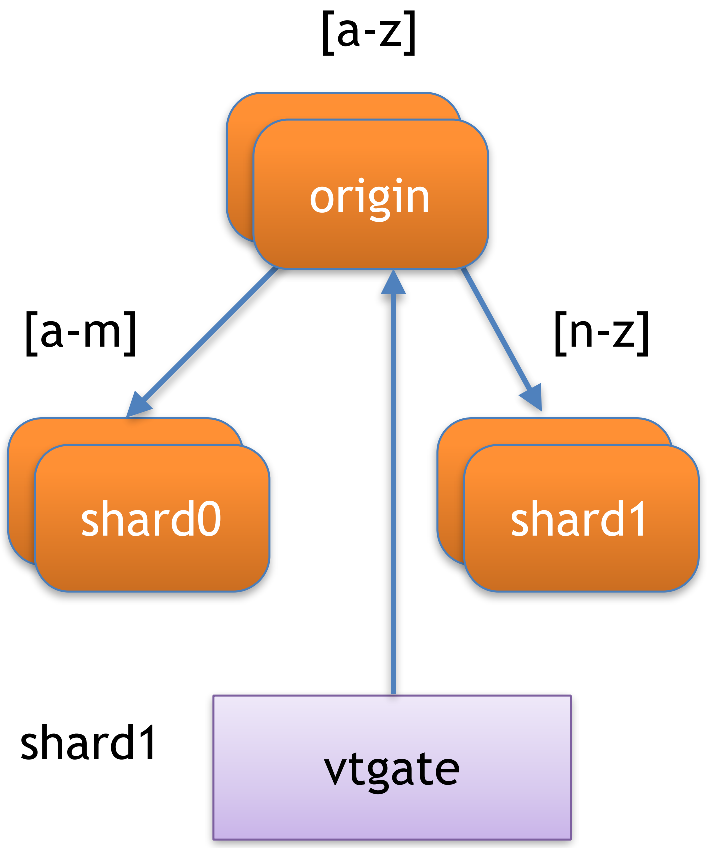
Vitess Resharding

- read scaling up ? —> replica
- write scale up ? —> split

- dynamic shard
- scale
- less than 5 seconds read-only downtime

Vitess Resharding

- shard0, shard1, origin master —> vttablet
- prepare two spare shard0, shard1
- vtgate point to origin master
- multi-snapshot : data & repl-pos
- multi-restore
- set shard0, shard1 read-only.
- shard0, shard1 replication to orign master
- set origin master read-only
- wait until shard0, shard1 catch origin master.
- update configure server, vtgate point to shard0, shard1



Vitess Snapshot

- prepare
- set read-only
- stop slave
- get master position & slave position.
- flush tables with read lock
- select {.keyspaceIdColumnName}, {.Columns} INTO outfile {.dataPath} character set binary fields terminated by ‘,’ optionally enclosed by “” escaped by ‘\\’ lines terminated by ‘\n’ from {.tableName}
- write manifest & gzip

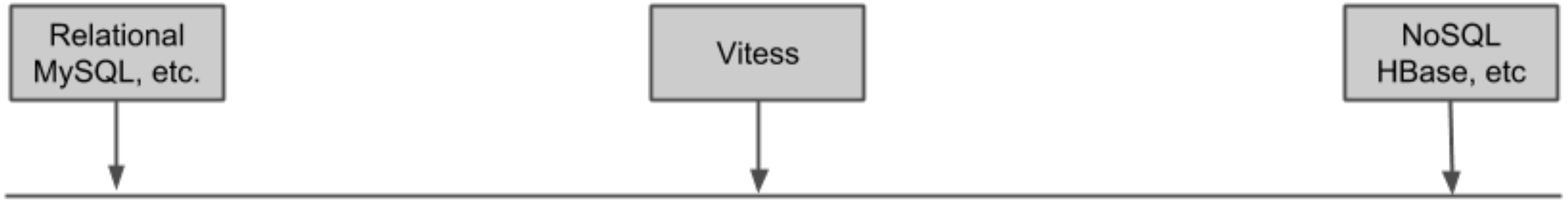
Vitess RowCache

- RowCache VS MySQL Buffer Pool
- LRUCache
- Base on binlog
- DDL ?
  - key=prefix +body
  - every DDL —> prefix:=prefix+1
  - lastest prefix + Body
  - vttable restart ?

Vitess Cons

- bson
- cross-shard groupBy, Limit, having, distinct, orderBy, insert, delete, update.
- cross-shard index
- cross-shard joins
- distribute transaction
- sync replication
- part of second index

Vitess



- Pros
  - Transactions
  - Indexes
  - Joins
- Cons
  - No sharding
  - ACID
  - Schema

- Pros
  - Transactions (limited)
  - Indexes
  - Joins
  - Sharding
- Cons
  - Eventual consistency
  - Schema

- Pros
  - Sharding
  - Unstructured data
- Cons
  - Eventual consistency
  - No transactions
  - No indexes
  - No joins

参考资料

- https://code.google.com/p/vitess/
- https://github.com/youtube/vitess
- http://vitess.googlecode.com/files/Vitess_Percona_2012.pdf
- http://vdisk.weibo.com/s/7JmTtKze0rn?from=page_100505_profile&wvr=6

Thank you