DTCC

2016中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2016

数 据 定 义 未 来

SequeMedia 盛拓传媒 ITPUB.com ChinaUnix ITPUB

MongoDB在58同城的应用实践@孙玄@58

关于我

- 58同城高级系统架构师
- 公司技术委员架构组主任
- 即时通讯、转转、C2C技术负责人
- 前百度高级工程师
- 代表58同城对外交流
  - QCon
  - SDCC
  - DTCC
  - Top100
  - 程序员
  - UPYUN
  - TINGYUN
  - ……

2016中国数据库技术大会
DATABASE TECHNOLOGY CONFERENCE CHINA 2016

代表58对外交流

- Qcon(全球软件开发大会)
- SDCC(中国开发者大会)
- Top100(全球案例研究峰会)
- DTCC(中国数据库技术大会)
- 《程序员》撰稿2次
- 58技术发展这10年[计划中]

OutLine

- MongoDB在58同城的使用情况
- 为什么要使用MongoDB
- MongoDB在58同城的架构设计与实践
- 针对业务场景我们在MongoDB中如何设计库和表
- 数据量增大和业务并发，我们遇到典型问题及其解决方案
- MongoDB如何监控

MongoDB在58同城的使用情况

- 使用业务线
  - 58帮帮
  - 58交友
  - 58招聘
  - 58信息质量
  - 58测试
  - 赶集
  - 58英才
  - ……

为什么要使用MongoDB

- Why?
  - Scalable & High Availability
    - Master-Slave
    - Replic Set
  - High-performace
    - MMAP
      - Persistent
      - Cache
      - 数据一致性
  - Rich Querying
  - Full Index Support
  - Auto-Sharding
- What?
  - 事务性要求低
  - 高访问量
  - 垂直业务扩展

为什么要使用MongoDB

- How?
  - free schema
    - schema(简化应用使用)
    - 真的FREE吗?
      - 重复的Schema
      - ALL Schema
    - 如何应对?
      - 减少字段名
      - 数据存储压缩
  - Auto-sharding
    - 库级sharding
    - Collection sharding
      - 手动sharding(切分大文档)
  - 内嵌文档
    - 自然、高效
    - 更多重复数据、反规范化
    - RDBMS表拆分
    - 类RDBMS处理
    - 字段名尽可能短小，上层做映射
  - 自动生成_id
    - 默认有(12个字节)
      - 存储空间大
    - 业务层客户端生成
      - 减少MongoDB服务端开销
  - …

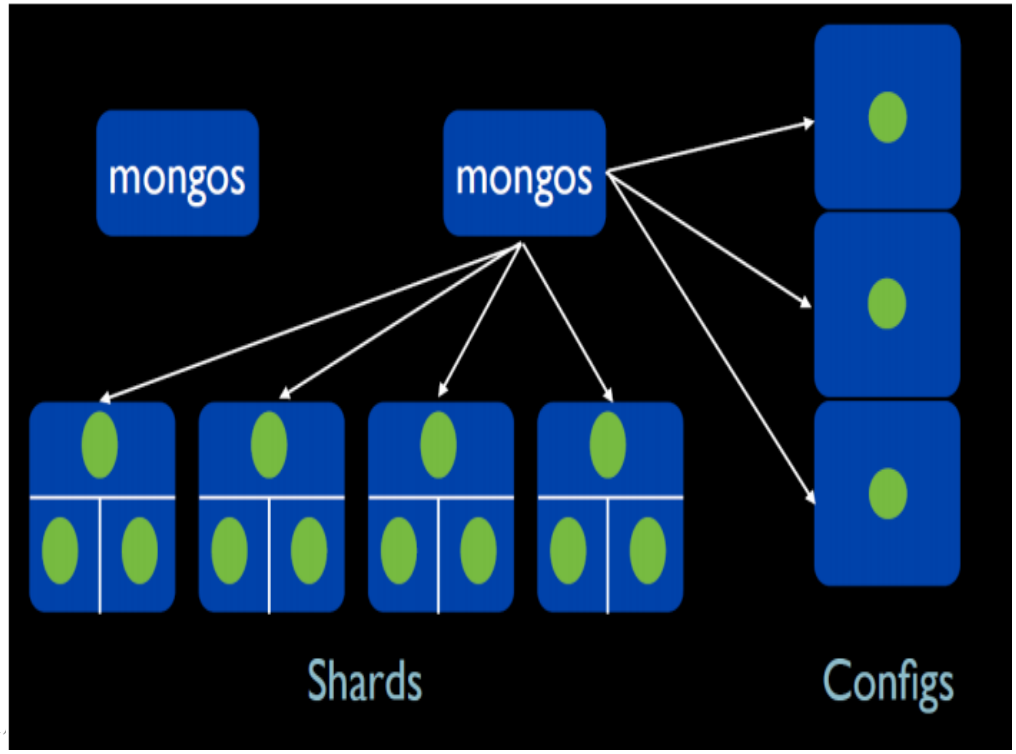
MongoDB在58同城的架构设计与实践

MongoDB部署

- Replica Set + Sharding
  - Shard Server (Replica Set)
  - Config Server（n）
  - Route Server（n）
  - Arbiter Server （n）

  - 增加Shard Server
  - RS内部增减
  - 读写分离
  - 故障转移

  - 库级sharding(move primary)
  - 表级手动sharding
  - auto-sharding(指定时间段凌



针对业务场景我们在MongoDB中如何设计库和表

- Auto-Sharding is not that Reliable
  - Why?
    - Sharding key
      - 单一key分片不均衡
      - 复合key性能消耗
    - Count值计算不准确
      - Chunk移动过程，计算可能偏大
    - Balancer的稳定性&智能性
      - Sharding发生的时间不确定性
      - 指定Sharding迁移时间

针对业务场景我们在MongoDB中如何设计库和表

- 我们如何设计库？
  - 线上环境禁用Auto-Sharding
  - 开启数据级别分片
    - db.runCommand({“enablesharding”: “im”});
  - 特定库指定到某一固定分片上
    - db.runCommand({movePrimary:“im”, to: “sharding1”});
  - 保证数据无迁移稳定
  - 避免Auto-Sharding带来的问题
  - 完全可控
  - 效果好

针对业务场景我们在MongoDB中如何设计库和表

- 我们如何设计库？
  - 库设计
    - 对业务增长情况要要求
    - 目前是什么量级
    - 半年~一年是什么量级
  - 库分片
    - Sharding分片数量
  - 容量规划的关键原则
    - Memory > Index + Hot Data
    - 索引+热数据要全部加载到内存(MMAP)
      - MongoDB的高性能

针对业务场景我们在MongoDB中如何设计库和表

- 我们如何设计库？
  - 如何设计频繁更新删除记录的collection
    - MongoDB的数据库是按文件来存储的
      - 例如：db1下的所有collection都放在一组文件内db1.0,db1.1,db1.2,db1.3…
    - 将频繁更新删除的表放在一个独立的数据库下，将会减少碎片，并提高性能
  - 单库单表绝对不是最好的选择
    - 原因有三：
      - 表越多，映射文件越多，从MongoDB的内存管理方式来看，浪费越多
      - 同理，表越多，回写和读取的时候，无法合并IO资源，大量的随机IO对传统硬盘是致命的
      - 单表数据量大，索引占用高，更新和读取速度慢
  - Local库
    - Local库主要存放oplog，oplog同样是要消耗内存的
    - 选择一个合适的oplog值，很重要
    - 高插入高更新，并带有延时从库的副本集需要一个较大的oplog值（比如20G）
    - 如果没有延时从库，则可以适当放小oplog值

针对业务场景我们在MongoDB中如何设计库和表

- 我们如何设计表？
  - RDBMS与MongoDB
    - 数据库、表/集合、行/文档
    - 三范式与嵌套
  - 举例“人”描述
    - RDBMS(People 、Address)
    - MongoDB(People)

People:

姓名	性别	年龄	住址
M	男	25	ID
.....	.....	.....	.....

Address:

住址	国家	城市	街道
ID	中国	北京	朝阳区北苑路 108 号
.....	.....	.....	.....

```
{
  “姓名” ： “M”,
  “性别” ： “男”,
  “年龄” ： “25”,
  “住址” ：{
            “国家” ： “中国”,
            “城市” ： “北京”,
            “街道” ： “朝阳区北苑路 108 号”
          }
}
```

针对业务场景我们在MongoDB中如何设计库和表

- 我们如何设计表？
  - 一对一
    - IM用户信息表
      - 用户uid、用户登录名、用户昵称、用户签名
      - 类RDBMS设计
      - 一张用户信息表
      - {uid:XX, loginname:XX, nickname:XX, sign:XX}
      - uid实际上为_id 主键

针对业务场景我们在MongoDB中如何设计库和表

- 我们如何设计表？
  - 一对多
    - 用户在线消息表
      - 一个人可以收很多条消息
      - 典型的一对多
    - 如何设计
      - 类RDBMS
      - {uid，msg，msg_content}
      - 123，1，你好
      - 123，2，在吗
    - MongoDB嵌套
      - {uid:XX，msg:{[{msgid:1,msg_content:你好}，{msgid:2，msg_content:在吗}]}}
    - 嵌套更自然、更新不方便、单条16MB的限制
    - RDBMS遵循范式、查询条件更灵活、扩展性高

针对业务场景我们在MongoDB中如何设计库和表

- 我们如何设计表？
  - 多对多
    - Team表&User表
      - 一个Team中有多个User
      - 一个User也可能属于多个Team
      - 典型的多对多关系
    - 如何设计
      - 类RDBMS(三张表)
        - » Team表
          - {teamid, teamname, ……}
        - » User表
          - {userid,username,……}
        - » Relation表
          - {refid, userid, teamid}

针对业务场景我们在MongoDB中如何设计库和表

- 我们如何设计表？
  - 多对多
    - Team表&User表
      - 一个Team中有多个User
      - 一个User也可能属于多个Team
      - 典型的多对多关系
    - 如何设计
      - MongoDB嵌套（2个集合）
      - Team表
        - {teamid,teamname, teammates:{[userid, userid, ……]}
      - User表
        - {useid,usename, teams:{[teamid, teamid,……]}}
        - {useid,usename}
      - 每次通过Team表中的teammates反查询得到teamid
      - Teammates需要建立索引

针对业务场景我们在MongoDB中如何设计库和表

- 我们如何设计表？
  - 单表数据量大如何Sharding？
    - Collection Sharding
      - 手动Sharding
    - 如何手动Sharding
      - 单表千万量级
      - 单ID查询key水平拆分表
    - 用户信息表
      - {uid, loginname, sign, ……}
      - 按照uid水平拆分
        - » uid%64
      - 按照uid的查询
    - 可控、可靠
    - 避免了Auto-Sharding带来的不稳定因素

针对业务场景我们在MongoDB中如何设计库和表

- 我们如何设计表？
  - 单表数据量大如何Sharding？
    - 混合ID查询
      - 商品表
        - {uid, infoid, info,……}
          - infoid包含uid的信息（infoid最后8个bit，uid的最后8个bit）
          - infoid水平拆分
            - infoid%64
          - 按照uid的查询
          - 按照infoid的查询

数据量增大和业务并发，遇到的问题及其解决方案

大量删除数据问题及其解决方案

- 背景
  - IM离线消息集合结构
    - 字段：
      - msgid, fromuid, touid, msgcontent, timestamp, flag
      - 其中touid为索引
      - 其中flag表示离线消息是否已读取，0未读，1读取
    - 删除已经读取的离线消息
      - db.collection.remove({“flag” : 1});
      - 命令非常简单
      - 看似很容易就搞定了
      - 5kw的数据条数
      - 200G的存储空间

数据量增大和业务并发，遇到的问题及其解决方案

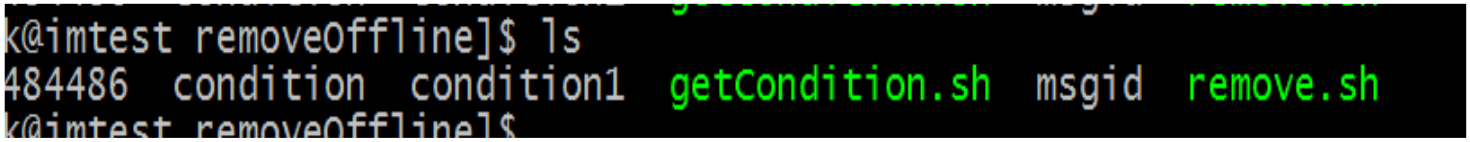
大量删除数据问题及其解决方案

- 遇到问题
  - 晚上10点后部署删除
  - 早上7点还没删除完毕
  - 断续有报警
  - 从库延迟大
  - QPS/TPS很低
  - 业务无法响应
- 分析原因
  - db.collection.remove({“flag” : 1});
  - flag不是索引字段
  - 全部扫描
  - 删除速度很慢
  - 冷数据和热数据swap，造成内存中全是冷数据，服务能力急剧下降

数据量增大和业务并发，遇到的问题及其解决方案

大量删除数据问题及其解决方案

- 如何解决
  - 紧急方案
    - 找到正在执行的op
    - kill opid
  - 长期删除
    - 业务层优化
      - 逻辑删除--》物理删除
    - 离线删除优化
      - 每晚定时从库导出要删除的数据
      - 通过脚本按照objectid的方式进行删除
      - 删除速度可以控制
      - 避免对线上服务影响



MongoDB遇到问题及其解决方案

大量数据空洞问题及其解决方案

– 遇到问题

- 大量删除数据，MongoDB存在大量的数据空洞
- 这些空洞数据也同时会加载到内存
- 导致内存有效负荷低
- 数据不断在swap
- MongoDB数据库性能并没有明显提升
- 怎么办？

MongoDB遇到问题及其解决方案

- 大量数据空洞问题及其解决方案
  - 解决方案
    - MongoDB数据空间的分配是以DB为单位
    - 不是以Collection为单位的
    - 问题的关键是大量碎片无法利用
    - 碎片整理、空洞合并收缩
  - 怎么做？
    - 方案一
      - Online Compress
        - » Compact命令
          - db.yourCollection.runCommand(“compact”);
          - Collection级别压缩
          - 去除Collectoin所在文件碎片
          - 影响服务
          - 压缩效果差，不推荐使用

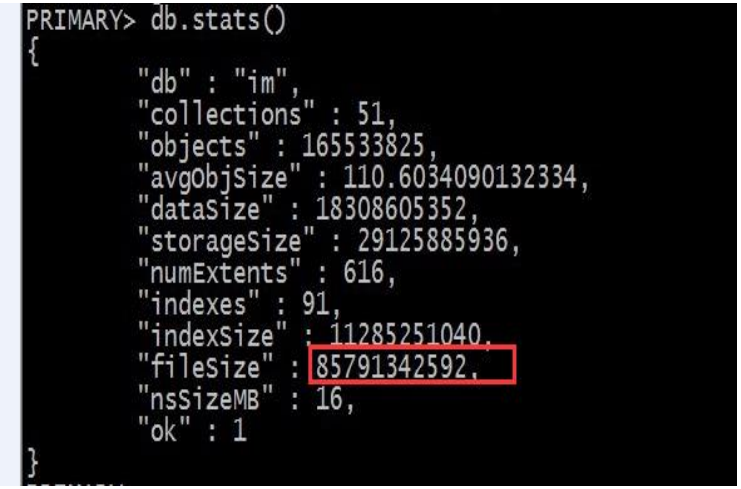
MongoDB遇到问题及其解决方案

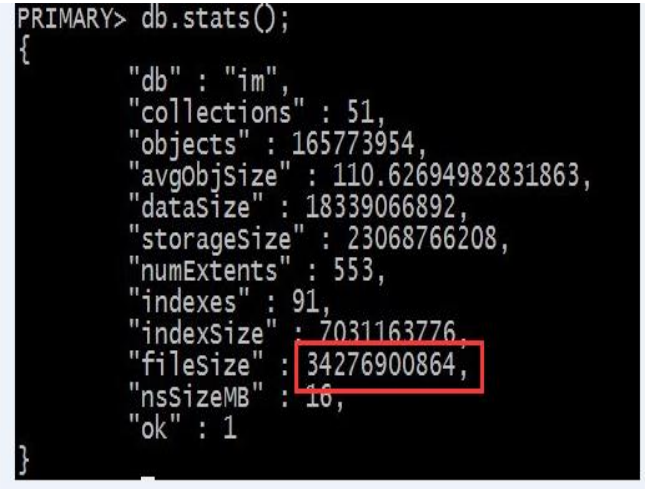
大量数据空洞问题及其解决方案

- 方案二
  - 收缩数据库
    - 把已有的空洞数据，remove掉，重新生成一份无空洞数据
    - 先预热从库
    - 把预热的从库提升为主库
    - 把之前主库的数据全部删除
    - 重新同步
    - 同步完成后，预热此库
    - 把此库提升为主库
    - 完全无碎片
    - 收缩率100%
    - 持续时间长、投入维护成本高
    - 收缩过程单点存在一定风险
      - » Replic Set

MongoDB遇到问题及其解决方案

- 大量数据空洞问题及其解决方案
  - 收缩数据库
    - 效果对比如下：
      - 收缩前85G存储文件，收缩后34G存储文件，节省了51G存储空间，大大提升了性能





如何监控MongoDB集群？

- MongoDB集群监控
  - mongosniff
  - mongostat
  - mongotop
  - db.xxoostatus
  - web控制台监控
  - MMS
  - 第三方监控
  - ……

如何监控MongoDB集群

- mongostat能监控什么，如何监控
  - mongostat
    - ./mongostat --host 127.0.0.1 --port 33333

```
[im_work@imtest bin]$ ./mongostat  --host 127.0.0.1 --port 33333
connected to: 127.0.0.1:33333
insert  query update delete getmore command flushes mapped  vsize    res faults locked % idx miss %     qr|qw   ar|aw  netIn netOut  conn       time
    0      0      0      0       0       1       0  62.9g   126g   277m      0        0          0       0|0     0|0    62b     1k    25   15:10:44
    0      0      0      0       0       1       0  62.9g   126g   277m      0        0          0       0|0     0|0    62b     1k    25   15:10:45
    0      0      0      0       0       1       0  62.9g   126g   277m      0        0          0       0|0     0|0    62b     1k    25   15:10:46
    0      0      0      0       0       1       0  62.9g   126g   277m      0        0          0       0|0     0|0    62b     1k    25   15:10:47
```

    - 字段说明
      - insert
        - » 每秒插入量
      - query
        - » 每秒查询量
      - update
        - » 每秒更新量
      - delete
        - » 每秒删除量

如何监控MongoDB集群

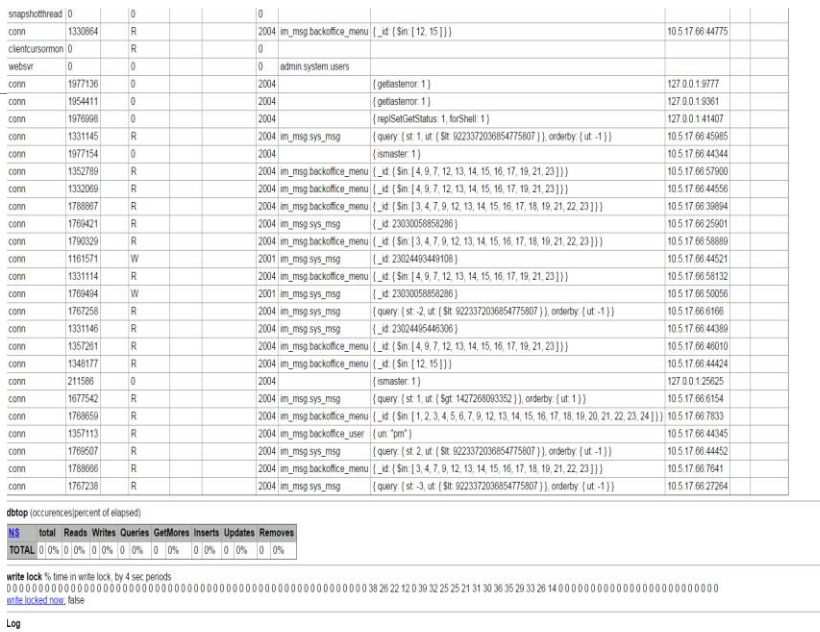
- mongostat能监控什么，如何监控
  - mongostat
    - 字段说明
      - locked
        - 加锁时间占操作时间百分比
      - faults
        - 缺页中断数量
      - miss
        - 索引miss的数量
      - qr|qw
        - 客户端查询排队长度（读|写）
      - conn
        - 当前连接数
      - ar|aw
        - 活跃客户端数量(读|写)
      - time
        - 当前时间

如何监控MongoDB集群

- mongostat
  - 经验实战
    - Locked、faults、miss、qr|qw
    - 值越小越好
    - 最好都为0
    - lokced最好不要超过10%
    - faults、miss原因
      - 内存不够
      - 内冷数据
      - 索引设置不合理
    - qr|qw堆积
      - 数据库处理慢

如何监控MongoDB集群

- Web控制台监控
  - Web控制台监控
    - 自带的，和MongoDB服务一同开启
    - MongoDB监听端口+1000
    - 如果MongoDB的监听端口33333，则Web控制台端口为34333
    - http://ip:port
    - http://8.8.8.8:34333
    - 可以监控什么
      - 当前MongoDB所有的连接数
      - 各个数据库和Collection的访问统计
      - 包括：Reads，Writes，Queries等
      - 写锁的状态
      - 最新的几百行日志文件



snapshotthread	0	0			0			
conn	1330864	R			2004	im_msg.backoffice_menu	{ _id: { $in: [12, 15] } }	10.5.17.66:44775
clientcursormon	0	R			0			
websvr	0	0			0	admin.system.users		
conn	1977136	0			2004		{ getlasterror: 1 }	127.0.0.1:9777
conn	1954411	0			2004		{ getlasterror: 1 }	127.0.0.1:9361
conn	1976998	0			2004		{ replSetGetStatus: 1, forShell: 1 }	127.0.0.1:41407
conn	1331145	R			2004	im_msg.sys_msg	{ query: { st: 1, ut: { $lt: 9223372036854775807 } }, orderby: { ut: -1 } }	10.5.17.66:45985
conn	1977154	0			2004		{ ismaster: 1 }	10.5.17.66:44344
conn	1352789	R			2004	im_msg.backoffice_menu	{ _id: { $in: [4, 9, 7, 12, 13, 14, 15, 16, 17, 19, 21, 23] } }	10.5.17.66:57900
conn	1332069	R			2004	im_msg.backoffice_menu	{ _id: { $in: [4, 9, 7, 12, 13, 14, 15, 16, 17, 19, 21, 23] } }	10.5.17.66:44556
conn	1788867	R			2004	im_msg.backoffice_menu	{ _id: { $in: [3, 4, 7, 9, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23] } }	10.5.17.66:39894
conn	1769421	R			2004	im_msg.sys_msg	{ _id: 23030058858286 }	10.5.17.66:25901
conn	1790329	R			2004	im_msg.backoffice_menu	{ _id: { $in: [3, 4, 7, 9, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23] } }	10.5.17.66:58889
conn	1161571	W			2001	im_msg.sys_msg	{ _id: 23024493449108 }	10.5.17.66:44521
conn	1331114	R			2004	im_msg.backoffice_menu	{ _id: { $in: [4, 9, 7, 12, 13, 14, 15, 16, 17, 19, 21, 23] } }	10.5.17.66:58132
conn	1769494	W			2001	im_msg.sys_msg	{ _id: 23030058858286 }	10.5.17.66:50056
conn	1767258	R			2004	im_msg.sys_msg	{ query: { st: -2, ut: { $lt: 9223372036854775807 } }, orderby: { ut: -1 } }	10.5.17.66:6166
conn	1331146	R			2004	im_msg.sys_msg	{ _id: 23024495446306 }	10.5.17.66:44389
conn	1357261	R			2004	im_msg.backoffice_menu	{ _id: { $in: [4, 9, 7, 12, 13, 14, 15, 16, 17, 19, 21, 23] } }	10.5.17.66:46010
conn	1348177	R			2004	im_msg.backoffice_menu	{ _id: { $in: [12, 15] } }	10.5.17.66:44424
conn	211586	0			2004		{ ismaster: 1 }	127.0.0.1:25625
conn	1677542	R			2004	im_msg.sys_msg	{ query: { st: 1, ut: { $gt: 1427268093352 } }, orderby: { ut: 1 } }	10.5.17.66:6154
conn	1768659	R			2004	im_msg.backoffice_menu	{ _id: { $in: [1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24] } }	10.5.17.66:7833
conn	1357113	R			2004	im_msg.backoffice_user	{ un: "pm" }	10.5.17.66:44345
conn	1769507	R			2004	im_msg.sys_msg	{ query: { st: 2, ut: { $lt: 9223372036854775807 } }, orderby: { ut: -1 } }	10.5.17.66:44452
conn	1788666	R			2004	im_msg.backoffice_menu	{ _id: { $in: [3, 4, 7, 9, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23] } }	10.5.17.66:7641
conn	1767238	R			2004	im_msg.sys_msg	{ query: { st: -3, ut: { $lt: 9223372036854775807 } }, orderby: { ut: -1 } }	10.5.17.66:27264

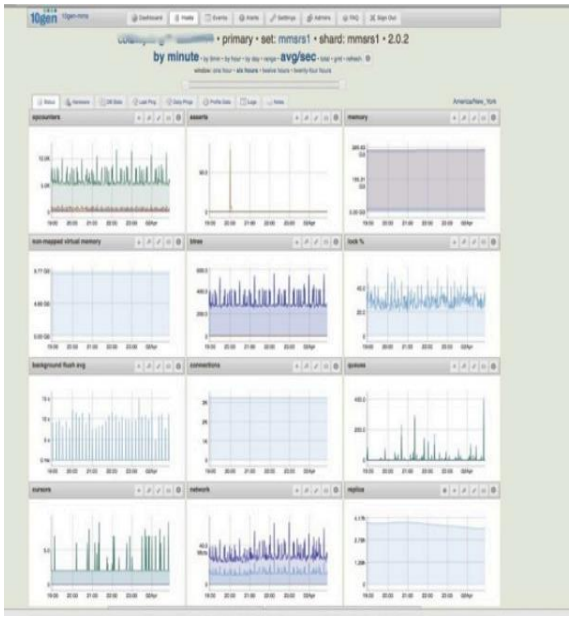
dbtop (occurences|percent of elapsed)

NS	total		Reads		Writes		Queries		GetMores		Inserts		Updates		Removes	
TOTAL	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%

write lock % time in write lock, by 4 sec periods
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 26 22 12 0 39 32 25 25 21 31 30 36 35 29 33 26 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
write locked now: false

Log

如何监控MongoDB集群



- MMS监控
  - MMS
    - 2011年发布
    - 官方的云监控服务
    - 可视化图形监控
    - 工作原理
      - 在MMS服务器上配置需要监控的MongoDB信息（ip/port/user/passwd等）
      - 在一台能够访问你MongoDB服务的内网机器上运行其提供的Agent脚本
      - Agent脚本从MMS服务器获取到你配置的MongoDB信息
      - Agent脚本连接到相应的MongoDB获取必要的监控数据
      - Agent脚本将监控数据上传到MMS的服务器
      - 登录MMS网站查看整理过后的监控数据图表
    - 安装、部署简单参考
      - http://mms.10gen.com

如何监控MongoDB集群

- 第三方监控
  - MongoDB开源爱好者和团队支持者较多
  - 在常用监控框架上扩展
    - zabbix ZABBIX
      - CPU负荷、内存使用、磁盘使用、网络状况、端口监视、日志监视等
    - nagios Nagios
      - 监控网络服务（HTTP等）、监控主机资源（处理器负荷、磁盘利用率等）、插件扩展、报警发送给联系人（EMail、短信、用户定义方式）、手机查看方式
    - ……

总结

- MongoDB在58同城的使用情况
- 为什么要使用MongoDB
- MongoDB在58同城的架构设计与实践
- 针对业务场景我们在MongoDB中如何设计库和表
- 数据量增大和业务并发，我们遇到典型问题及其解决方案
- MongoDB如何监控

THANKS

SequeMedia 盛拓传媒 ITT168.com ChinaUnix ITPUB

THANKS