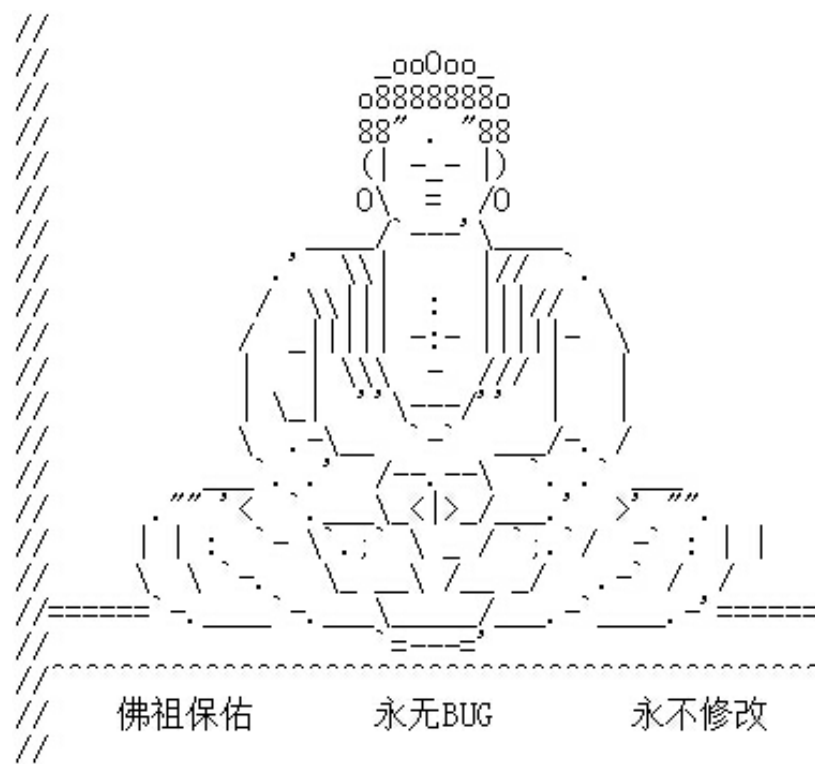


Istore-ssd存储系统介绍

基础平台组

yuhuliu@tencent.com



不用加班 永无Bug !!!

大纲

 Istore-ssd系统介绍





 Istore-ssd设计与实现

 Istore 集群设计






Istore-ssd系统介绍

Istore-ssd 存储项目背景/目标

背景：

-  原有redis集群数据冷热不均
-  数据保存在内存，断电后丢失
-  个性化推荐需要海量存储容量
-  成本和性能难以均衡

目标：

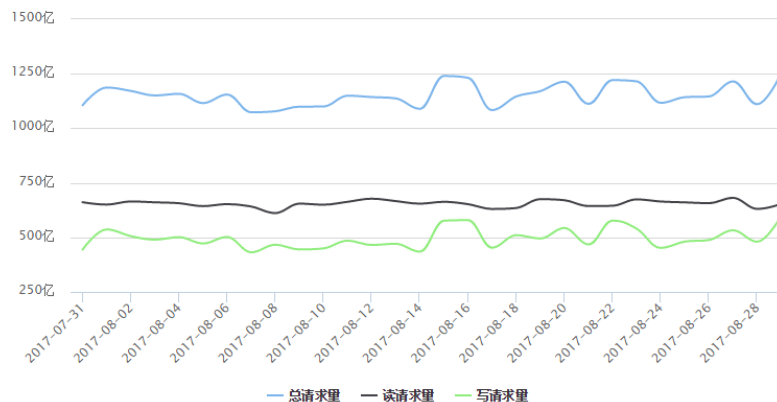
-  兼容redis协议，方便业务使用
-  多线程架构，充分利用多核cpu性能
-  海量存储容量
-  高性能低毛利
-  业务数据安全

Istore-ssd 运营数据

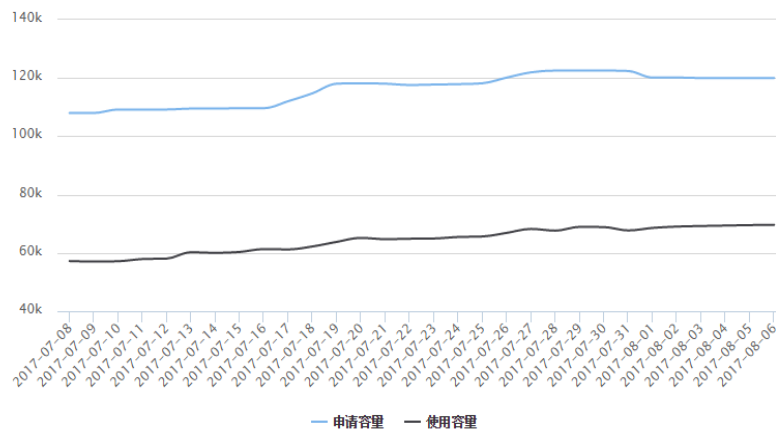
🌐 **400+** 机器
🌐 **800+** 实例
🌐 **70TB+** 数据

🌐 日访问量**1200亿+**
🌐 日写入数据**20TB+**
🌐 峰值qps **300w+/s**

请求量趋势



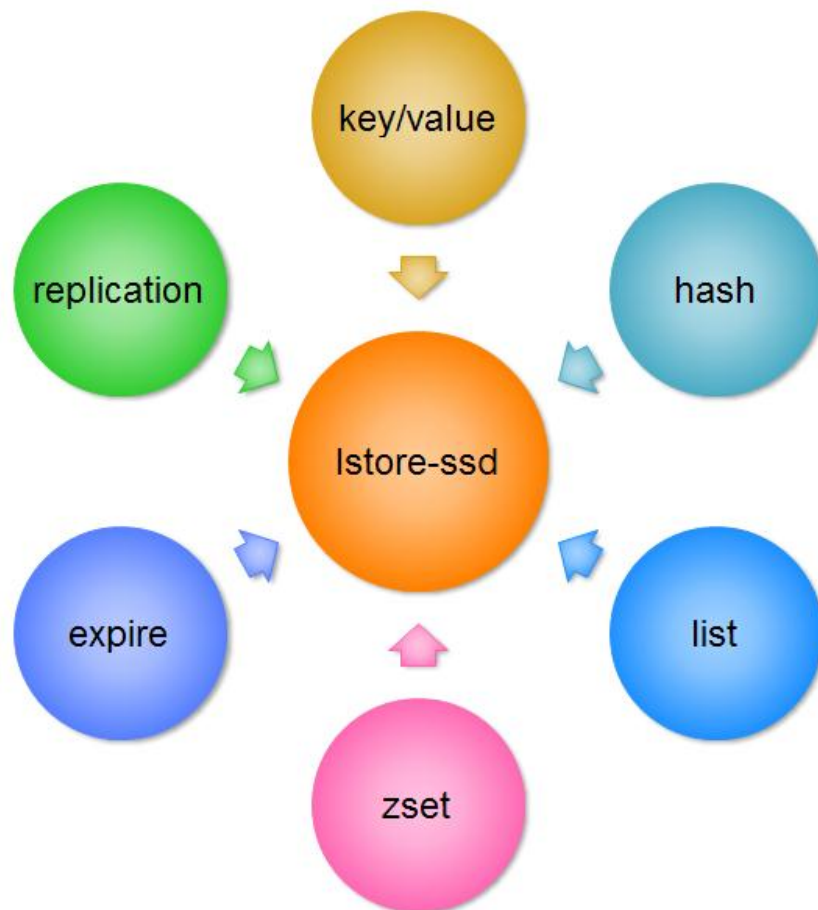
存储容量趋势 (GB)



Istore-ssd vs ssdb

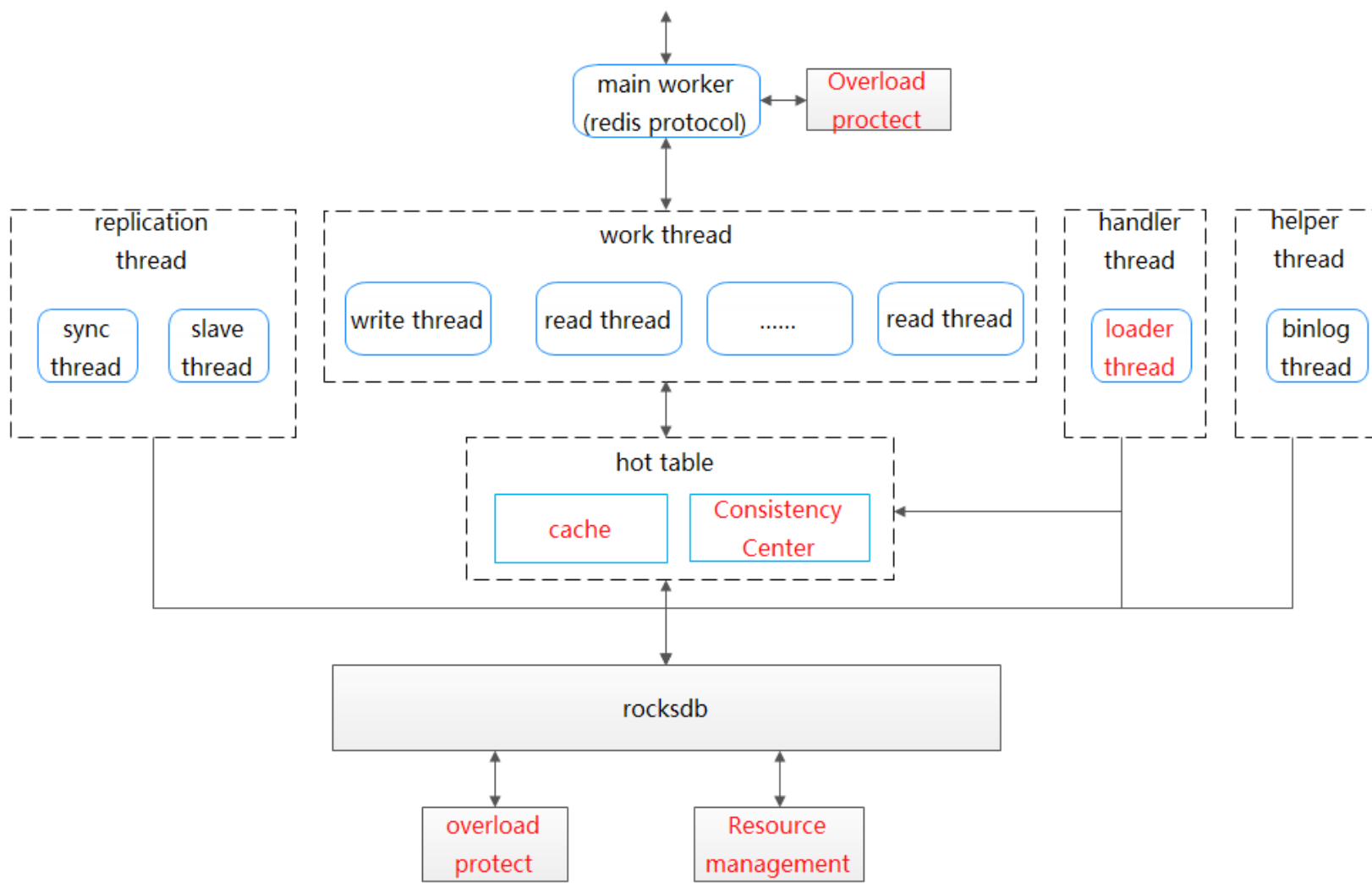
特性	Istore-ssd	ssdb
api支持	c/c++ php java python go 支持redis api	c/c++ php java python go 支持redis api
redis 协议	支持redis二进制协议、私有redis异步协议、redis文本协议	支持redis二进制协议
数据类型	支持string / hash / list / zset 引擎支持expire kv/hash filed	支持string / hash / list / zset 异步支持expire kv
存储引擎	rocksdb	leveldb
分布式	支持	不支持
热数据cache	有	无
redis同步导入数据	支持	不支持
key 统计	支持	不支持
slowlog 监控	支持	不支持
读线程动态调整	支持	不支持
过载保护	支持	不支持
数据安全	写入流水、可配置同步/异步落地方式	异步落地

Istore-ssd 支持功能



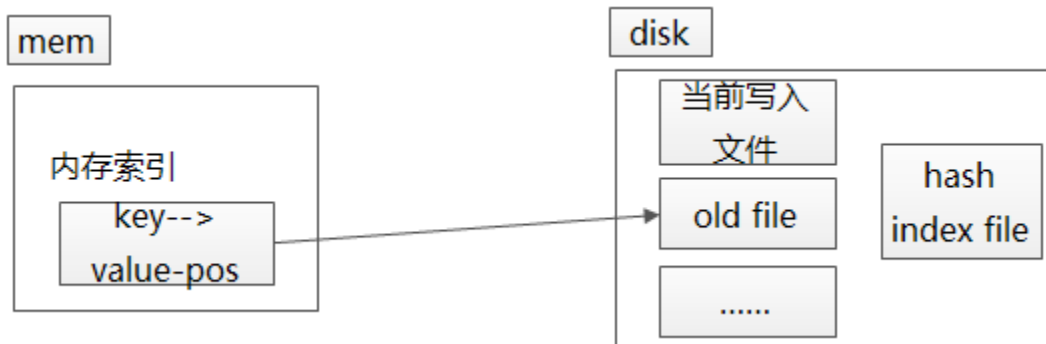
Istore-ssd设计与实现

Istore-ssd 存储架构

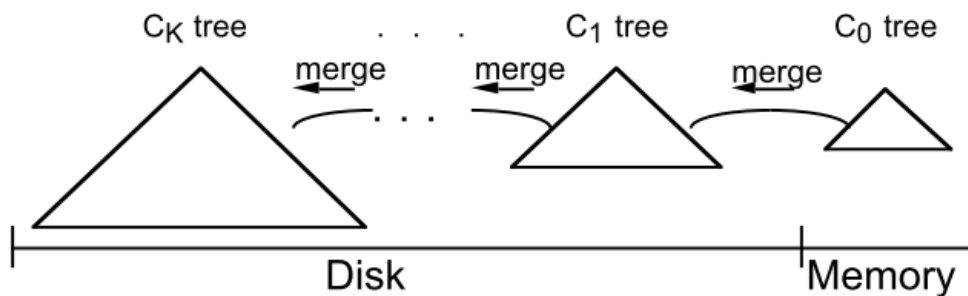


存储模型

bitcask



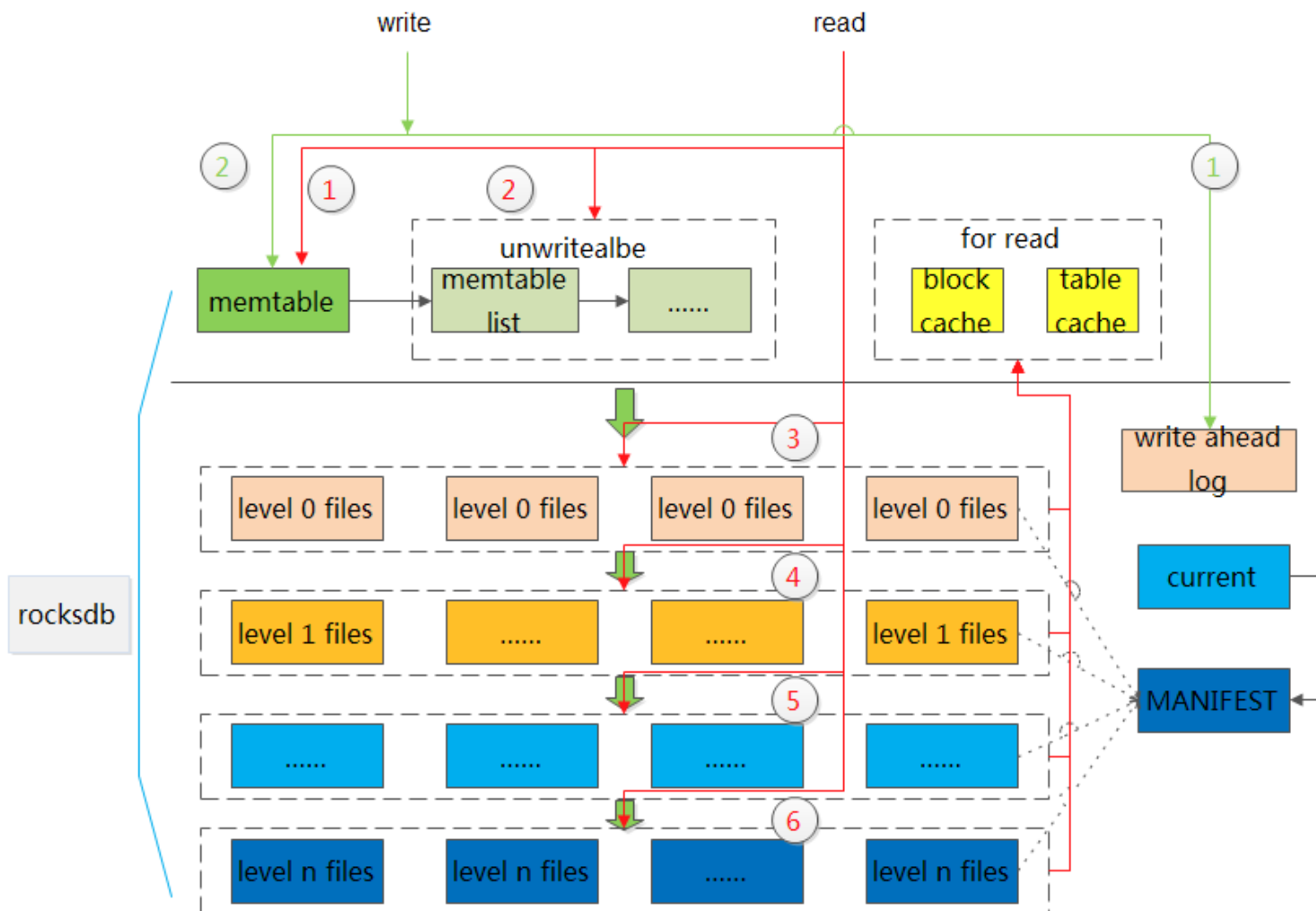
lsmtree



内部实现

- grocery、tssd-- bitcask
- 微信朋友圈--leveldb

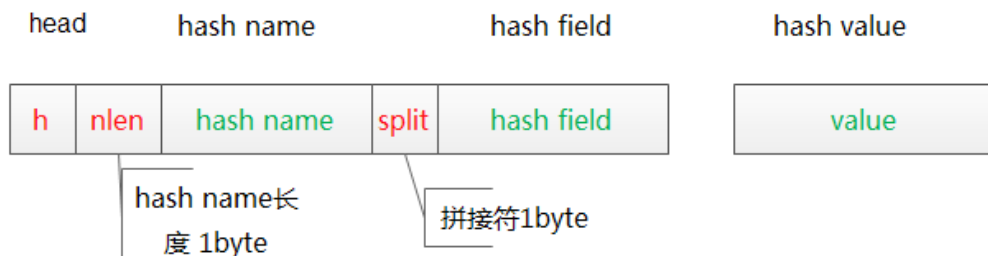
Istore-ssd rocksdb引擎



Istore-ssd 主要类型

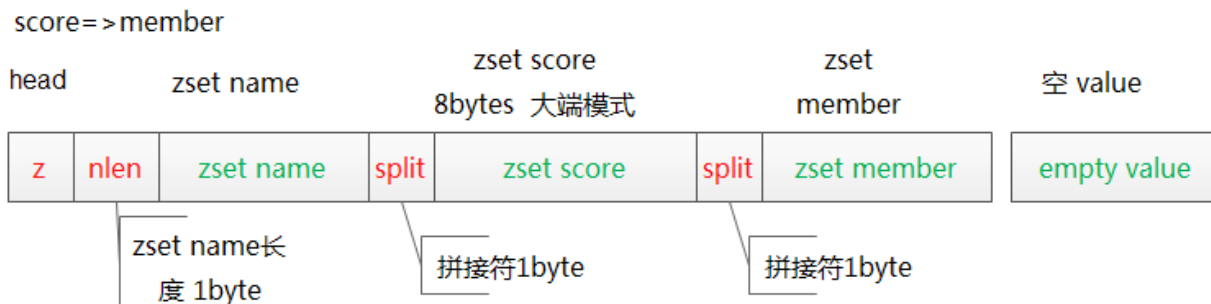
hash

- hash-field个数
- hash名字遍历

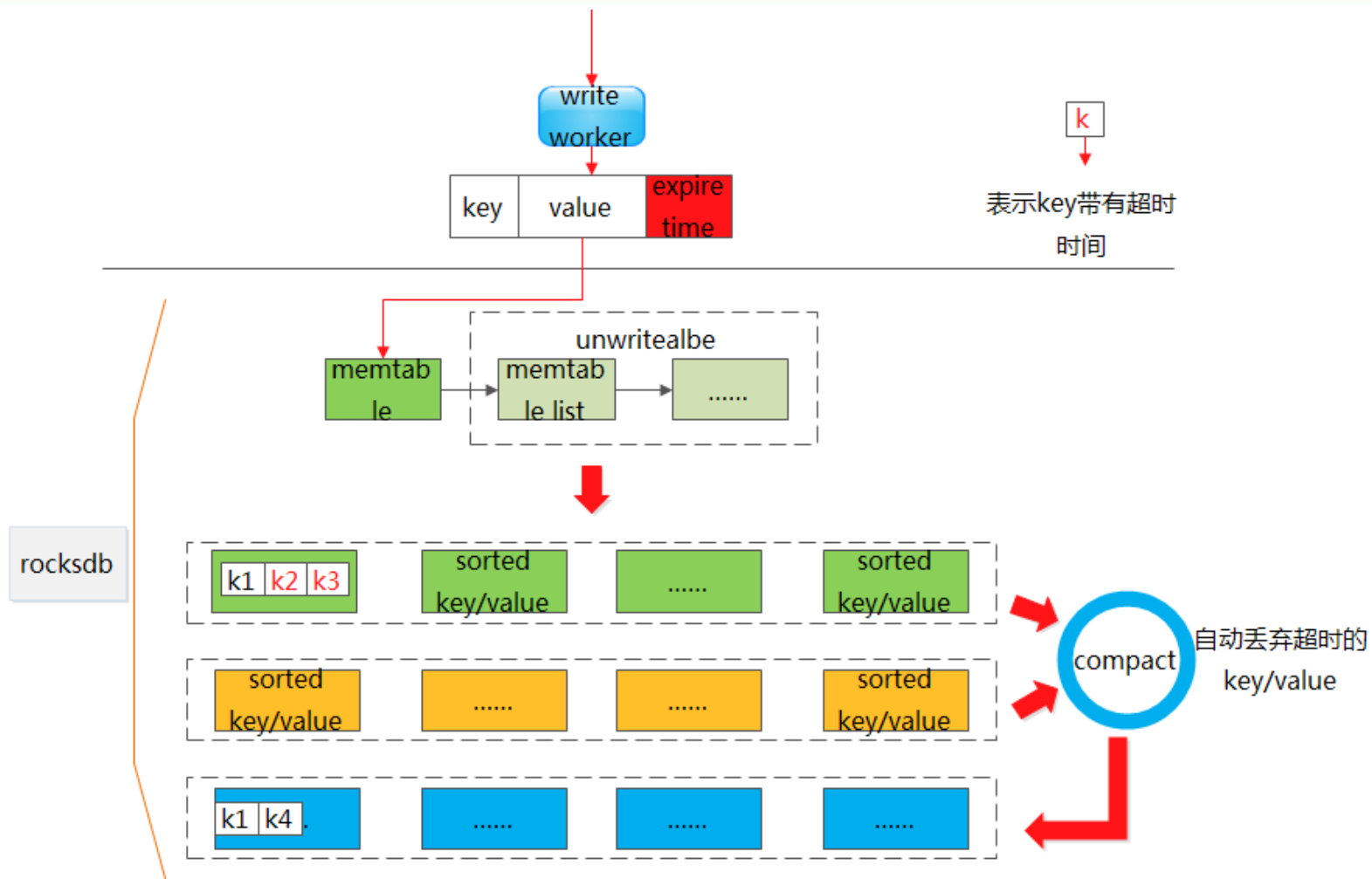


zset

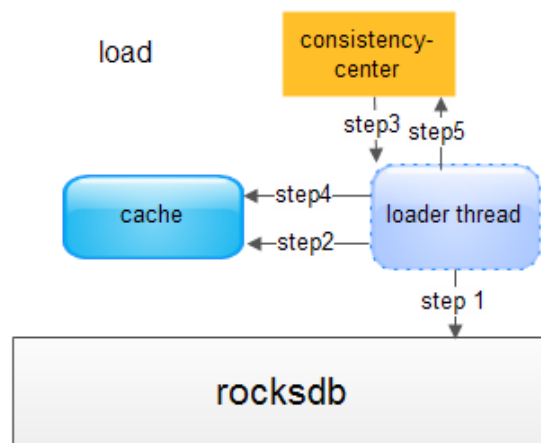
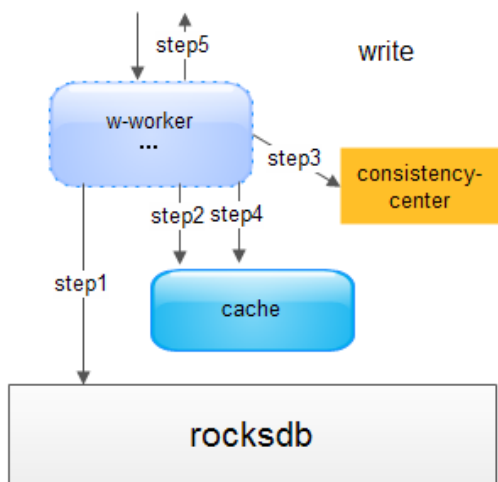
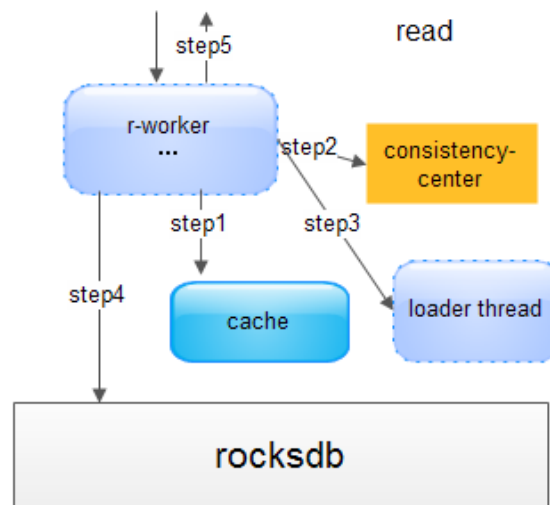
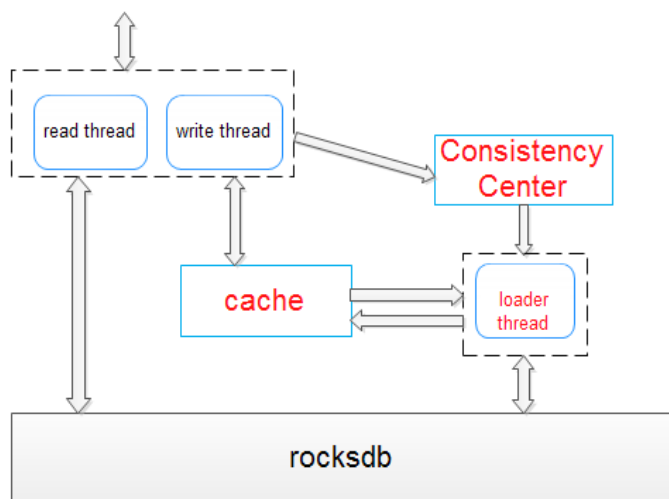
- member->score 查找
- score->member 遍历



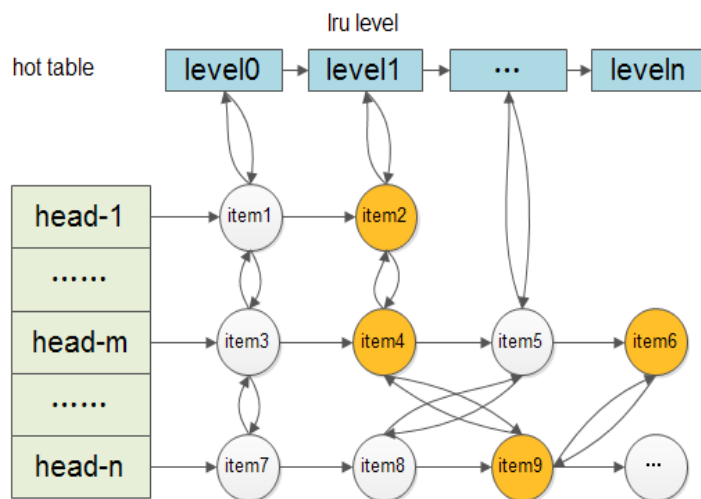
Istore-ssd 超时机制



Istore-ssd 内建cache



Istore-ssd 热表实现



table设计

```
class ShardedReqLRUCache
{
    shard_bits_;
    capacity_;
    ...

    ReqLRUCache *shards_
};
```

ReqLRUCache 0
ReqLRUCache 1
ReqLRUCache 2
...
...
...
ReqLRUCache n-1
ReqLRUCache n

最小cache单元

```
class ReqLRUCache{
    mutex_;
    usage_;
    ...

    ReqLRUHandle lru_[LRU_LIST_SIZE]
    ReqHandleTable table_;
};
```

```
struct ReqLRUHandle
{
    void *value;
    ReqLRUHandle *next_hash;
    ReqLRUHandle *next;
    ReqLRUHandle *prev;
    uint8_t type;//value type
    uint8_t level;//lru level
    uint8_t flags; uint8_t refs;
    uint16_t key_length;
    uint16_t req_cnt;
    uint32_t charge; uint32_t hash;
    uint64_t expire_time;
    char key_data[0];
};
```


Istore-ssd 数据同步1

主备同步

- copy阶段: last_key
- sync阶段: last_seq

mirror同步

- 连接类型

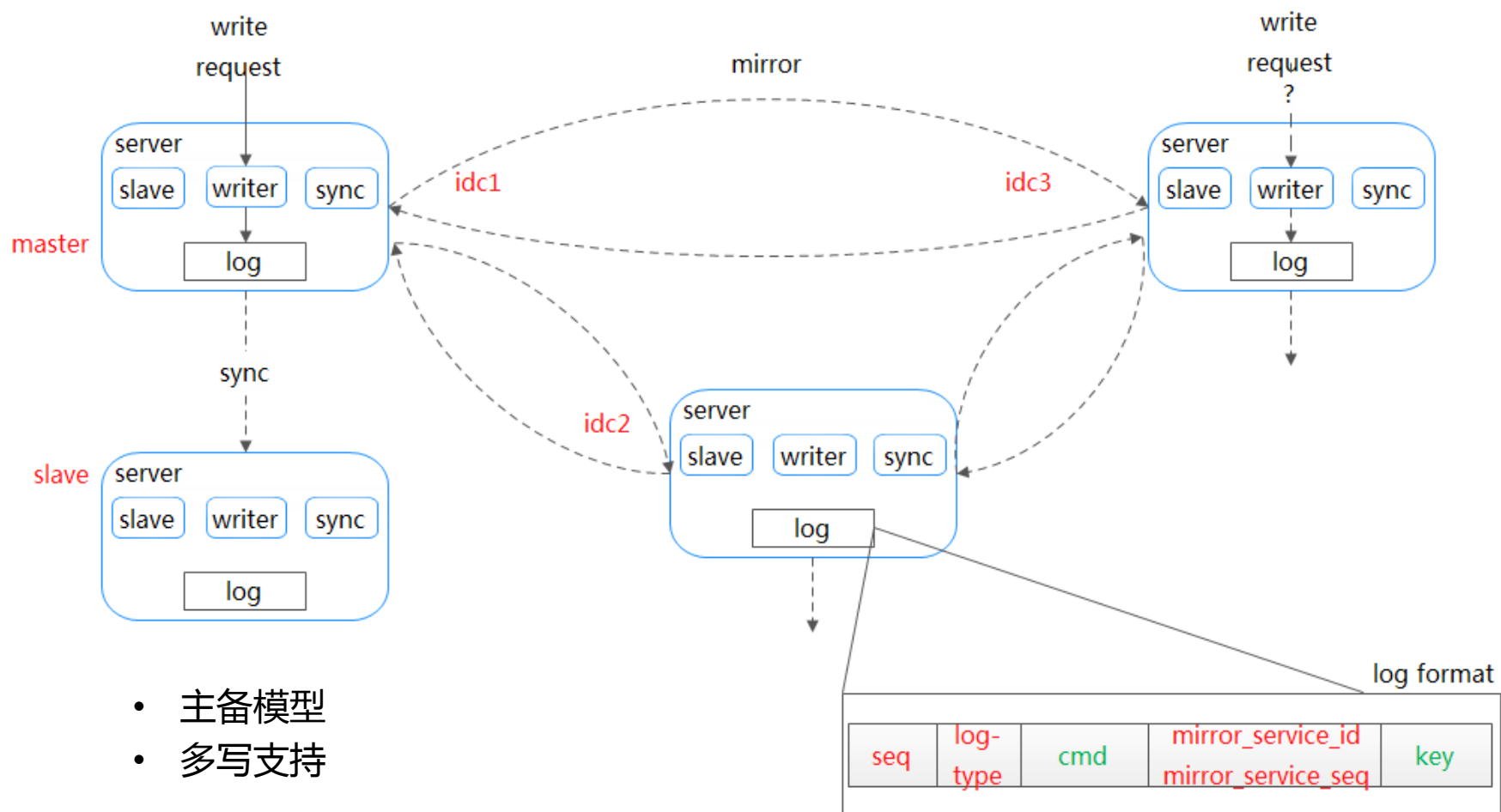
binlog 结构:

binlog key 9bytes

binlog value 10+key.size() bytes

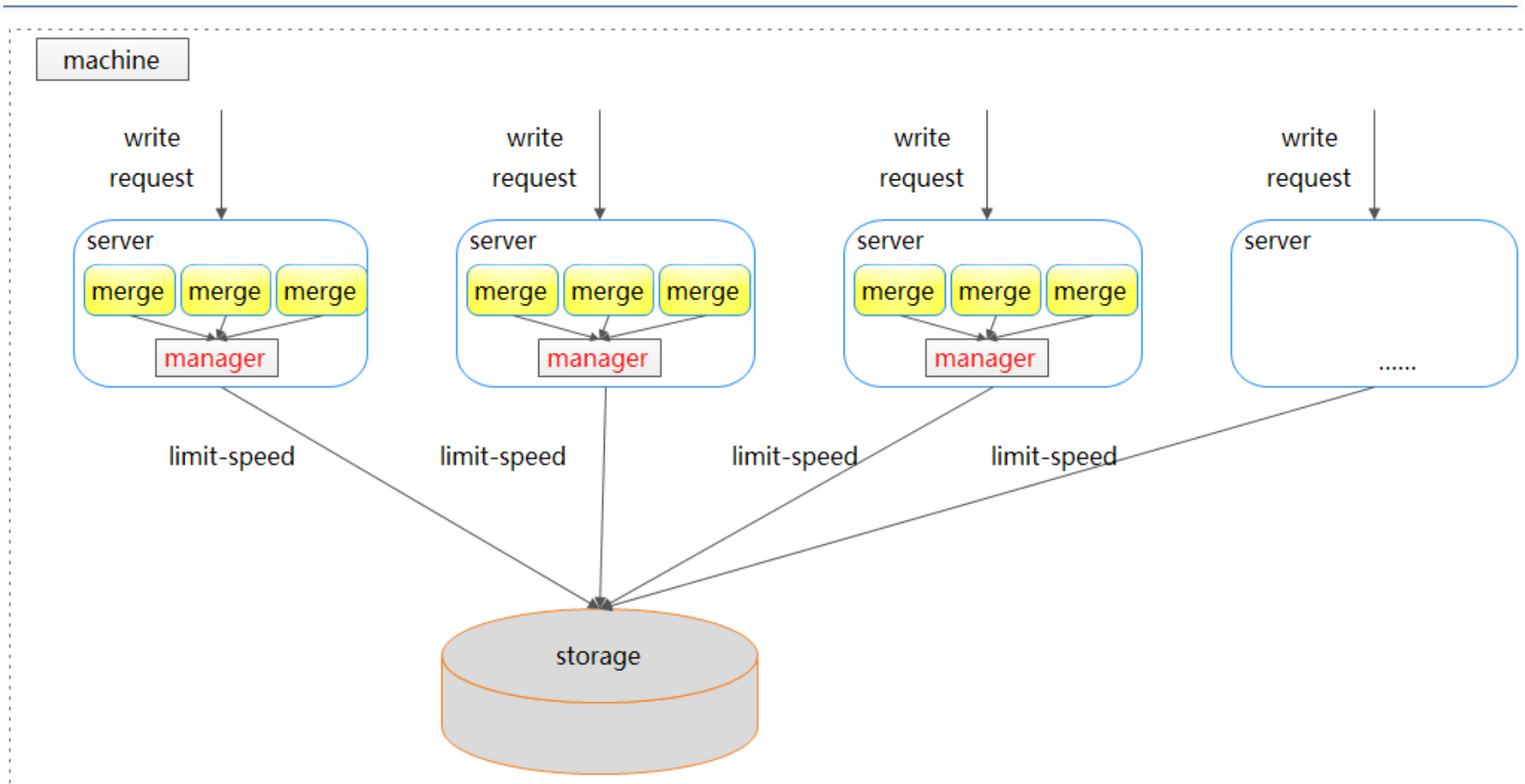


Istore-ssd 数据同步2



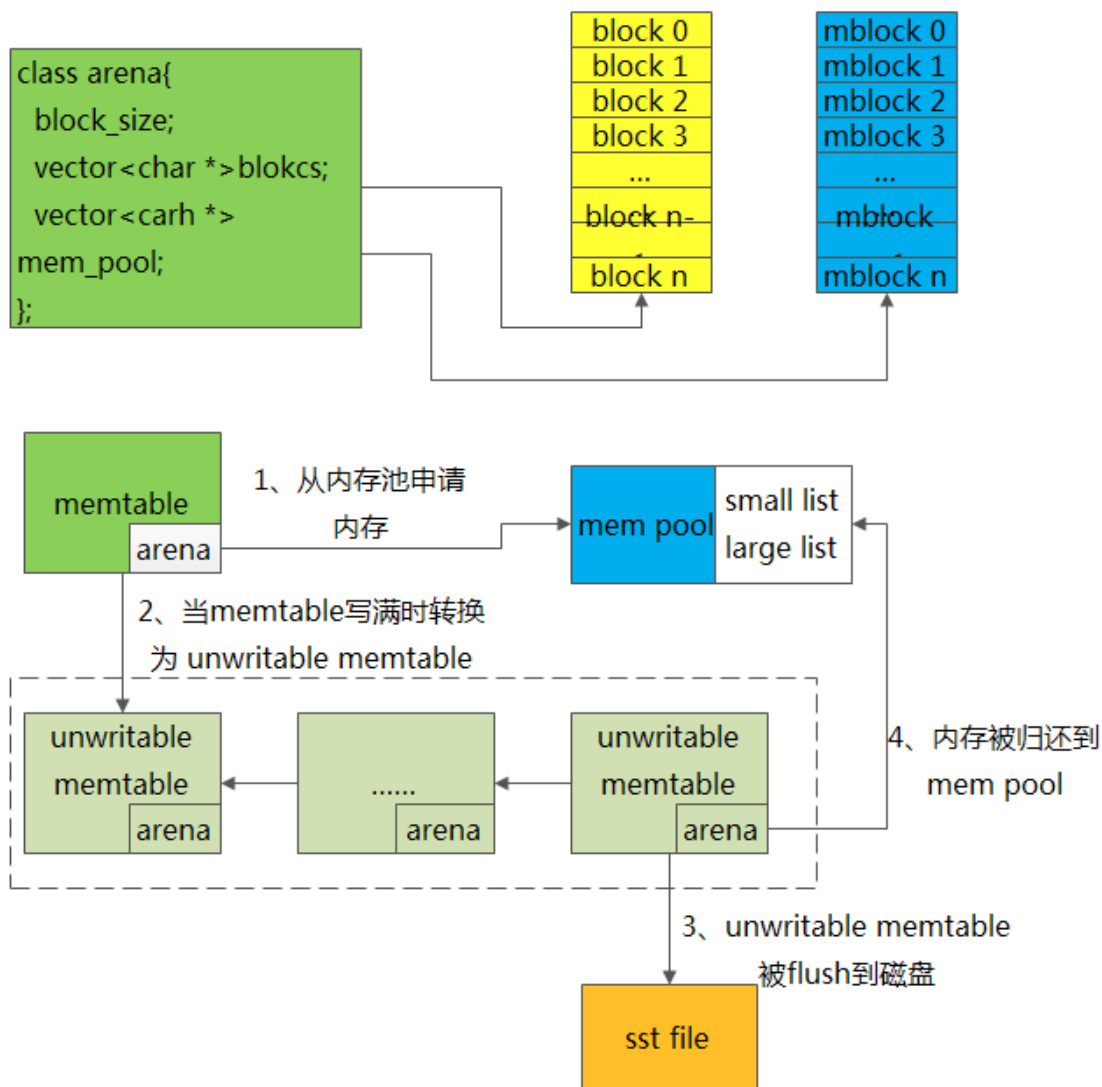
- 主备模型
- 多写支持

Istore-ssd 单机多实例模型



- 降低写对读的影响
- 提升搬迁速度

Istore-ssd 内存使用优化



Istore-ssd 系统参数调整

- 1、min_free_kbytes/extra_free_kbytes
- 2、swappiness / swapoff
- 3、dirty_background_ratio
- 4、dirty_ratio
- 5、numa相关参数zone_reclaim_mode

Istore-ssd 整体优化



Istore-ssd 性能数据

- 1、单机写入速度上限约50MB/s
- 2、访问ssd有一定毛刺，延时高于100ms，几率低于万分之一
- 3、99.9%延时低于50ms，业务平均延时低于1ms

结论：

- 1、Istore-ssd支持更大的弹性，线上运行支撑的访问密度范围（0~100），在业务容量较小时访问密度远超100
- 2、TSSD主要面向冷数据存储，在k/v较小时性能相对较低，能保证数据在容灾切换时一致

Istore-ssd性能数据 -z3





数据长度 (字节)	写性能 (每秒请求量)	读性能 (每秒请求量)	写 + 读 (每秒请求量)
100	70000	70000	30000 + 30000
300	50000	60000	25000 + 25000
1000	30000	50000	15000 + 15000
3000	15000	35000	9000 + 9000
10000	5000	15000	2500 + 2500
30000	2000	6000	1200 + 1200

TSSD性能-ts8



平均记录大小			200B	2K	4K	10K
设备类型	容量	说明	模块单组机器 支撑标准(次/s)			
TS8/TS8-2	1430	get	53000	40000	30000	12000
		set	8000	5000	6750	2750

Istore-ssd 系统优点、不足

优点：

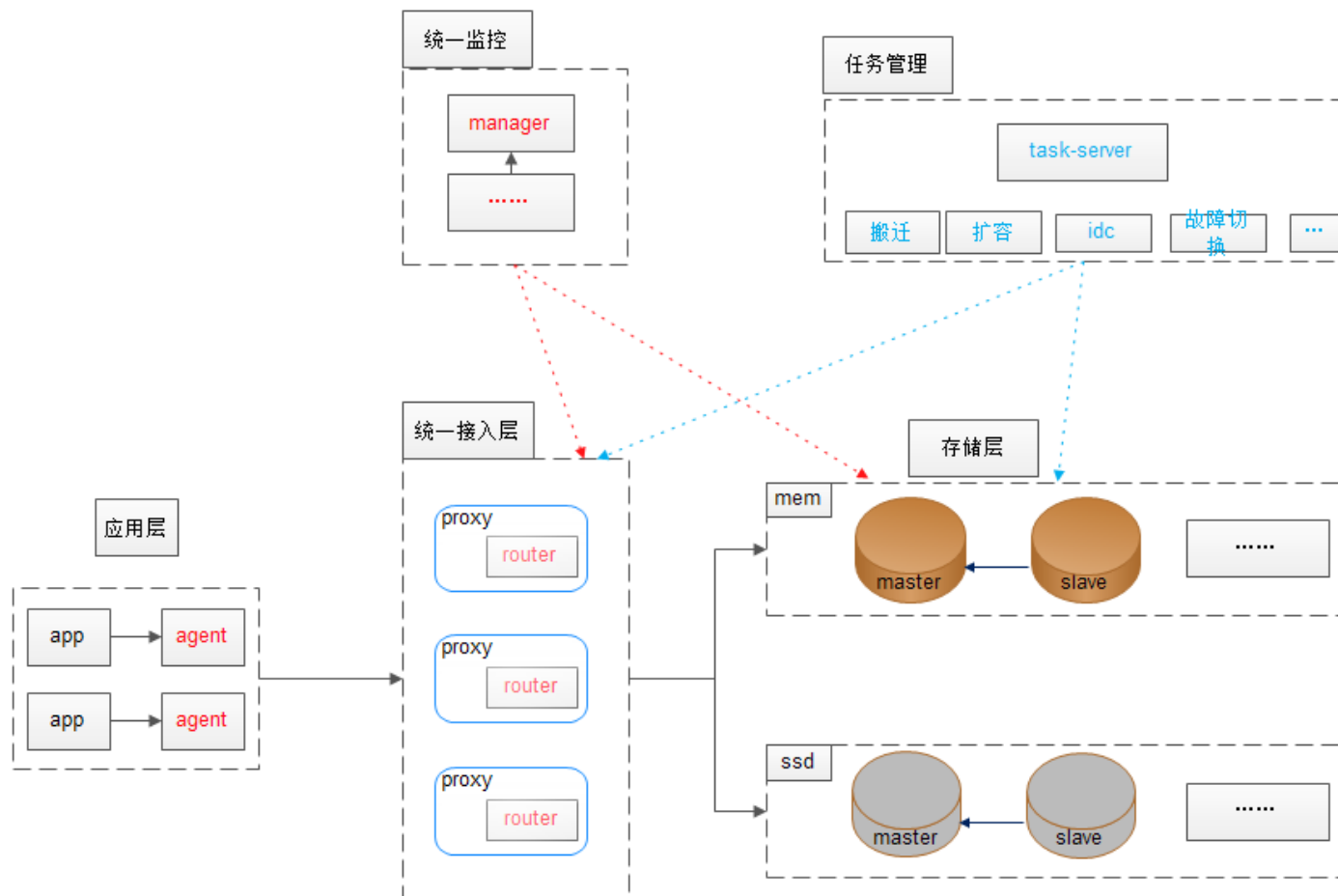
-  多种数据类型
-  不错的性能数据
-  海量存储容量
-  支持多地部署

不足：

-  容灾切换数据一致性问题(paxos/两阶段提交/同步传输/第三方保证)
-  访问毛刺

Istore 集群设计

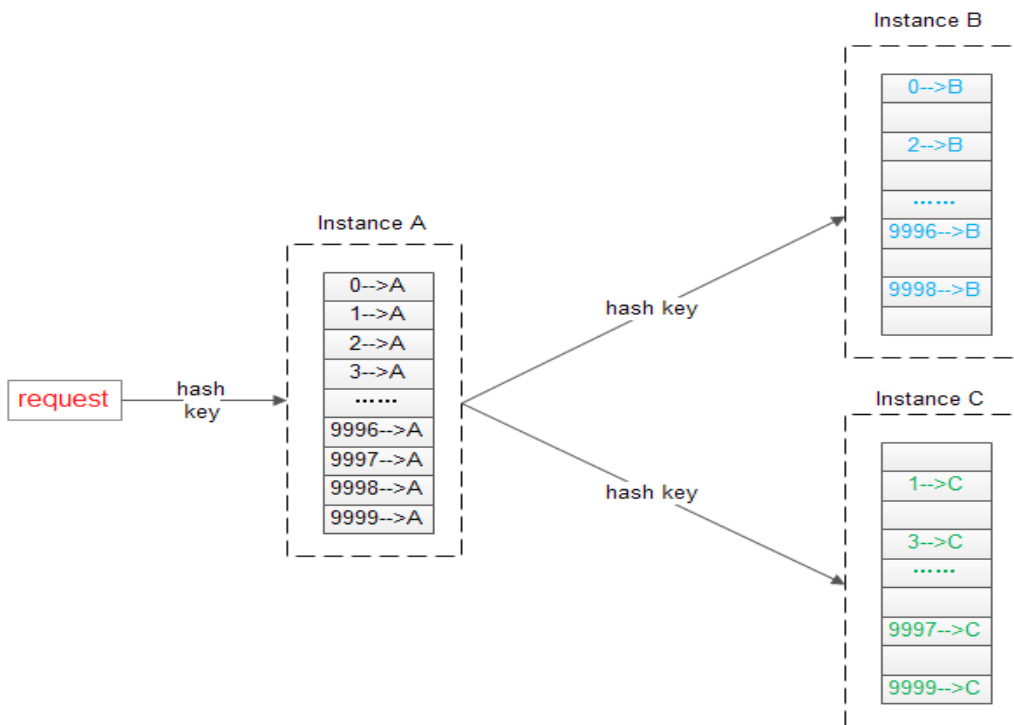
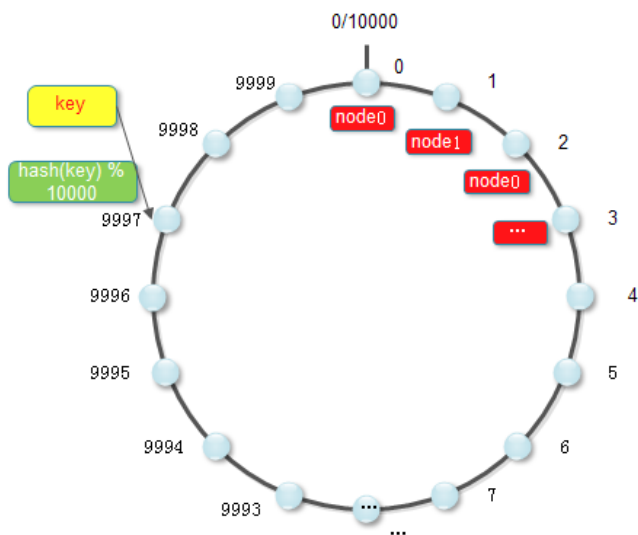
Istore 集群架构



Istore 数据分片

hash slot设计:

- node等差分布
- 限定hash范围



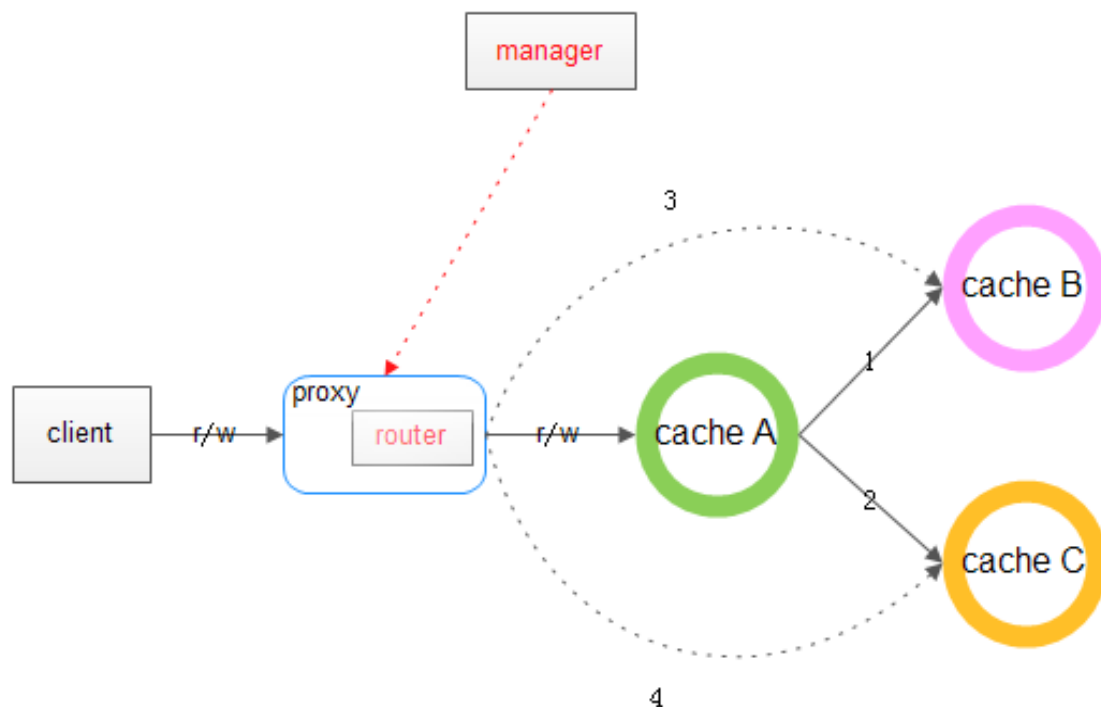
Istore 拆分流程

数据如何拆分

- 源机拆分
- 目标机过滤




路由如何变更

- 先改路由,后迁数据
- 先迁数据,后改路由
- 路由是推送还是拉



Istore 元数据管理

当前实现（中心化）



-  元数据存储于CDB，方便运营管理
-  manager无状态设计,双机探测
-  路由版本通知，无性能瓶颈

Next step



发展

高写入专用存储

-  1、写入能力提升，将部分level置于tmpfs中
-  2、新硬件探索，使用ts80/3dxdpoint存储

强一致性存储

谢谢！