# OneValue



http://www.onexsoft.com

技术交流 QQ 群: 324460822

# OneValue — 高性能、可持久化的 KeyValue 数据库

#### 开发初衷

1.现在主流的缓存数据库 memcache、redis 都是常驻内存的,一旦出现突然关机等问题,内存中数据将会丢失,在这种情况下,如果有比较重要的数据是很难保证得 到备份的。

2.为了实现一个基于磁盘文件的 KeyValue 数据库,所有的实时操作都能正确回写到磁盘,保正数据的正确性,持久性。任意时刻能最大程度保证数据的完整性。

## 产品介绍

软件的存储引擎采用的是 Google 的 leveldb 框架, leveldb 是一个 C++库, 是一个非常高效的 KV 数据库(提供了类似 Redis 的 set,get,del 操作方法),但不包含网络层,也不支持类似 list,hashmap 等的数据结构。我们是在此基础上实现了其他 Redis 命令的支持,增加了高性能的网络层,并扩展了一些其他有用的功能。

#### 主要功能

- 1.网络通信上采用的协议同 Redis 一样,所以客户端操作上是同 Redis 是一样的,在命令支持上覆盖了大部分的 Redis 命令,所以可以当作可持久化的 Redis 来使用。 多线程高并发模型。能同时处理大量的客户端,这一点和单线程的 Redis 是不一样的。
- 2.底层多个 DB 实例,采用 HASH 分片方式分配到不同的实例,可以充分利用底层存储功能。
- 3.支持生成 Binlog 文件,可用于主从同步。
- 4.支持主从同步。采用异步方式进行同步,节点 A 可以认定节点 B 为主、节点 C 可以认定节点 B 为主,因此是可以构建出多级主从模型。

## 软件使用

# 软件只需要一个配置文件即可以启动,配置文件及参数如下:

```
<onevalue port="8221" thread_num="15" hash_value_max="80" work_dir="mydb" daemonize="0" guard="0" log_file="" unix_socket_file="">
 <!-- port: onevalue 工作端口 -->
 <!-- thread_num: 线程数 -->
 <!-- hash_value_max: hash 槽个数 -->
 <!-- work_dir: onevalue 存储目录 -->
 <!-- daemonize: 是否后台运行 1=yes 0=no -->
 <!-- guard: 是否开启守护进程 1=yes 0=no -->
 <!-- log_file: 日志文件路径 -->
 <!-- unix_socket_file: unix_socket 文件路径 -->
 <db option sync="0" compress="0" lru cache size="0" write buf size="0"></db option>
 <!-- sync: 是否采用同步写入方式 1=yes 0=no -->
 <!-- compress: 是否启用压缩 1=yes 0=no -->
 <!-- lru_cache_size: LRU 大小(MB) -->
 <!-- write_buf_size: write buffer 大小(MB) -->
 <db_node name="db1" hash_min="0" hash_max="19"></db_node>
 \label{limin} $$ \db_node name="db2" hash_min="20" hash_max="39" ></db_node> $$
  <db node name="db3" hash min="40" hash max="59"></db node>
  <db_node name="db4" hash_min="60" hash_max="79"></db_node>
  <!-- name: 数据库名称 -->
 <!-- hash_min: 所使用的 hash 槽(min) -->
 <!-- hash_max: 所使用的 hash 槽(max) -->
 \label{log_size} $$ \langle binlog max\_binlog\_size="64" enabled="0"></binlog> 
 <!-- max_binlog_size: 单个 binlog 文件最大大小(MB) -->
 <!-- enabled: 是否启用 1=yes 0=no -->
 <master ip="172.30.12.12" port="8221" sync_interval="5"></master>
 <!-- ip: 主的 IP 地址 -->
 <!-- port: 主的端口 -->
 <!-- sync_interval: 请求更新间隔(秒) -->
</onevalue>
```

## 性能测试

```
测试环境描述
```

```
机器类型:物理机(16 Intel(R) Xeon(R) CPU E5620 @ 2.40GHz 32G 内存 千兆网卡)
这个是服务端配置, 15个线程数, 12个DB实例, 其他参数都是默认的
\label{lem:conevalue_port="8221" thread_num="15" hash_value\_max="240" work\_dir="mydb">
  \label{limin} $$ \db_node name="db1" hash_min="0" hash_max="19" ></db_node> $$
  \label{localization} $$ \db_node name="db2" hash_min="20" hash_max="39" ></db_node> $$
  \label{localization} $$\db_node\ name="db3"\ hash_min="40"\ hash_max="59"></db_node> $$
  \label{limits} $$ \db_node name="db4" hash_min="60" hash_max="79" > </db_node > $$
  \label{limits} $$ \db_node name="db5" hash_min="80" hash_max="99" > </db_node > (db_node) $$
  \label{limits} $$\db_node\ name="db6"\ hash_min="100"\ hash_max="119"></db_node> $$
  \label{limits} $$\db_node\ name="db7"\ hash_min="120"\ hash_max="139"></db_node> $$
  <db_node name="db8" hash_min="140" hash_max="159"></db_node>
  \label{limits} $$\db_node\ name="db9"\ hash_min="160"\ hash_max="179"></db_node> $$
  <db_node name="db10" hash_min="180" hash_max="199"></db_node>
  \label{localization} $$ \db_node name="db11" hash_min="200" hash_max="219" > </db_node > $$
  \label{localized} $$\db_node\ name="db12"\ hash_min="220"\ hash_max="239"></db_node> $$
</onevalue>
然后在另外一台机器开启多个 redis-benchmark 进行并发测试,测试脚本如下:
for i in \{1..5\}
```

nohup ./redis-benchmark -h 172.30.12.12 -p 8221 -t get, set -r 1000000 -n 1000000 -q > \${i}.log 2>&1 &

同时启动 10 个 redis-benchmark 进行测试,结果会定向到不同的文件。

#### 测试完后,我们查测试结果,如下

```
[root@localhost redis]# cat *.log
SET: 60971.89 requests per second
GET: 84288.60 requests per second
SET: 60244.59 requests per second
GET: 83619.03 requests per second
SET: 60620.75 requests per second
GET: 80952.00 requests per second
SET: 60444.87 requests per second
GET: 81017.58 requests per second
SET: 60459.49 requests per second
GET: 87943.01 requests per second
```

SET 性能大概是 30W 每秒, GET 性能大概是接近 45W 每秒。如果是 SSD 则效果更好。