BeansDB 设计与实现

刘洪清
Davies@douban.com
2011/4/8

豆瓣 douban

- 需求
- 设计
- 实现
- 回顾

产品需求

产品需求

- 图片类应用
  - 个人相册,群体相册(线上活动), 图片墙
  - FM

产品需求

- 图片类应用
  - 个人相册,群体相册(线上活动), 图片墙
  - FM
- 文本类应用
  - 评论,日记,作品,博客

技术需求

技术需求

- 存储大量小文件(字段)

技术需求

- 存储大量小文件(字段)
- 可扩展性

技术需求

- 存储大量小文件(字段)
- 可扩展性
- 可靠性

技术需求

- 存储大量小文件(字段)
- 可扩展性
- 可靠性
- 可用性

技术需求

- 存储大量小文件(字段)
- 可扩展性
- 可靠性
- 可用性
- 一致性

技术需求

- 存储大量小文件(字段)
- 可扩展性
- 可靠性
- 可用性
- 一致性
- 低成本

两个原则

两个原则

- **KISS**

豆瓣 douban

两个原则

- **KISS**

豆瓣 douban

两个原则

- **KISS**
- 尽量复用开源方案

豆瓣 douban

开源项目参考

豆瓣 douban

开源项目参考

- **MogileFS**

开源项目参考

- **MogileFS**
- **Dynamo**

豆瓣 douban

开源项目参考

- **MogileFS**
- **Dynamo**
- **Memcached**

豆瓣 douban

开源项目参考

- **MogileFS**
- **Dynamo**
- **Memcached**
- **Riak**

豆瓣 douban

初步方案

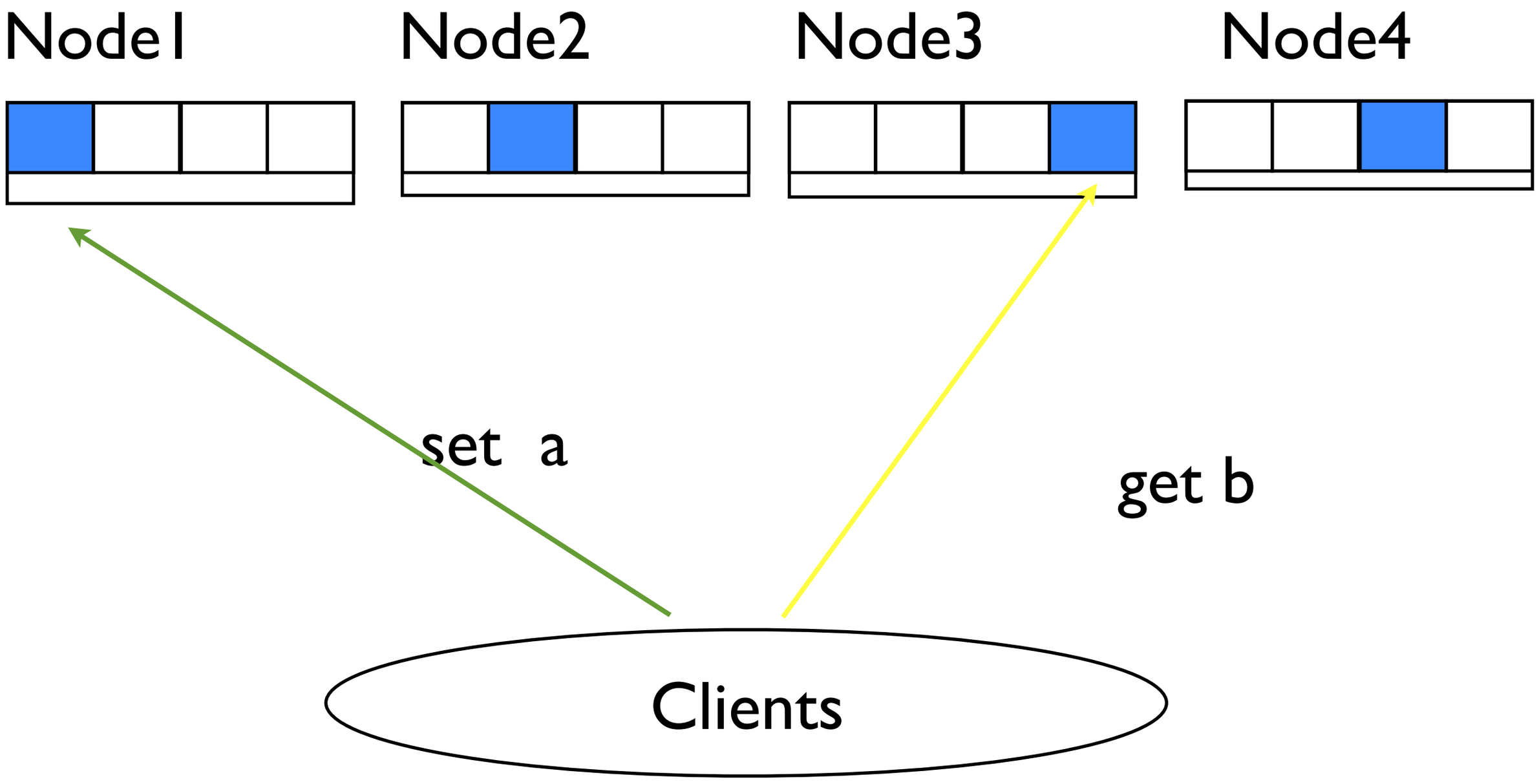
初步方案

- 存储大量小文件
  - key/value, get/set/delete
  - hash, bucket

初步方案

- 存储大量小文件
  - key/value, get/set/delete
  - hash, bucket
- 可扩展性
  - 独立存储节点,独立数据目录
  - bucket 扩展

初步方案



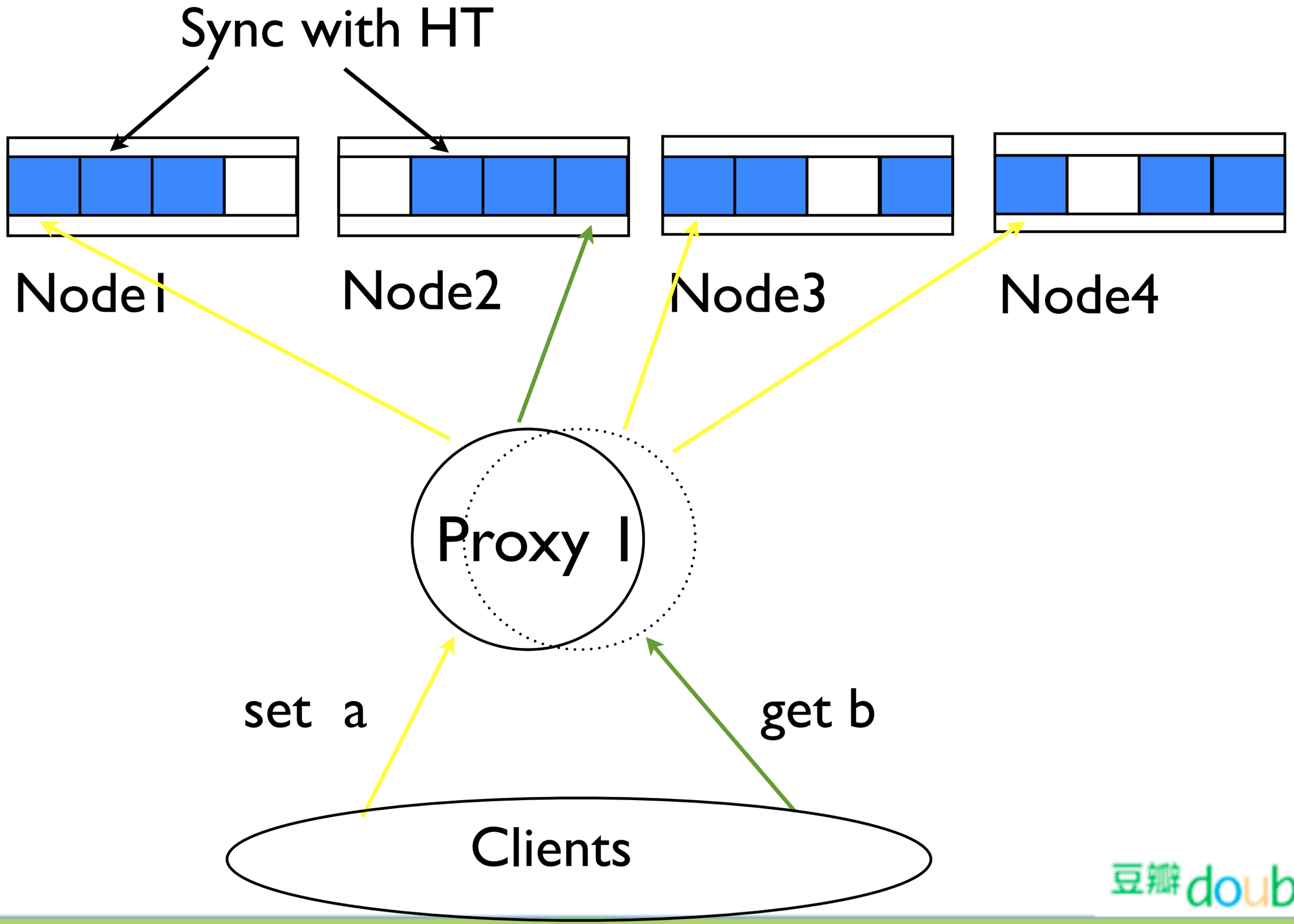
深入设计

深入设计

- 可靠性
  - 多机冗余(N=3)
  - 同步写(W=2), 依次读(R=1)

深入设计

- 可靠性
  - 多机冗余(N=3)
  - 同步写(W=2), 依次读(R=1)
- 一致性
  - 最终一致性
  - Hash Tree 同步 (外部脚本)
  - 冲突: 独立版本号+更新时间


Sync with HT
Node1
Node2
Node3
Node4
Proxy I
set a
get b
Clients

存储节点实现

豆瓣douban

存储节点实现

- 数据存储引擎
  - Key/Value, ACID(宽松)

存储节点实现

- 数据存储引擎
  - Key/Value, ACID(宽松)
- 网络协议
  - memcache 协议

豆瓣 douban

存储节点实现

- 数据存储引擎
  - Key/Value, ACID(宽松)
- 网络协议
  - memcache 协议
- 同步接口
  - 哈希树

豆瓣 douban

存储节点实现

- 数据存储引擎
  - Key/Value, ACID(宽松)
- 网络协议
  - memcache 协议
- 同步接口
  - 哈希树
- 线程模型

豆瓣 douban

存储引擎

豆瓣 douban

存储引擎

- **TokyoCabinet**
  - 简单, 轻量, 小数据时性能不错
  - crash, 大数据量时性能不好

存储引擎

- **TokyoCabinet**
  - 简单, 轻量, 小数据时性能不错
  - crash, 大数据量时性能不好
- **BerkeleyDB**
  - 过重, 存储稍大的文件性能不好

豆瓣 douban

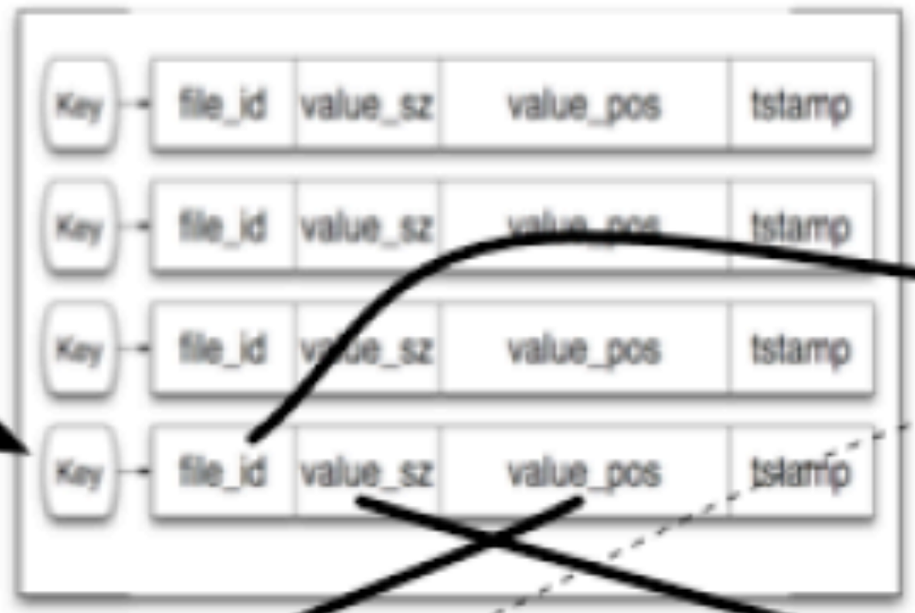
存储引擎

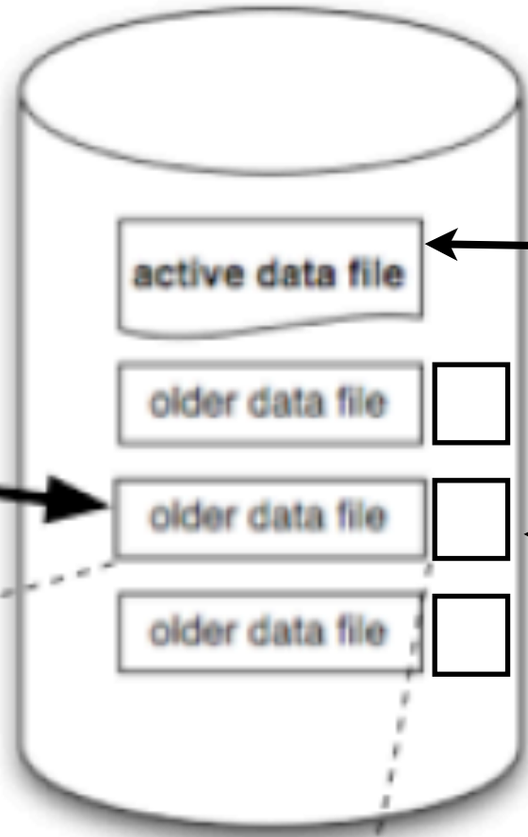
- **TokyoCabinet**
  - 简单, 轻量, 小数据时性能不错
  - crash, 大数据量时性能不好
- **BerkeleyDB**
  - 过重, 存储稍大的文件性能不好
- **Bitcask**
  - 内存索引 + 日志结构数据文件

豆瓣 douban

get(Key)

内存索引





追加写入

hint

tstamp	ksz	value_sz	value_pos	key
tstamp	ksz	value_sz	value_pos	key
tstamp	ksz	value_sz	value_pos	key
tstamp	ksz	value_sz	value_pos	key
tstamp	ksz	value_sz	value_pos	key
tstamp	ksz	value_sz	value_pos	key
tstamp	ksz	value_sz	value_pos	key
tstamp	ksz	value_sz	value_pos	key
tstamp	ksz	value_sz	value_pos	key

1 2 3 4

数据格式

crc	tstamp	ksz	value_sz	key	value
crc	tstamp	ksz	value_sz	key	value
crc	tstamp	ksz	value_sz	key	value
crc	tstamp	ksz	value_sz	key	value
crc	tstamp	ksz	value_sz	key	VALUE
crc	tstamp	ksz	value_sz	key	value
crc	tstamp	ksz	value_sz	key	value
crc	tstamp	ksz	value_sz	key	value
crc	tstamp	ksz	value_sz	key	value
crc	tstamp	ksz	value_sz	key	value

Bitcask

豆瓣 douban

优化Bitcask

豆瓣douban

优化Bitcask

- **每bucket一个Bitcask**

优化Bitcask

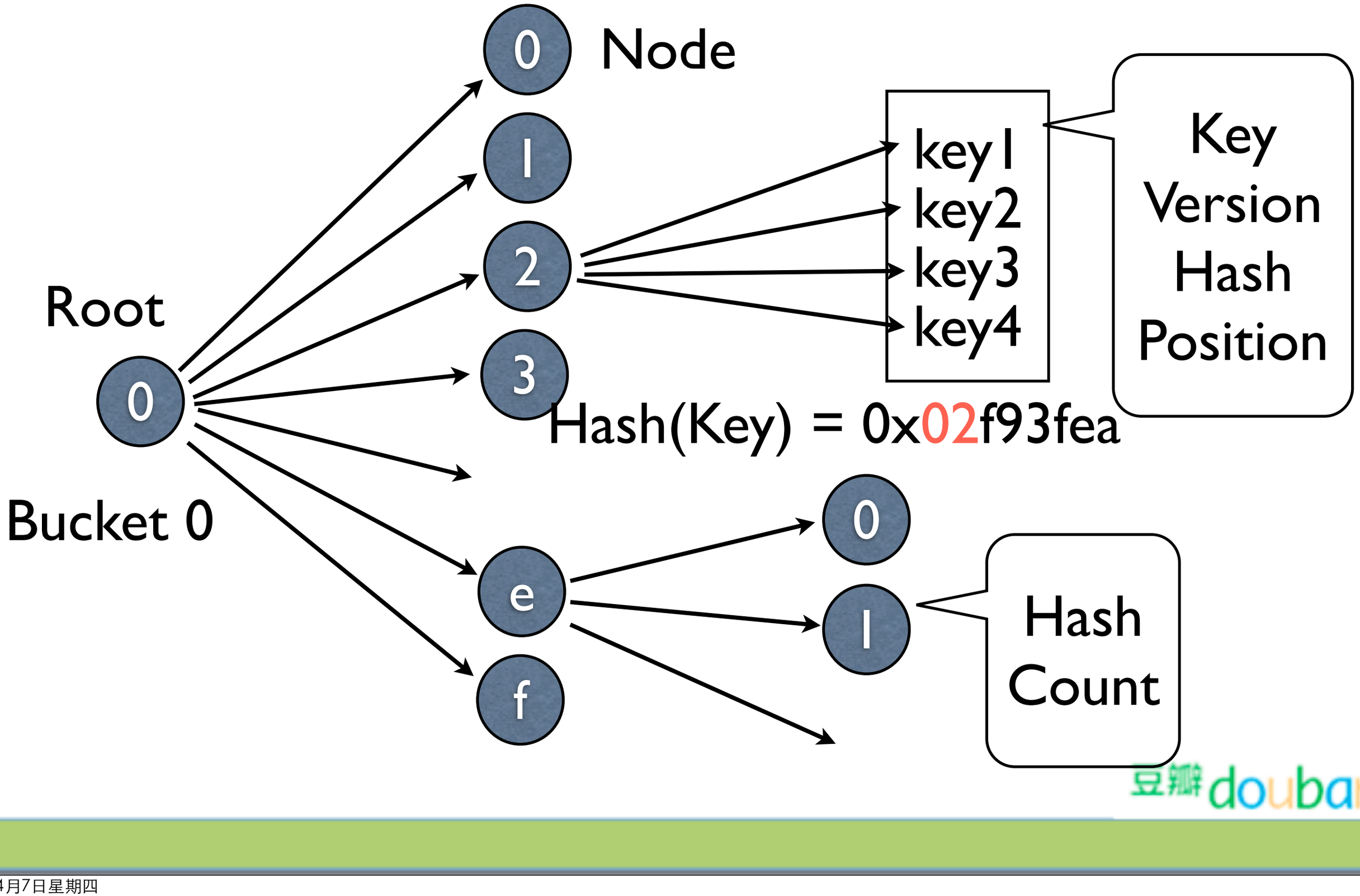
- 每**bucket**一个**Bitcask**
- 数据空间
  - 256 对齐, 最多256个, 自动压缩
  - 在线GC, 外部控制

优化Bitcask

- 每**bucket**一个**Bitcask**
- 数据空间
  - 256 对齐, 最多256个, 自动压缩
  - 在线GC, 外部控制
- 启动时间
  - hint 文件, QuickLZ L3压缩
  - 多线程

豆瓣 douban

Hash Tree



Hash Tree优化

豆瓣 douban

Hash Tree优化

- **Hash Table**

Hash Tree优化

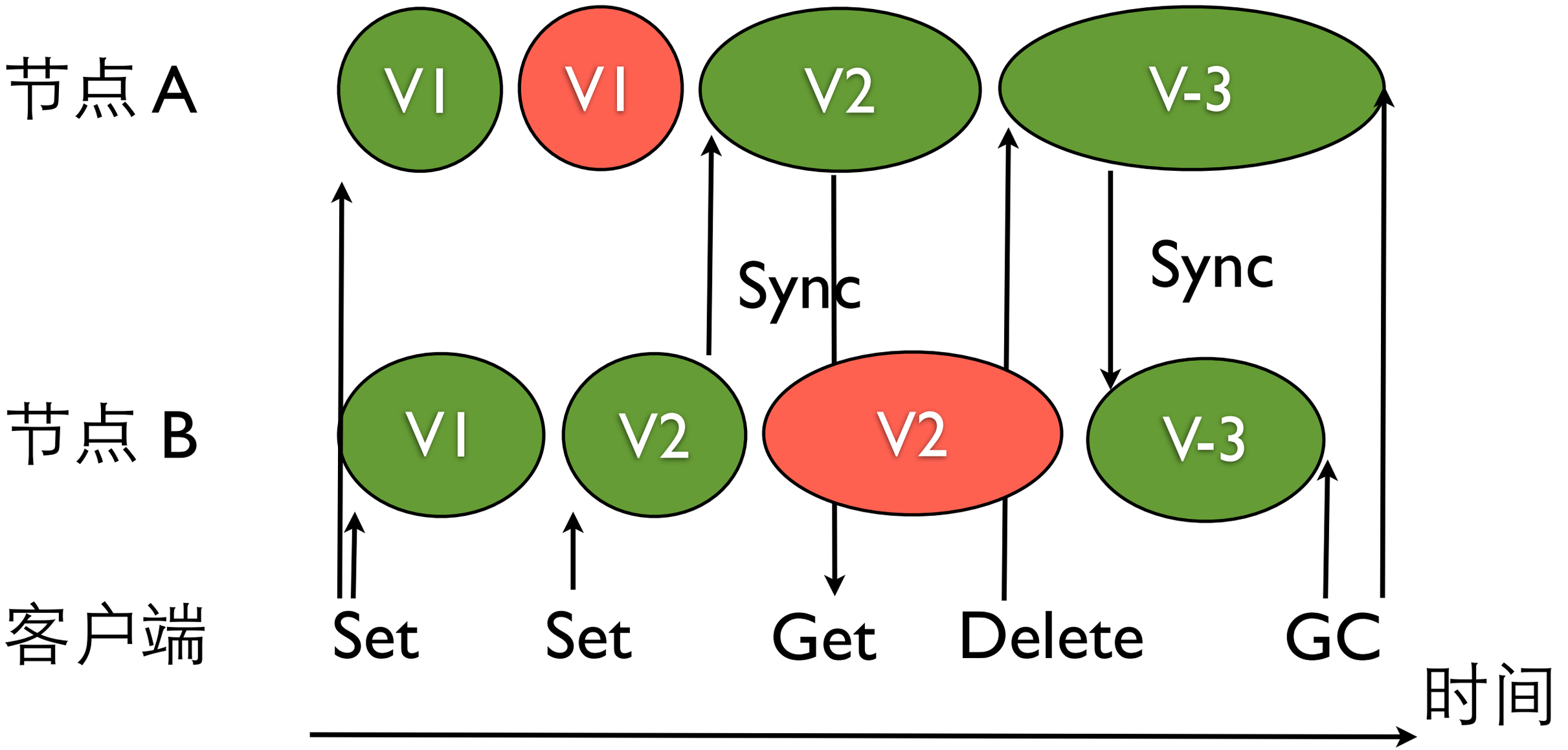
- **Hash Table**
- **Node, Item** 连续存放
  - {1}{16}{256}
  - {Size, Count, [{Ver, Hash, Pos, Key}, ...]
  - 128 个自动分裂

豆瓣 douban

Hash Tree优化

- **Hash Table**
- **Node, Item** 连续存放
  - {1}{16}{256}
  - {Size, Count, [{Ver, Hash, Pos, Key}, ...]
  - 128 个自动分裂
- 重新编码**key**
  - /topic/1234567/body=>{/topic/%d/body, 123}

同步过程


节点 A
V1
V1
V2
V-3
Sync
Sync
节点 B
V1
V2
V2
V-3
客户端
Set
Set
Get
Delete
GC
时间

网络协议

网络协议

- **memcache** 文本协议
  - 复用memcached的网络协议层代码
  - 有大量客户端可用

豆瓣 douban

网络协议

- **memcache** 文本协议
  - 复用memcached的网络协议层代码
  - 有大量客户端可用
- 适当扩展
  - @fff, ?xxxx, flush_all
  - get_multi, set_multi, delete_multi

豆瓣 douban

IO与多线程模型

豆瓣 douban

IO与多线程模型

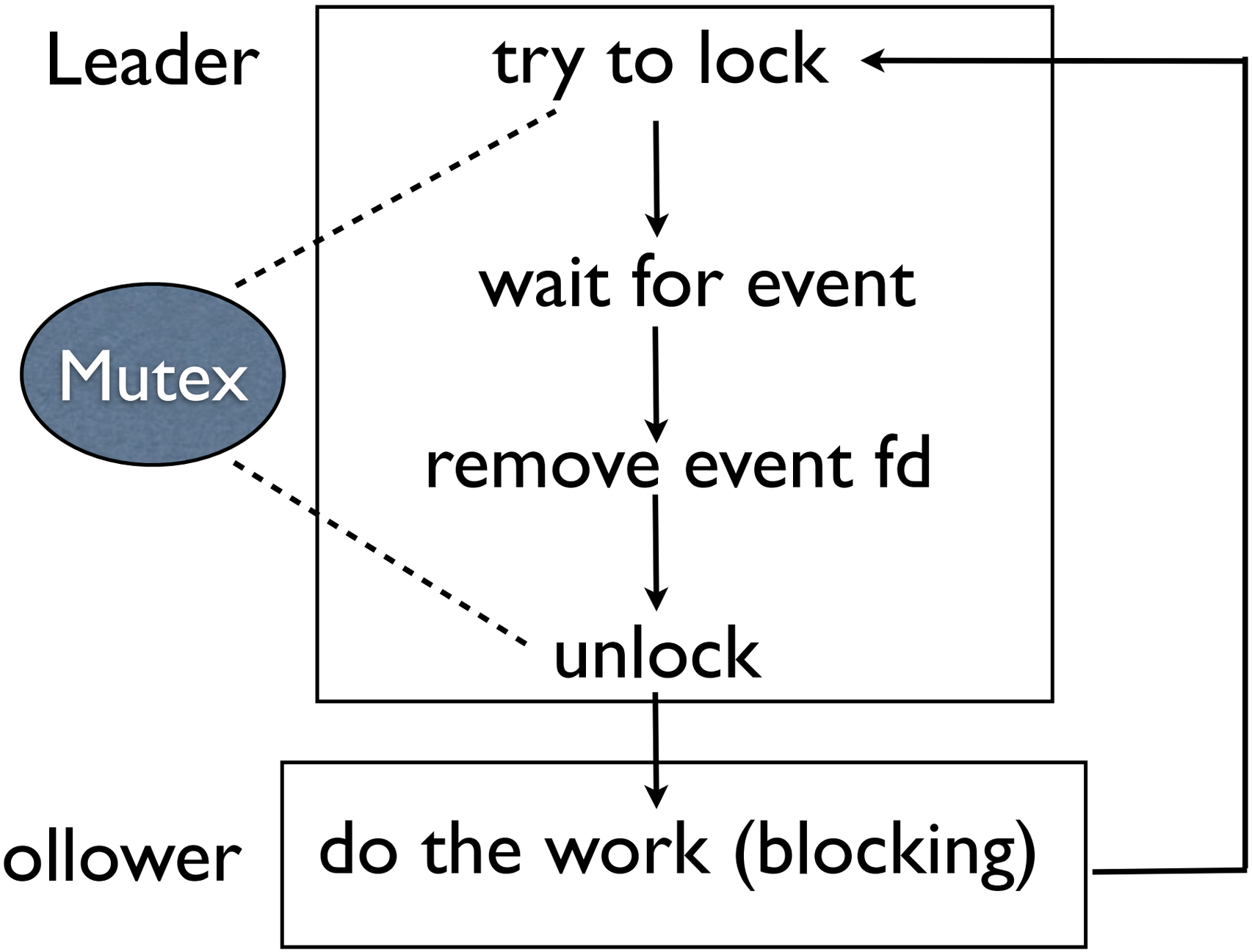
- 并发与IO
  - 并发连接多, 并发请求少
  - 异步网络IO, 同步磁盘IO

IO与多线程模型

- 并发与IO
  - 并发连接多, 并发请求少
  - 异步网络IO, 同步磁盘IO
- 多线程模型
  - 连接/线程绑定
  - 半同步/半异步
  - leader/follower

豆瓣 douban

Leader/Follower模型



Proxy 实现

豆瓣 douban

Proxy 实现

- **Go 实现**
  - 易于实现高并发应用, 性能可接受

豆瓣 douban

Proxy 实现

- **Go 实现**
  - 易于实现高并发应用, 性能可接受
- **自动路由**
  - 根据 Merkle Tree得到数据分布
  - 更高可用性

Proxy 实现

- **Go 实现**
  - 易于实现高并发应用, 性能可接受
- **自动路由**
  - 根据 Merkle Tree得到数据分布
  - 更高可用性
- **负载均衡**
  - 减少IO慢的节点的请求量

豆瓣 douban

版本历史

版本历史

- **0.1, like MogileFS　　2008.8**
  - WebDAV, inotify, sync, client

版本历史

- **0.1, like MogileFS 2008.8**
  - WebDAV, inotify, sync, client
- **0.2, like TokyoTrant 2008.12**
  - TC, HTree, sync, mc client

豆瓣 douban

版本历史

- **0.1, like MogileFS 2008.8**
  - WebDAV, inotify, sync, client
- **0.2, like TokyoTrant 2008.12**
  - TC, HTree, sync, mc client
- **0.3, like memcachedb 2009.6**
  - HTree in TC, on Disk

豆瓣 douban

版本历史(2)

版本历史(2)

- **0.4** **2010.2**
  - proxy

版本历史(2)

- **0.4** **2010.2**
  - proxy
- **0.5 reload** **2010.10**
  - Bitcask, Leader/Follower

原型开发

原型开发

- 原型开发
  - 快速验证想法

原型开发

- 原型开发
  - 快速验证想法
- **Python**
  - Cython 扩展

原型开发

- 原型开发
  - 快速验证想法
- **Python**
  - Cython 扩展
- **Go**
  - cgo 扩展, 容易开发高并发应用

豆瓣 douban

实际部署案例(I)

实际部署案例(1)

- 数据量
  - 图片, 音频: 1k到20M
  - 310M x 3 = 930 M
  - 590G x 16 x 3 = 28T
  - 17 个节点, 约 50 块SATA硬盘

实际部署案例(1)

- 数据量
  - 图片, 音频: 1k到20M
  - 310M x 3 = 930 M
  - 590G x 16 x 3 = 28T
  - 17 个节点, 约 50 块SATA硬盘
- 性能 :
  - 250 qps左右, 有CDN
  - Med/Avg/90%/99%: 16/29/69/232 ms

豆瓣 douban

各节点数据分布

server	All records in buckets										
	buckets										
	0	1	2	3	4	5	6	7	8	9	1
baggins1:7900						19538558	19544660		19534254		
baggins2:7900				19520627							
baggins3:7900			19524528								
baggins4:7900										19525843	195
baggins5:7900				19520627				19519711			
bifur:7900	19526508	19503579									
bofur:7900		19506719					19543488		19532966		
dori:7900				19520552	19538834			19519643			
dwalin:7902							19544646		19534247	19525831	
eomer:7902											
fili:7900										19521578	195
gimli:7902											
kili:7900			19524412			19538423					
nori:7900	19288985				19299871	19294702					
ori:7900			19520876					19515818			195
theoden:7902		19504461									
thorin:7900	19525640				19537979						

线上运行状态截图 (图片, mp3, 3亿)

host	version	uptime	mem	threads	conn	records(c)	get
baggins1:7900	0.5.3	2894597	3551M	16	68	58155311 / 2316400	50952545 / 32
baggins2:7900	0.5.3	2937606	3565M	16	76	58153813 / 2420324	49629017 / 43
baggins3:7900	0.5.3	2803862	3423M	16	49	58146662 / 1901306	42637992 / 31
baggins4:7900	0.5.3	1295223	2208M	16	66	38737899 / 1767610	17786846 / 32
baggins5:7900	0.5.3	2802912	2479M	16	58	38731622 / 1261877	29735614 / 21
bifur:7900	0.5.3	515035	2786M	16	55	58107089 / 1121785	4538611 / 19
bofur:7900	0.5.3	469781	2735M	16	46	58120628 / 1058139	3107480 / 15
dori:7900	0.5.3	4406992	3565M	16	45	58115802 / 1249844	51214023 / 23
dwalin:7902	0.5.3	2403856	3271M	16	57	58142230 / 1051219	28047323 / 26
eomer:7902	0.5.3	363438	1799M	16	51	38775736 / 543650	2802228 / 13
fili:7900	0.5.3	207000	2236M	16	35	58117083 / 473691	1186319 / 9
gimli:7902	0.5.3	4689003	2152M	16	63	38781077 / 1061328	42176674 / 13
kili:7900	0.5.3	558086	2808M	16	59	58146931 / 1251213	3959253 / 19
nori:7900	0.5.3	1919189	4227M	16	48	77533294 / 710580	21168180 / 12
ori:7900	0.5.3	189662	2796M	16	44	77490185 / 563174	1196371 / 11
theoden:7902	0.5.4	590486	2047M	16	63	38744768 / 861115	6068847 / 23
thorin:7900	0.5.3	4696786	3778M	16	41	58155749 / 1600129	41422354 / 20

set	delete	slow	get/set	hit	read	write
5689809 / 3	1502803 / 0	0.41% / 0.00%	8.96 / 8.52	97% / 83%	2060165M / 644k	186091M / 77
5752355 / 4	1524843 / 0	0.38% / 0.62%	8.63 / 10.05	97% / 88%	2271565M / 907k	190048M / 106
5551398 / 5	1581923 / 0	0.35% / 0.00%	7.68 / 5.79	97% / 87%	1630273M / 820k	170024M / 123
1973789 / 2	494846 / 0	0.29% / 0.00%	9.01 / 12.21	96% / 91%	651352M / 882k	61342M / 71
3795355 / 3	991080 / 0	0.34% / 0.80%	7.83 / 6.30	97% / 84%	1023936M / 320k	115580M / 86
1231902 / 4	305104 / 0	4.83% / 9.12%	3.68 / 4.12	80% / 73%	179596M / 784k	39881M / 118
1156017 / 3	299029 / 0	4.02% / 3.19%	2.69 / 4.36	70% / 57%	112066M / 529k	37540M / 64
8310790 / 4	2301624 / 0	2.57% / 5.59%	6.16 / 5.34	96% / 80%	1881573M / 290k	250278M / 129
4853643 / 3	1269867 / 0	1.09% / 0.66%	5.78 / 7.37	96% / 81%	1052262M / 413k	137172M / 77
595021 / 3	159657 / 0	0.51% / 0.57%	4.71 / 3.39	80% / 83%	107869M / 280k	19572M / 92
517225 / 4	190812 / 0	1.48% / 4.44%	2.29 / 2.29	25% / 24%	21898M / 79k	17573M / 112
6042126 / 2	1581302 / 0	2.12% / 2.48%	6.98 / 4.73	97% / 67%	2148423M / 167k	180420M / 76
1371854 / 4	333833 / 1	7.12% / 9.70%	2.89 / 4.50	78% / 70%	170635M / 707k	44533M / 97
5442700 / 5	1383079 / 1	7.72% / 15.63%	3.89 / 2.31	94% / 47%	839572M / 68k	157633M / 158
614200 / 6	175177 / 1	1.96% / 3.89%	1.95 / 1.85	33% / 42%	20777M / 393k	19370M / 166
944925 / 2	245816 / 0	2.08% / 1.51%	6.42 / 7.77	89% / 80%	234069M / 941k	31177M / 72
9017277 / 3	2342076 / 0	6.60% / 10.08%	4.59 / 5.19	95% / 80%	1592735M / 577k	268129M / 129

实际部署案例(2)

豆瓣 douban

实际部署案例(2)

- 数据量
  - 文本字段: 100到100k
  - 550M x 3 = 1.65 B
  - 50G x 16 x 3 = 2.4T
  - 13 个节点, 约 13 块SATA硬盘

实际部署案例(2)

- 数据量
  - 文本字段: 100到100k
  - 550M x 3 = 1.65 B
  - 50G x 16 x 3 = 2.4T
  - 13 个节点, 约 13 块SATA硬盘
- 性能 :
  - 200 qps左右, 有memcached作缓存
  - Med/Avg/90%/99%: 1/14/15/104 ms

豆瓣 douban

Thanks!

Q/A ?

豆瓣 douban