

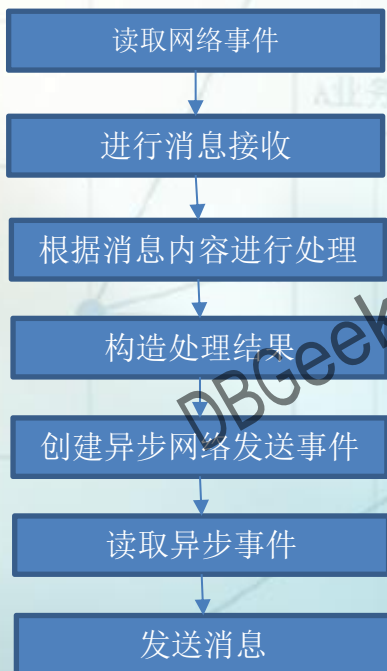
# DBGeek数据库沙龙(上海站)



姜宇祥

## 1) 单线程

### 串行的异步事件



## 2) 无持久存储

# RocksDB的问题



优点

持久化存储  
高效的分层存储

缺点

嵌入式插件，无服务框架

DBGeek数据库沙龙(上海站)

- 1) SQL 标准化
- 2) 多线程并发
- 3) 多套HA方案

DBGeek数据库沙龙(上海站)

# Redis+RocksDB



- get<t>
- set<t>
- getvalues<t>
- containskey
- hget
- expireentryin
- del
- getvaluefromhash
- getvaluesfromhash
- getrangefromlist
- getallitemsfromset
- additemto
- popitemfromset
- getse

```
2016-11-22 14:25:00
get<t> 23654684
set<t> 8673120
getvalues<t> 5615741
containskey 1751935
hget 1635898
expireentryin 1498841
del 1099546
getvaluefromhash 928200
getlistcount 814423
setentryifnotexists 674395
```

DBGeek数据库沙龙(上海站)

将Redis中的set/get等命令中的操作转换为对RocksDB的操作

通过存储引擎中间层，将MySQL的逻辑层和RocksDB进行连接。  
将SQL语句映射为RocksDB的操作。

DBGeek数据库沙龙（上海站）

- 1、kv存储引擎
- 2、rocksdb接口的封装
- 3、SQL语法到rocksdb封装接口映射转换

DBGeek数据库沙龙（上海站）



# 效率



- 1、插入效率
- 2、查询效率

DBGeek数据库沙龙（上海站）



# RocksDB的效率对比



Test step	LevelDB	RocksDB	HyperLevelDB	LMDB
Write 100M values	36m8.29s	21m18.60s	<b>10m45.41</b>	<b>1h13m21.30s</b>
DB Size	<b>2.7G</b>	3.2G	3.2G	<b>7.6G</b>
Query 100M values	2m55.37s	<b>2m44.99s</b>	<b>13m49.01s</b>	5m24.80s
Delete 50M values	3m47.64s	<b>1m53.84s</b>	6m0.38s	<b>6m15.98s</b>
Compaction	3m59.87s	<b>3m20.27s</b>	6m33.36s	<b>1.548us</b>
DB Size	<b>1.4G</b>	1.6G	1.6G	<b>7.6G</b>
Query 50M values	12.12s	13.59s	<b>23.98s</b>	<b>8.48s</b>
Write 50M values	3m5.28s	<b>1m26.9s</b>	1m54.56s	<b>3m25.96s</b>
DB Size	<b>673M</b>	993M	928M	<b>2.5G</b>

DBGeek数据库沙龙 (上海站)

# MySQL插入效率改进尝试



修改MySQL的存储引擎层接口，调整插入的框架，批量改进

```
virtual void start_bulk_insert(ha_rows rows) {}  
virtual int end_bulk_insert() { return 0; }  
protected:
```

```
int ha_write_row(uchar * buf);  
int ha_write_row_batch(TABLE *insert_table, void *insert_many_values);  
int ha_update_row(const uchar * old_data, uchar * new_data);
```

批量	非批量
0.53	0.69
0.45	0.66
0.47	0.64
0.46	0.67

改后时间缩短为改进前的 72.1%

DBGEEK数据库沙龙(上海站)

- 1、kv接口的标准化
- 2、MySQL存储引擎接口的改进

DBGeek数据库沙龙（上海站）