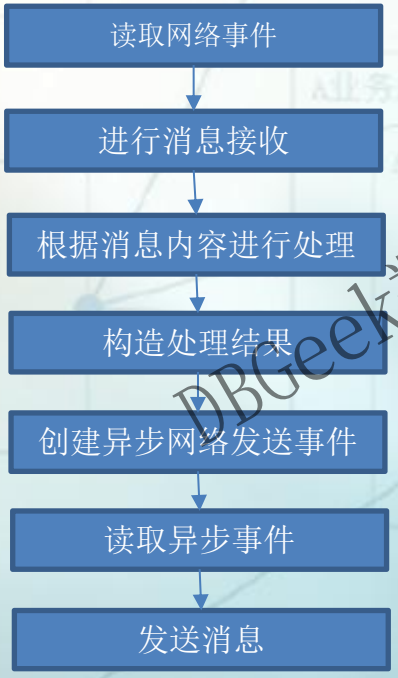
姜宇祥

Redis的问题

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

MySQL

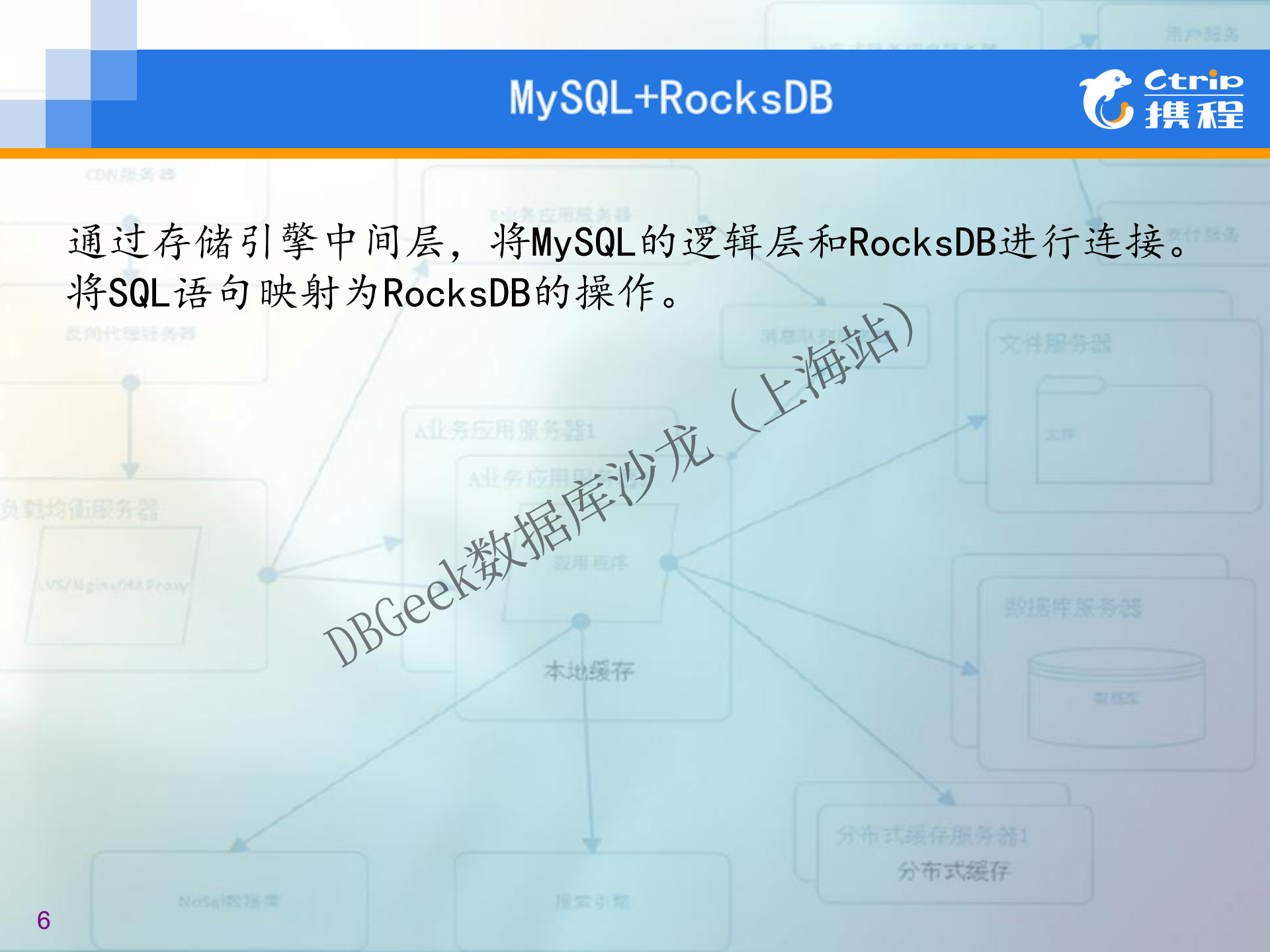
1）SQL标准化
2）多线程并发
3）多套HA方案

Redis+RocksDB

将Redis中的set/get等命令中的操作转换为对RocksDB的操作

MySQL+RocksDB

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。



实践

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

效率

1、插入效率
2、查询效率

RocksDB的效率对比

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

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_);
```

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%

未来之路

1、kv接口的标准化
2、MySQL存储引擎接口的改进