DTCC2013

MongoDB at Qihoo 360

王 超

- 背景
- 发展历程
  a) 初涉 - 千万
  b) 挑战 - 亿级
  c) 试炼 - 百亿
- 展望未来

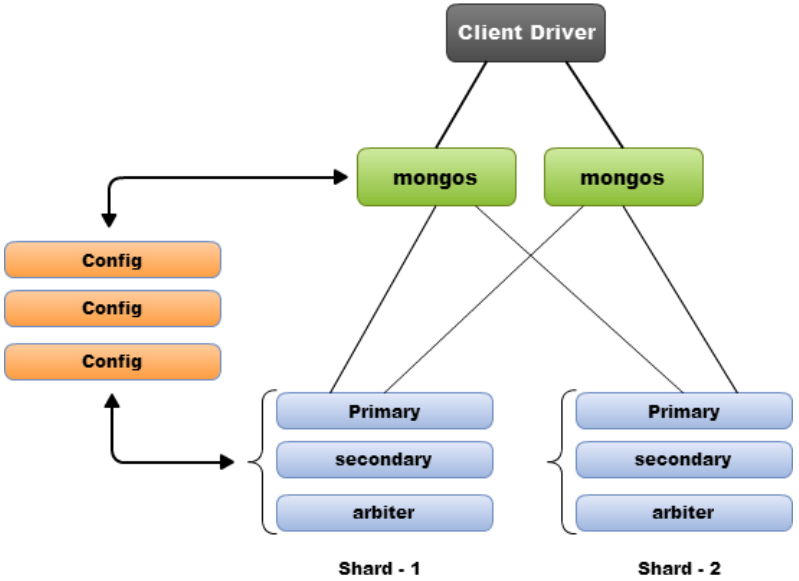
So Exciting!

- High Performance
- Scalability
- Schema free

2012.01

- MongoDB 2.0.2
- 2 * mongos + 3 * mongod + 3 * config
- mongod(1 primary + 1 secondary + 1 arbiter)

- 3 Servers(Xen)
- 32G RAM
- SAS 15K – RAID 5



千万级数据规模

- Keeping data in RAM

- QPS < 500
- R:W ~ 4:1
- Opcounters < 20 Million
- Document < 50 Million

问题

- 每天一万多个Timeout (3s)

排查

- Profiling Levels(1), slowms
- mongostat
- iostat

现象

- iostat，w/s, wkB/s 有规律的间隔出现
- 与slow log能对应上，批量呈现
- 产生I/O时，mongostat状态如下
  - insert/query/update 持续几秒内有所下降
  - lock > 80%
  - flushes = 1

--syncdelay

mmap, flush memory data into disk

- 默认60秒
- 不建议太长，早晚要刷入磁盘，出来混迟早要还的！
- RWLOCK, global Lock(ver 2.0.x)

or

db.runCommand({ setParameter: 1, syncdelay: N})

- 缩短syncdelay为5秒
- 减少了60%的timeout

继续观察…

slow query总伴随着moveChunk出现

- 调整balancer启动时间，避免高峰期工作

```
db.settings.update
(
        { "_id" : "balancer" },
        { $set : { "activeWindow" : { start : "00:00", stop : "8:00" } } },
        true
)
```

- Mongos Connection Pool / VersionManager Bug，偶尔超时

超时问题总结

- Syncdelay
- moveChunk, activeWindow
- BUG - Connection Pool / VersionManager

亿级数据规模 - 2012.04

- 6 Servers, 64G RAM
- SAS 15K – RAID 5
- Opcounters > 50 Million
- Document > 100 Million

问题

- Timeout (3s) again!
  - 平均latency上涨 (毫秒->百毫秒)
  - 平均lock > 50%
  - 缺页非常严重

- 0:00-8:00已无法均衡白天产生的数据

归根结底

- 数据超出了内存
- 纯随机读写

如何让数据重返内存？

- 节省空间使用
- 增加内存资源

业务应用场景

老的结构:

- _id: BSON string, hash(160 bit)
- cnum: Array
- ……

```
{
        _id: “d0be2dc421be4fcd0172e5afceea3970e2f3d940”,
        cnum: [0, 1, 2],
        ……
}
```

压缩后的结构:

- _id: BSON Binary, hash(160 bit)
  - ✓40 bytes -> 20 bytes
- cnum: Int32
  - ✓Array -> 位运算

……

空间节省一半

其他的好处…

TIPS:

注意document长度对QPS的影响

- 6000万数据
- 随机读写，数据小于内存

测试结果：

- 3K: r/s > 6000, w/s > 500
- 1K: r/s > 11000, w/s > 1500

预热数据

何时预热？

- 机器重启
- 增加secondary
- 增加shard

预热工具

- dd / cat 不好使
- vmtouch：http://hoytech.com/vmtouch/
  - 内置touch command (version 2.2)

0:00-8:00已无法均衡白天产生的数据

原因：

- IOPS瓶颈
  - shardkey: sha1, 数据散列在磁盘

解决

- moveChunk 加入限速功能
- balancer开始时间恢复为 0:00-24:00

内存问题？

预估两个月后，数据会再度超出内存

SSD in MongoDB

- No Raid! HBA直连，性能发挥到最好！
- PageFault? Memory? 浮云！
- Low latency

diao丝->高富帅

count: 3150170254 avgObjSize: 79.6102010326

method	count	averageTime(s)	threshold1	threshold2
MongoCursorException	49	0.001141	0.001939(0.96)	0.001982(1)
MongoCursorTimeoutException	56	3.002706	3.003143(0.95)	3.003162(0.98)
client::find	1440	0.000079	0.000110(0.99)	0.000124(1)
client::findOne	43087080	0.002107	0.008929(0.99)	0.116581(1)
client::insert	15050924	0.001071	0.002092(0.99)	0.005267(1)
client::update	2525748	0.001355	0.004065(0.99)	0.092776(1)
curosr::__destruct	1440	0.001301	0.002060(0.99)	0.044803(1)

count: 20029660 avgObjSize: 71.0063166324

table	method	count	averageTime(s)	threshold1	threshold2
	MongoCursorException	11	0.005424	0.008620(0.82)	0.023921(1)
	MongoCursorTimeoutException	30000	0.099879	0.115683(0.99)	0.157383(1)
	client::findOne	509160736	0.000275	0.004059(0.99)	0.007258(1)
	client::update	329099	0.001246	0.004160(0.99)	0.006655(1)

百亿级数据规模

- 100+ Servers, 64G RAM, SSD * 5
- Cluster: 20+
- Opcounters: 2+ Billion
- Document: 30+ Billion

高枕无忧？NO!

- NUMA架构
- 连接的选择
- 跨IDC应用
- 如何在线迁移业务

NUMA架构

现象：

- 内存无规律换入换出，pgscand/s、pgscank/s 飙升(sar –B)
- 某核CPU使用率 100%
- mongostat Lock > 90%
- 持续阻塞时间十秒左右（64G内存）

原因：

- 使用默认内存访问策略时，单NUMA节点(特别是0节点)内存使用超出单节点内存大小时, 上述问题与linux的行为有关。
- 关闭swap问题依旧存在

解决：

numactl --interleave=all ./xxx

echo 0 > /proc/sys/vm/zone_reclaim_mode

参考：

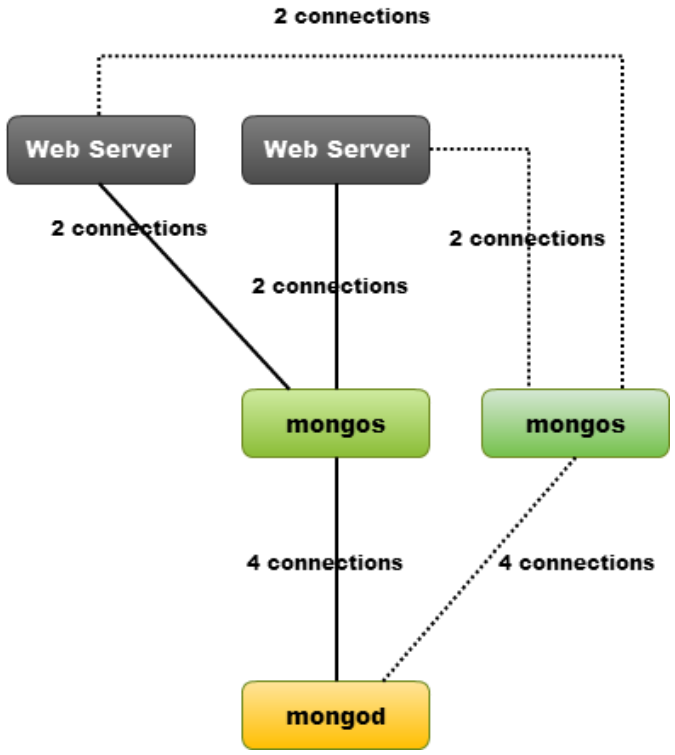
http://docs.mongodb.org/manual/administration/production-notes/#production-numa

http://blog.jcole.us/2010/09/28/mysql-swap-insanity-and-the-numa-architecture/

http://blog.jcole.us/2012/04/16/a-brief-update-on-numa-and-mysql/

连接的选择

- 使用长连接的一次事故
  - mongos/mongod crash
  - 启动即挂



pthread_create failed，达到系统最大上限

为什么？

- 一个连接一个线程的网络模型
- php driver < 1.2.10版本有连接泄露(超时异常时)，client设置timeout为100ms
- Client与每个mongos都建立连接，导致mongod连接X倍
- mongos/mongod 服务器复用

Mongod Conns(Threads) Nummber：

N Web Servers * N FastCGI Process * N mongos

e.g. 100 * 128 * 2 ~= 25K Conns(Threads) > maxConns

如何解决？

- Fix bug - php driver
- 调整系统参数

  ulimit [open files| max user processes]

  /proc/sys/kernel/threads-max

  /proc/sys/kernel/pid_max

  /proc/sys/vm/max_map_count
- 改代码去掉maxConns限制
- Client只连接一个mongos (Zookeeper解决可靠性问题)
- 做好连接的预先规划

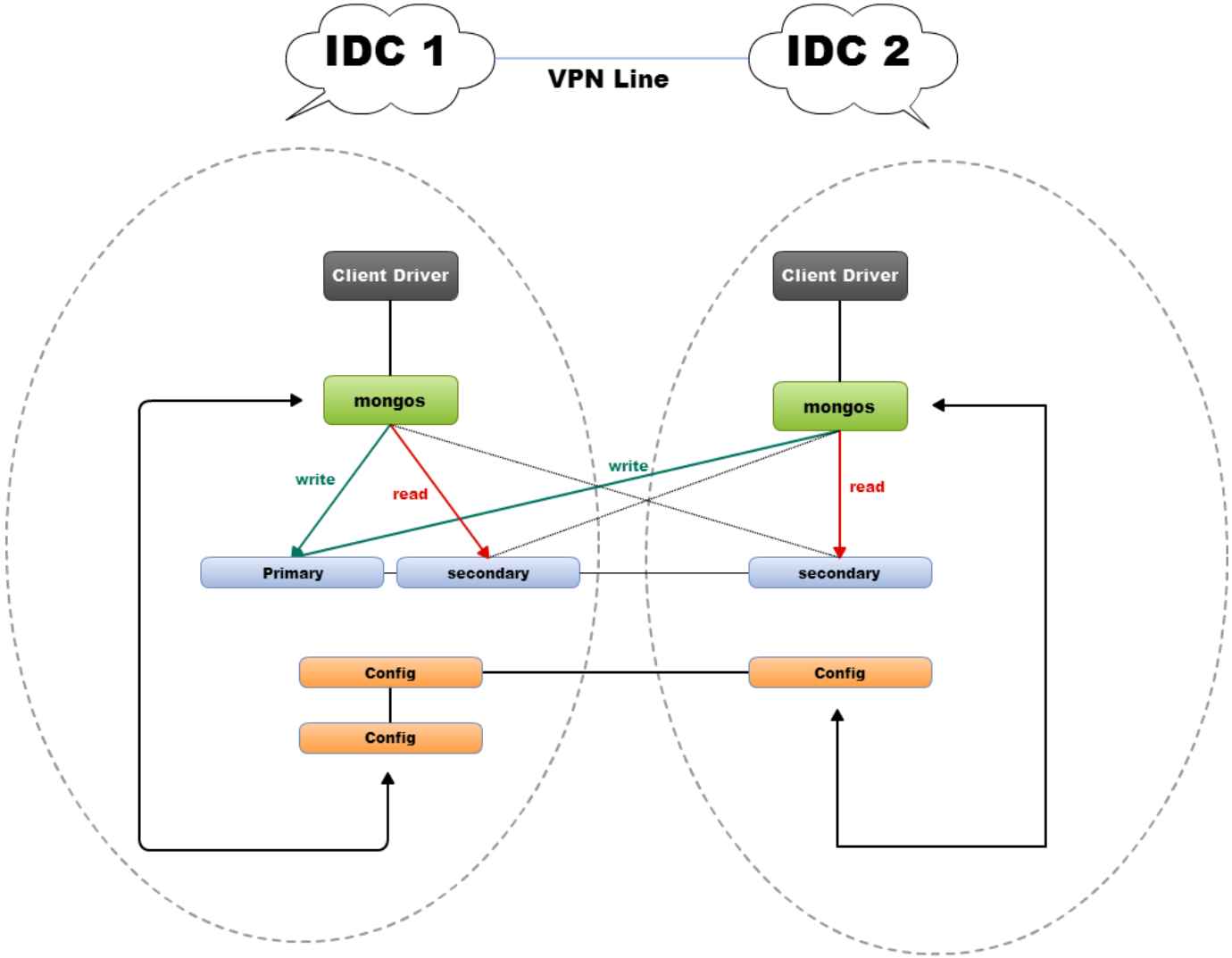
短连接

- 创建关闭连接的开销
- 创建关闭线程的开销(no threads cache)
- Mongos Connection Pool / VersionManager Bug，触发超时逻辑

跨IDC应用-单集群

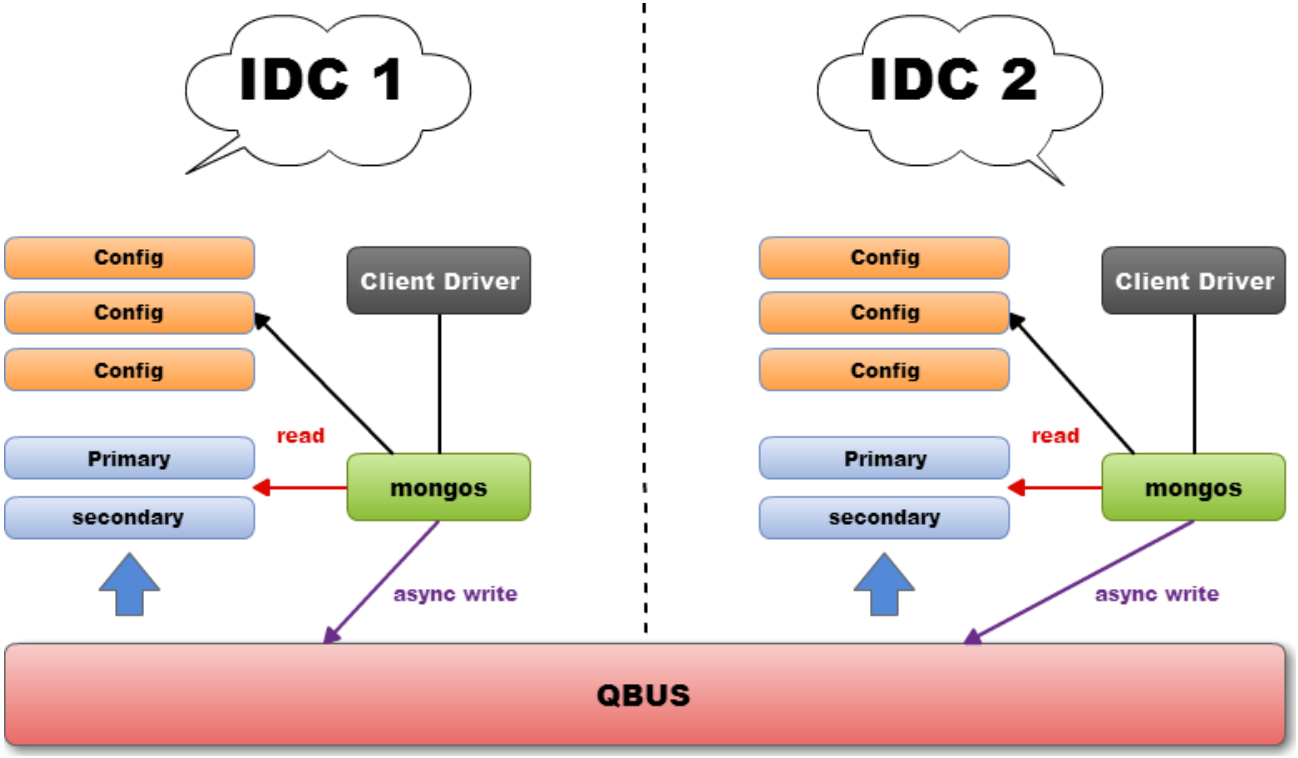
特点：

- 多机房容灾，架构、部署简单
- IDC之间依赖光纤
- 区分主次机房
- 适用于读多写少
  - 基于就近选择策略（2.2版本官方自带）


IDC 1
VPN Line
IDC 2
Client Driver
Client Driver
mongos
mongos
write
read
write
read
Primary
secondary
secondary
Config
Config
Config

跨IDC应用-多集群

- 数据同步 – QBUS（分布式消息队列）



相比单集群优点：

- 集群独立，调整灵活
- 光纤断不影响写入，仅影响新数据同步的实时性
- IDC瘫痪无需干预，业务切域名

  （单集群模式时，主IDC瘫痪需要手动切primary）

如何在线迁移业务？

- oplog实时同步程序(2.2版本自带)
  - 从Secondary copy数据

(mongodump太慢，同步完oplog就跟不到了)

展望未来

- WEB化集群管理
- 数据压缩
- 多线程数据同步、迁移
  - 新增secondary
  - 新增shard

期待:

collection lock! or document lock?

Q & A

Thanks

We Are Hiring...

Weibo: http://weibo.com/chancey
Email: chanceycn@gmail.com

Qihoo 360

360 www.360.cn