NOSQL 研发之路

孙立 @ 凤凰网
http://t.ifeng.com/sunli
http://t.sina.com.cn/sunli1223

NOSQL 介绍

- NOSQL=Not Only SQL
- NOSQL 分类
- NOSQL+Mysql+Memcached

参见：http://nosql-database.org/

MYSQL 的不足，让 NOSQL 来弥补

- Mysql 的扩展问题
- Mysql 的 Queycache 问题
- Mysql 的 SQL 解析和协议太重

Tokyocabinet/Tokyotyrant

Tokyo Cabinet 是日本人 Mikio Hirabayashi（平林幹雄）のページ 开发的一款 DBM 数据库（注：大名鼎鼎的 DBM 数据库 qdbm 就是他开发的），该数据库读写非常快。 insert:0.4sec/1000000 recordes(2500000qps)，写入 100 万数据只需要 0.4 秒。 search:0.33sec/1000000 recordes (3000000 qps)，读取 100 万数据只需要 0.33 秒。下图为各种 key-value 数据库读写数据的性能测试，可以看出 Tokyo Cabinet 的速度是非常快的。

Tokyotyrant 是 TC 的网络接口，提供兼容的 memcached 协议， http 协议，还提供更加强大的二进制协议。

因为 Tokyotyrant 的进程名字是 ttserver, 我们习惯性的称其为 ttserver

为什么使用 ttserver

- 08 年开发全站评论系统开始使用
- 高性能
- 支持主从复制
- 兼容 memcached 协议
- 数据文件小
- 备份，增加从库方便

Ttserver 的缺陷和解决办法

- 不支持 memcached 的 flag 和 expire

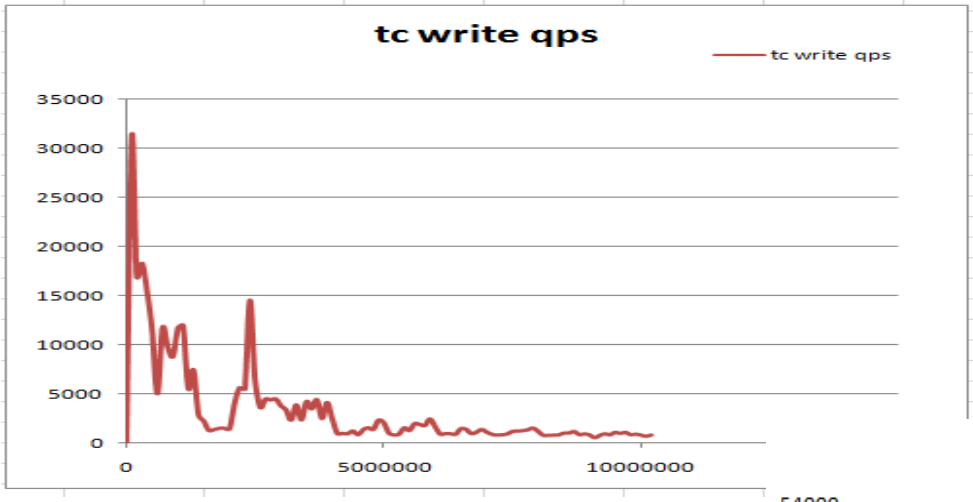
- 大规模出错问题

Ttserver 的问题

- 大数据崩溃，甚至无法重启。

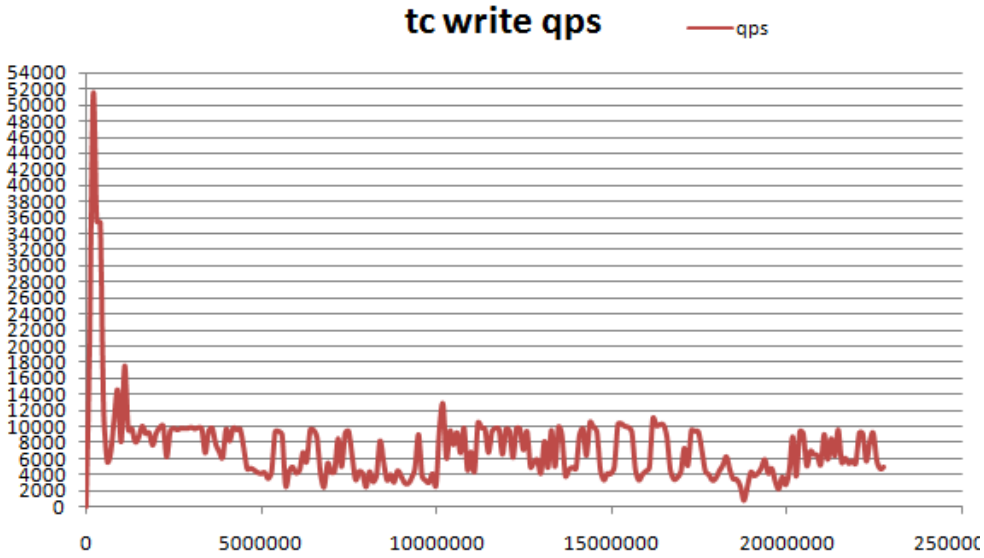
- 单文件，更新随机写严重
- 吞吐量不稳定

Ttserver 的问题



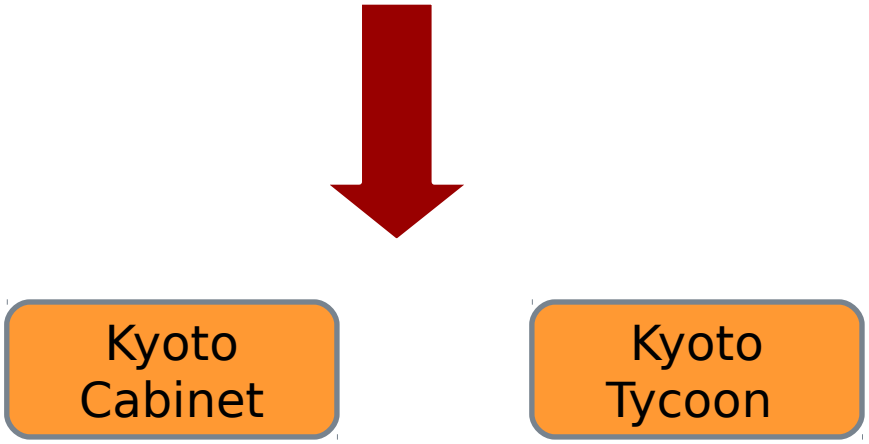
.
tch#bnum=10000000#xmsiz=802400000

.
tch#bnum=100000000#xmsiz=3802400000

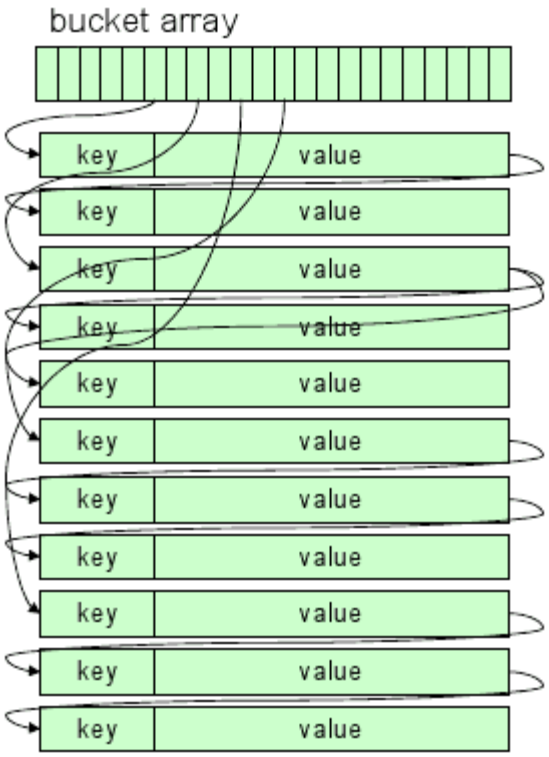


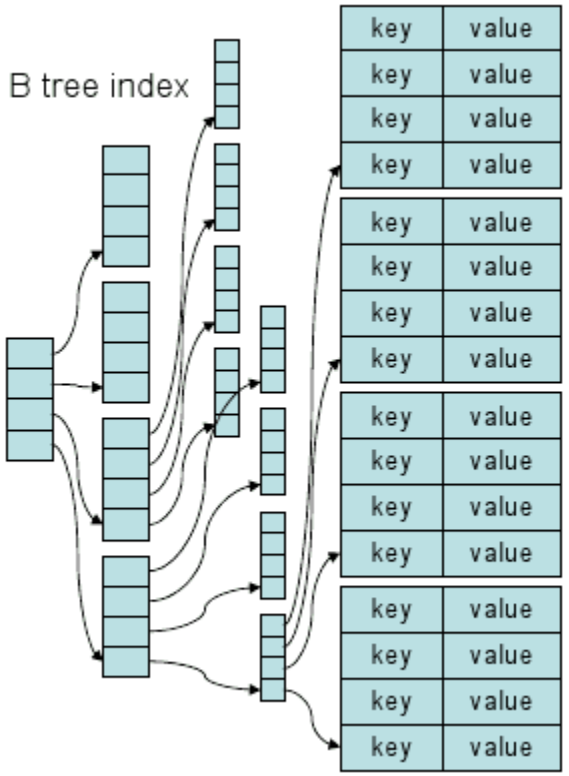
ttserver 的问题

- IO 不稳定，造成延迟
- 作者目前宣称不再升级

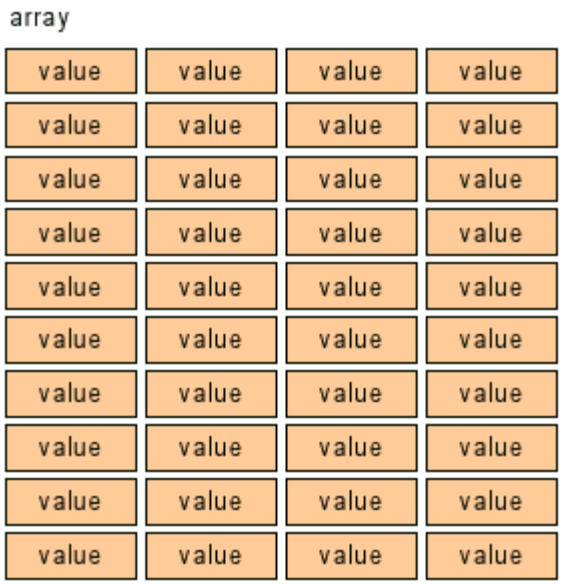


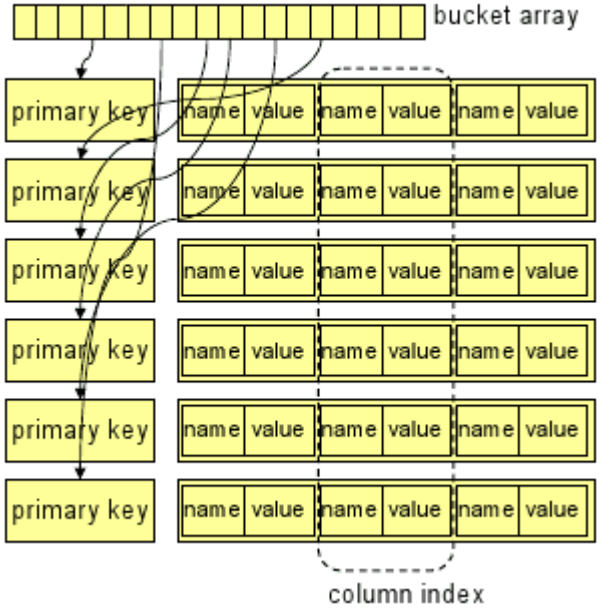
TC 的存储机制





TC 的存储机制


array
value

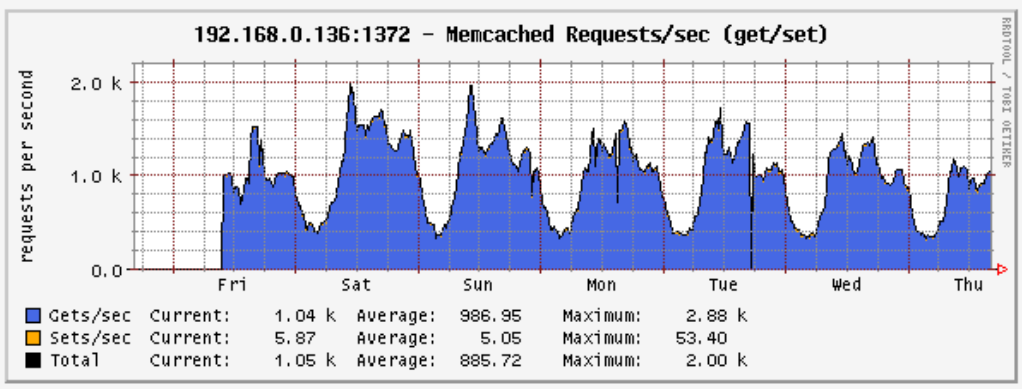

bucket array
primary key
name
value
column index

开发自己的 NOSQL

- ttserver 不稳定的风险
- 线上产品都是用了 memcached 协议（无缝切换）
- 增强对底层存储的技术控制能力
- 为此，我们开发了 INetDB

我们用在

- IMCP 系统的所有新闻存储，目前使用 lzf 压缩后超过 30GB
- 论坛的持久缓存启用了两个实例，使用了 php 客户端压缩，分别占用 42GB
- 论坛其中一个实例的监控图

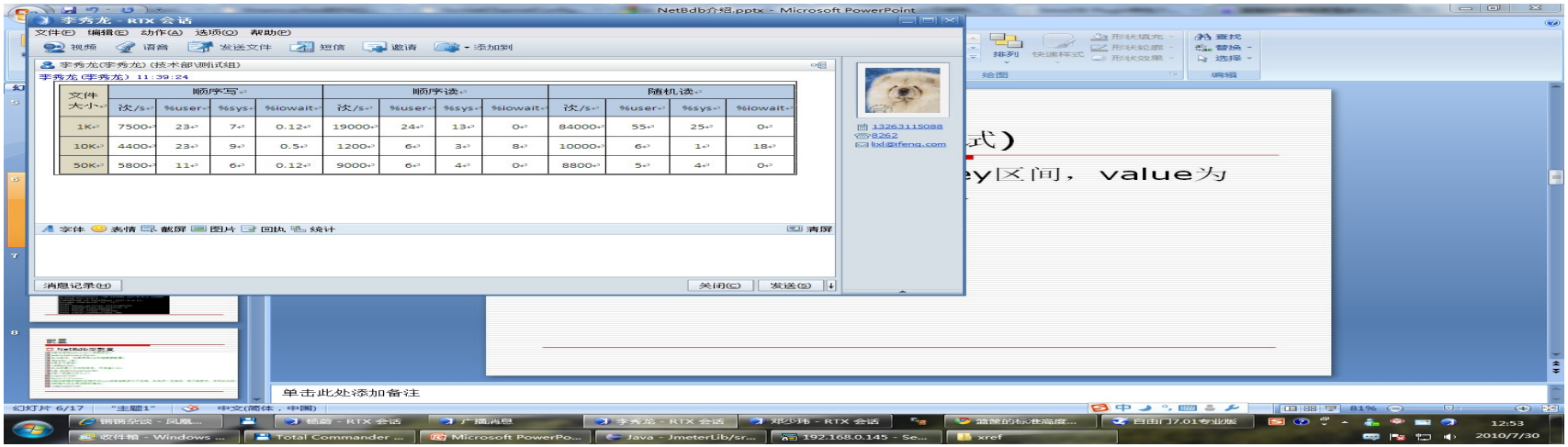


Features

- 兼容的 memcached 协议
- Master-slave 主从复制
- 支持 ttserver 复制协议
- 高性能
- 支持内部数据压缩（ gzip,lzf ）
- 数据遍历
- 复制无需保存类似 ttserver 的 bin-log
- 全面的监控数据接口

Benchmark(基准测试)

- 50 线程随机读取 10 万 key 区间， value 为 1K, 模拟应用的热点读取能达到 84000r/s

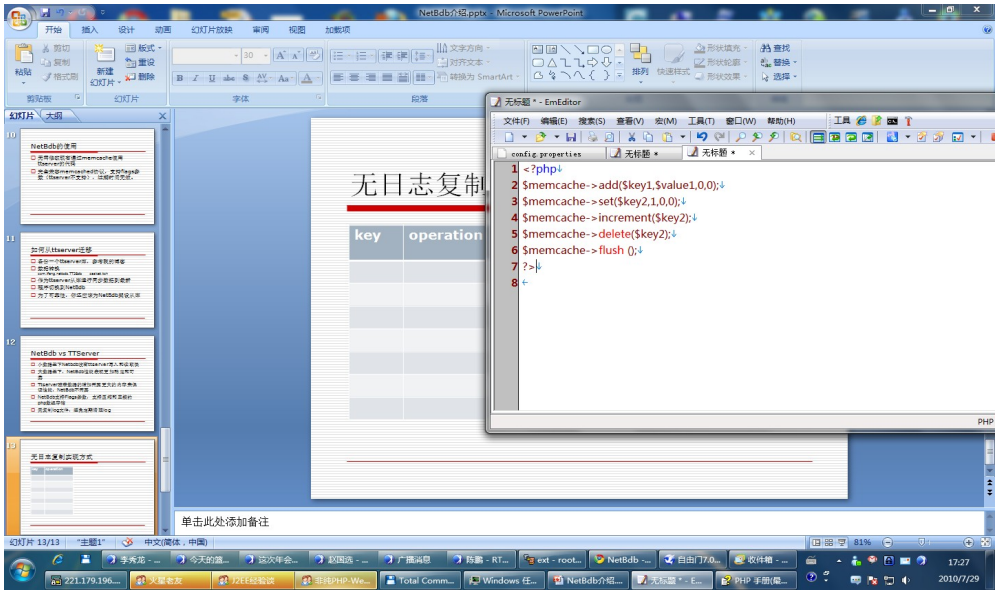


INetDB vs TTServer

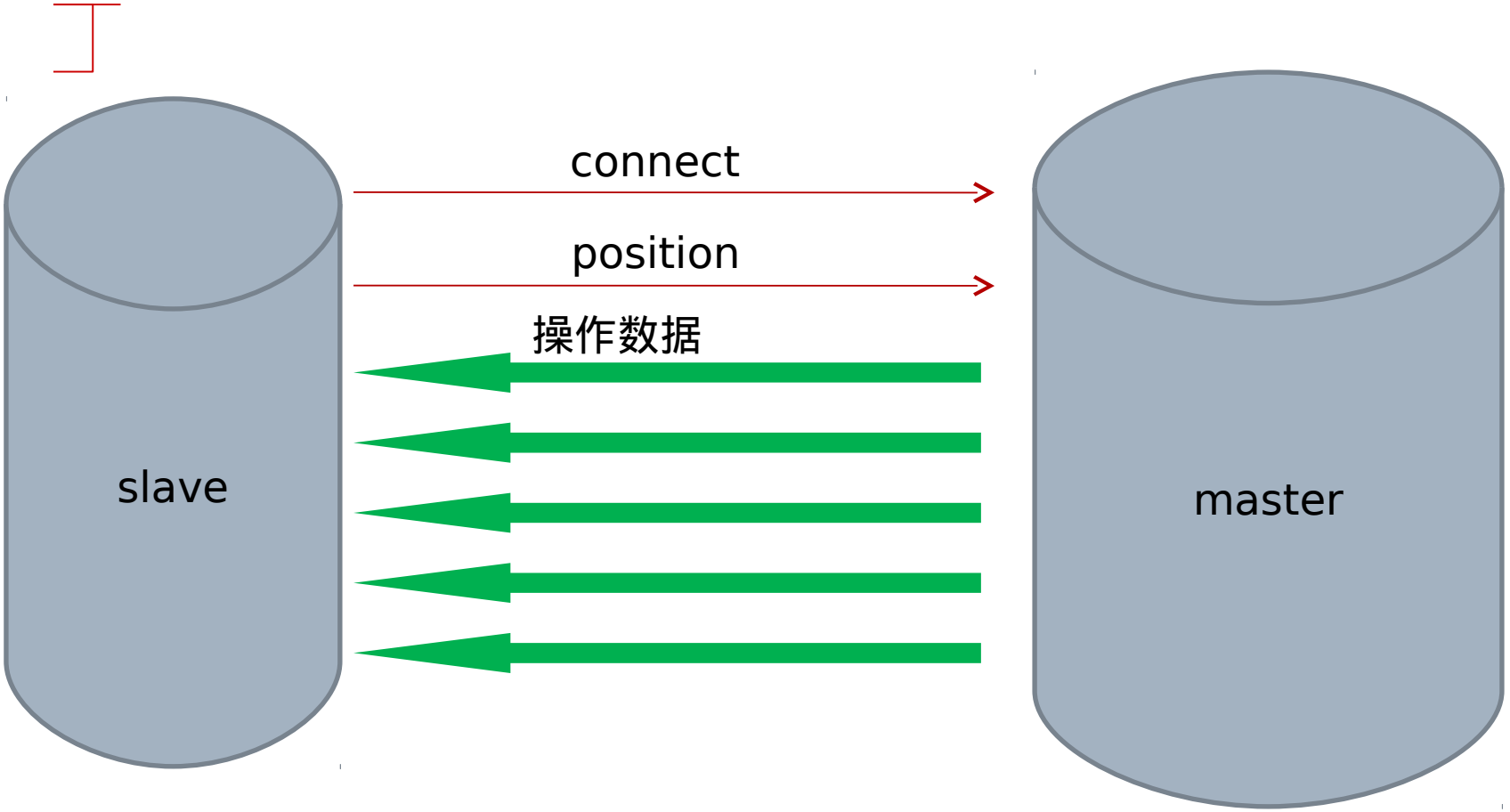
- 小数据量下 INetDB 没有 ttserver 写入和读取快
- 大数据量下， INetDB 性能表现更加稳定和可靠
- INetDB 支持 Flags 参数，支持压缩和直接的 php 数组存储 . 同时也支持过期参数。
- 无复制 log 文件，避免定期清理 log
- INetDB 的状态监控更多

无日志复制实现方式

pos	key	operation
1	$key1	add
2	$key2	set
3	$key2	increment
4	$key2	delete
5		flush



复制协议流程



提高性能的一些调整

- 你应该使用 php 的长连接
- 如果数据超过 5K 建议使用 php 的压缩选项
- 请求如果落在一定的 key 区间，也就是热点访问，可以提高缓存命中率
- ulimit -SHn 51200

延伸

- 利用复制协议可以做很多事情

关于磁盘 IO

- 随机写和随机读是很慢的
- 顺序写和顺序读是很快的
- 比较好避免随机写，难于避免随机读
- 充分利用内存仍然是最好的优化方式

关于开发自己的 NOSQL 存储

- 有现成的最好不要自行开发
- 弄清你需要的存储类型
- 结合你的业务特点，不要盲目对比性能
- 网络协议很重要
- 一切可替换，避免 Cassandra 在 digg 的遭遇

谢谢

Q&A