PingCAP

Go 和 TiDB 创造另一个 mongodb

lixia@pingcap.com
@紫沐夏_go

PingCAP

About me

目前在 PingCAP 就职

15 年中加入 PingCAP

目前负责 TiDB 方面各模块的开发和 review

之前在京东就职

12 年末入职，学习 go 并且开始做云推送项目

13 年末开始做存储方面的工作，云存储和弹性块存储项目

联系方式

微博：@紫沐夏_go　　Email: lixia@pingcap.com

PingCAP

Agenda

- mongodb introduction
- gonzo with memory engine
- gonzo with TiKV engine
- TiDB and TiKV
- Transaction
- Q & A

PingCAP

mongodb introduction

SQL术语/概念	MongoDB术语/概念	解释/说明
database	database	数据库
table	collection	数据库表/集合
row	document	数据记录行/文档
column	field	数据字段/域
index	index	索引
table joins		表连接,MongoDB不支持
primary key	primary key	主键,MongoDB自动将_id字段设置为主键

mongodb introduction

Docment

一个键值（key-value）对（即BSON）。MongoDB 的文档不需要设置相同的字段，并且相同的字段不需要相同的数据类型。

id	user_name	email	age	city
1	Mark Hanks	mark@abc.com	25	Los Angeles
2	Richard Peter	richard@abc.com	31	Dallas

```
{
   "_id": ObjectId("5146bb52d8524270060001f3"),
   "age": 25,
   "city": "Los Angeles",
   "email": "mark@abc.com",
   "user_name": "Mark Hanks"
}
{
   "_id": ObjectId("5146bb52d8524270060001f2"),
   "age": 31,
   "city": "Dallas",
   "email": "richard@abc.com",
   "user_name": "Richard Peter"
}
```

PingCAP

gonzo with memory engine

一个 mongo 的内存服务

解析客户端的协议，然后将操作存到内存，最后将操作结果返回给客户端

实现简单的增删改查

PingCAP

gonzo with memory engine

演示

PingCAP

gonzo with memory engine

insert 实现

```
func (c *MemoryCollection) Insert(doc interface{}) error {
        c.mu.Lock()
        defer c.mu.Unlock()

        mdoc, ok := doc.(bson.M)
        if !ok {
                return fmt.Errorf("cannot insert instance of this type: %v", doc)
        }

        var idStr string
        if id, ok := mdoc["_id"]; ok {
                idStr = fmt.Sprintf("%s", id)
        } else {
                oid := bson.NewObjectId()
                mdoc["_id"] = oid
                idStr = oid.String()
        }
        c.docs[idStr] = mdoc

        return nil
}
```

PingCAP

gonzo with memory engine

find 实现

```
func (c *MemoryCollection) Match(pattern bson.M) (result []interface{}) {
        if pattern == nil {
                return c.All()
        }

        c.mu.RLock()
        defer c.mu.RUnlock()
        for _, doc := range c.docs {
                if isPatternMatch(doc, pattern) {
                        result = append(result, doc)
                }
        }

        return result
}
```

gonzo with TiKV engine

目前只实现了 insert 和 find 的接口

用最简单的方式接入

- 与 TiDB 一样分成了 schema 信息和 table 数据存储
- 用了 TiDB 的事物接口 RunInNewTxn
- 都直接从 TiKV 获取数据，可以用 TiDB 自带

PingCAP

gonzo with TiKV engine

insert 实现

```
func (c *TikvCollection) insert(doc []byte) error {
        id, err := c.alloc.Alloc(c.id)
        if err != nil {
                return err
        }

        return kv.RunInNewTxn(c.store, false, func(txn kv.Transaction) error {
                key := tablecodec.EncodeRecordKey(c.docPrefix, id)
                value, err := tablecodec.EncodeRow([]types.Datum{types.NewDatum(doc)}, []int64{0})
                if err != nil {
                        return err
                }
                return txn.Set(key, value)
        })
}
```

gonzo with TiKV engine

find 部分实现

```
err := kv.RunInNewTxn(c.store, false, func(txn kv.Transaction) error {
        startKey := tablecodec.EncodeRecordKey(c.docPrefix, 0)
        it, err := txn.Seek(startKey)
        if err != nil {
                return err
        }
        defer it.Close()
        if !it.Valid() {
                return nil
        }

        valsMap := make(map[int64]*types.FieldType)
        valsMap[0] = types.NewFieldType(mysql.TypeString)
        for it.Valid() && it.Key().HasPrefix(c.docPrefix) {
                id, err := tablecodec.DecodeRowKey(it.Key())
                if err != nil {
                        return err
                }
                vals, err := tablecodec.DecodeRow(it.Value(), valsMap)
                if err != nil {
                        return err
                }

                _, doc, err := convert(vals[0])
                if err != nil {
                        return err
                }
                if isPatternMatch(doc, pattern) {
                        docs = append(docs, doc)
                }

                key := tablecodec.EncodeRecordKey(c.docPrefix, id)
                if err = kv.NextUntil(it, util.RowKeyPrefixFilter(key)); err != nil {
                        return err
                }
        }
        return nil
})
```

gonzo with TiKV engine

演示

PingCAP

mongodb

优：

非结构化存储，schemaless 灵活，查询快速

高可用（Replica Set）、可扩展性、容错能力等属性

劣：

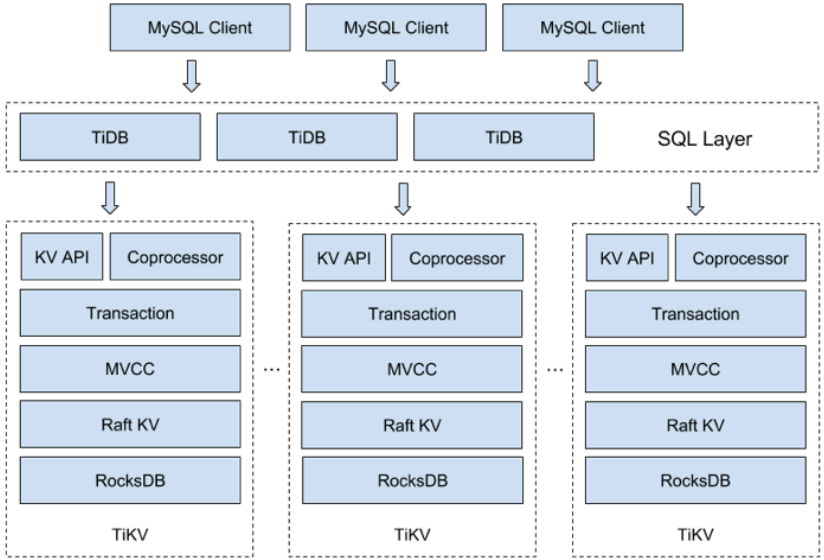
很多做不到ACID特性，不支持事务

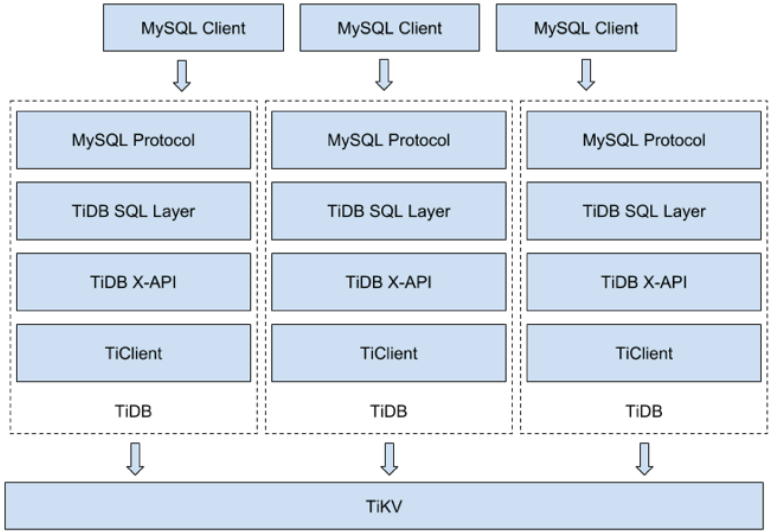
不支持SQL

Cluster 同步带宽占用过多的问题

PingCAP

TiDB and TiKV





TiDB

有如下特性:

MySQL 协议

用户从 MySQL 的相关解决方案迁移过来时几乎没迁移成本。

异步 schema 变更

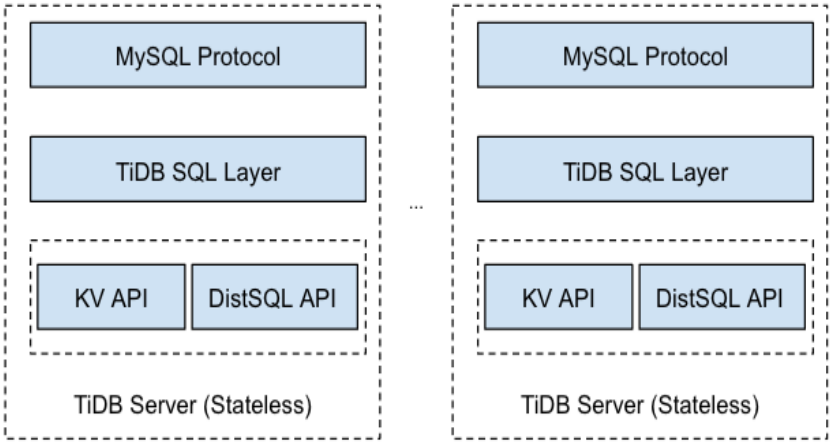
参考 Google 动态变更 schema 的论文

支持 hash join

小表放到内存，等值 key 建立哈希表

大表是用 goroutine 分批取值，匹配哈希表

之后会支持 merge sort join 等常用算法



PingCAP

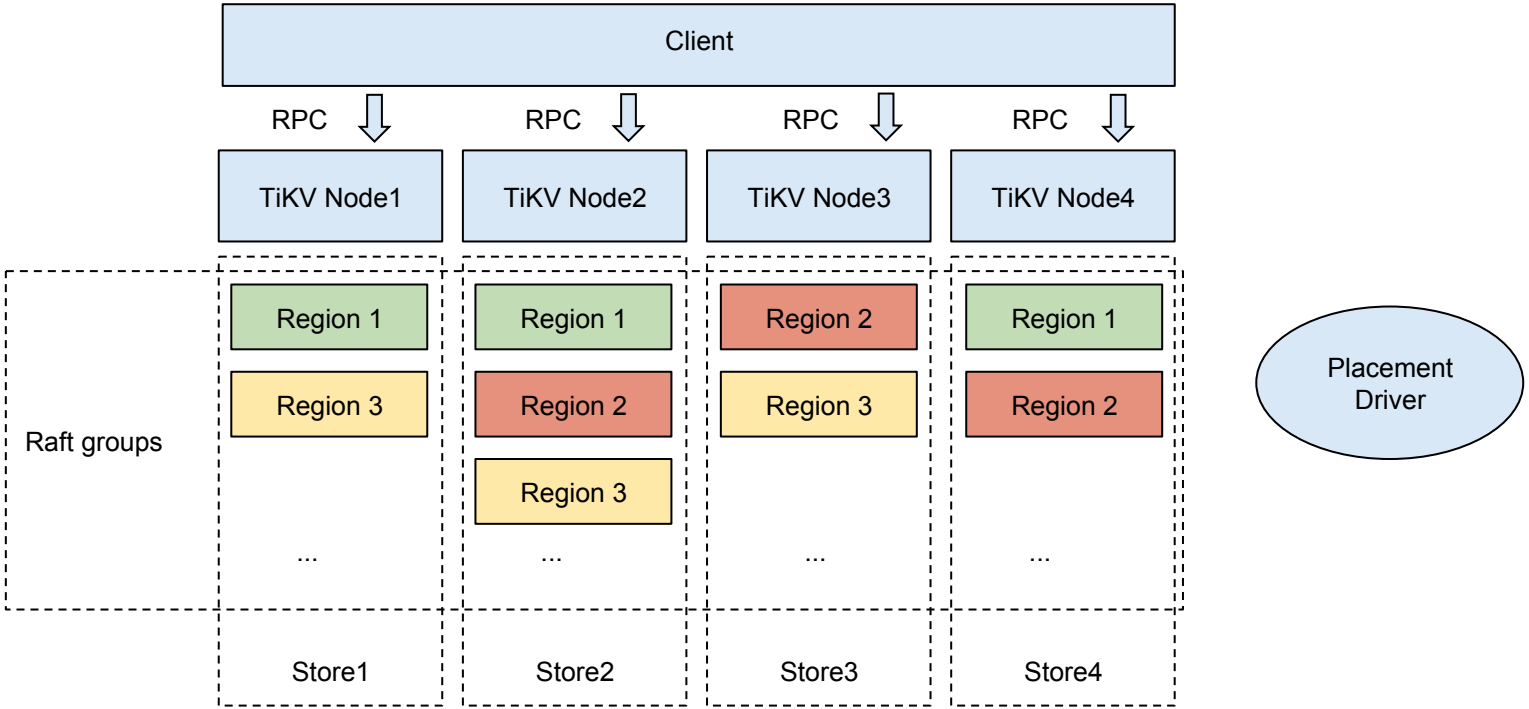
TiDB

有如下特性:

- 分布式事务
  - 2PC（二阶段提交），参考 Google percolator的论文
  - 隔离级别 （SI + 乐观锁）
- 水平扩容/缩容
  - raft 协议 + PlacementDriver
- 容错

PingCAP

TiKV



Transaction

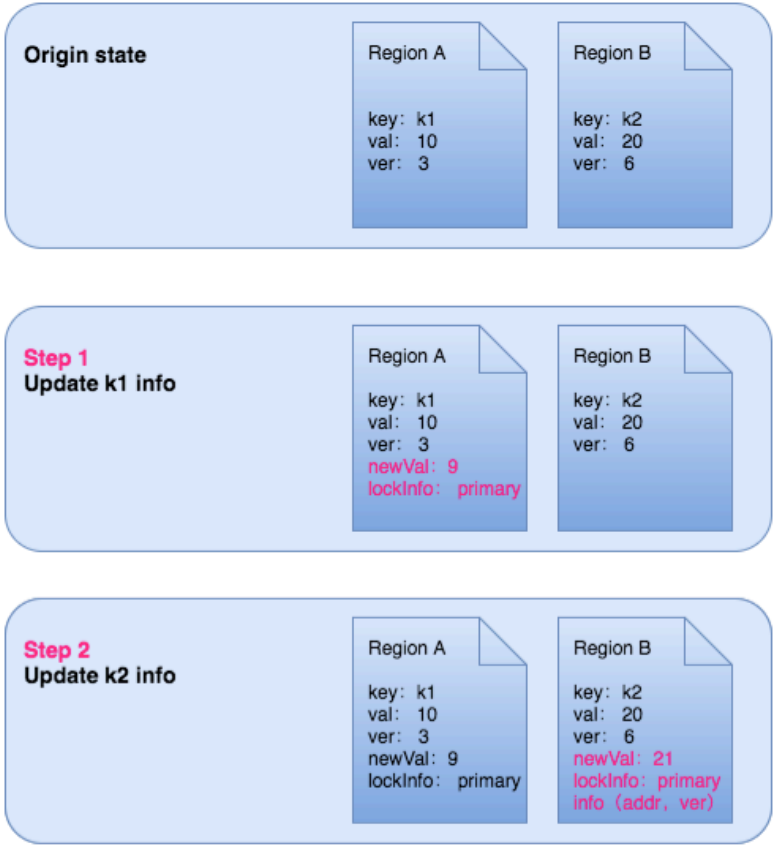
在 kv 上支持事务

k1 和 k2 两人

k1 在银行中存 10 元

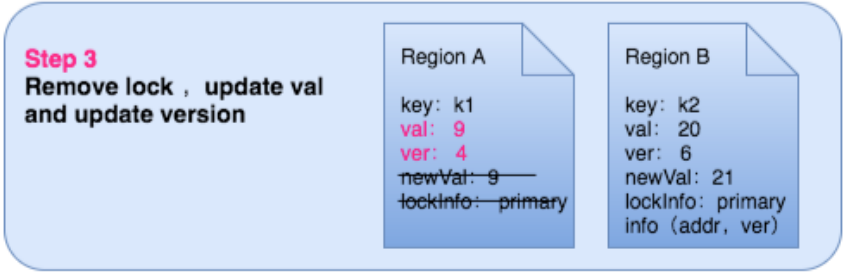
k2 在银行中存 20 元

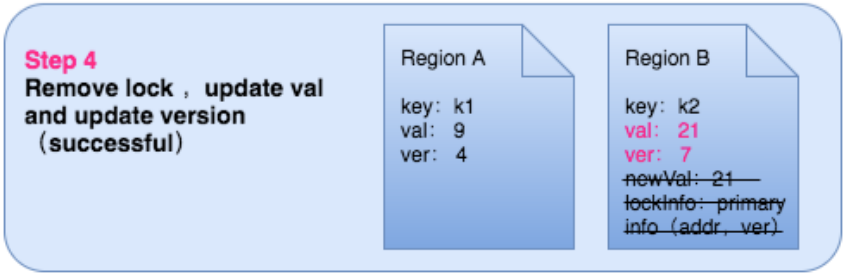
现 k1 给 k2 账户上转 1 元



Transaction

在 kv 上支持事务





假设 step 4 失败

k1 给 k3 转 10 元，当操作进行到 step 3 时

k2 查询自己账户余额

PingCAP

Transaction

目前 TiKV 事务支持的实现

3 column families （存于 rocksdb）

- **lock**: 未提交的事务会填写 primary key 的信息
- **write**: 存储 commit 时间戳
- **data**: 存储数据

PingCAP

TiDB

演示

PingCAP

Feelings

关于代码风格

对于原先的我更多的是代码命名方面的问题，一步一步的修改

在这个团队里，让我的感受是没有挺好，你可以更好

分享个小故事

有时候你也想偷个小懒，某个问题改不改差一点点，于是乎就没改，但是还是会被眼尖的同事发现，然后默默地改过来，是不是有点小忧（活）伤（该）

PingCAP

Q & A

项目地址

TiDB: http://github.com/pingcap/tidb （4700+ stars）

TiKV: https://github.com/pingcap/tikv （1000+ stars）

PingCAP公众号