AWS EC2 & MongoDB

点滴心得交流

2012.2 by Night Sailer（潘凡）

摘要

* EC2选型要点
* 数据库设计和部署
* 数据备份
* 一些技巧和注意事项

EC2选型要点

MongoDB关注内存而不是CPU，计算负载不高

理想是所有数据的都能放在内存中

退而其次确保内存能把常用数据集都放下（索引 + 热数据）

EC2 Instance Type

适合MongoDB的Instance Type

* m1.large (经济适用型）

    7.5GB, 4 CU (2 cores)

* m1.xlarge / m2.xlarge (标准型）

    15GB/17.1GB, 8 CU/6.5CU, 4/2 cores

* m2.2xlarge / m2.4xlarge (高富帅起步）

    34.2GB/68.4GB, 13/26 CU,4／8 cores

EC2 Instance Type

Region，必选Tokyo

* 价格略高，但国内访问良好

Reserved Instances

* 根据24x7需求购买Heavy Utilization

* 1 App + 2 mongodb instance

* Light Utilization, > 32%/y, > 2800h

* Medium Utilization, >48%/y,

On-Demand Instance(< 32%/y)

EC2 AMI

AMI选择（注意安全）

* S3 based AMI（个人优选）

    * 所有节点都是无状态,强制架构设计必须可以容灾和scale out

    * Instance storage,省钱

* Amazon Linux

    * RHEL系，个人偏好

    * 内置AWS工具 便于定制上手

MongoDB Storage

MongoDB 存储

* Instance storage
    * 可存放log
    * 临时目录（repairDatabase)
* 数据存储
    * EBS

EBS的问题

局限性

* NAS，IO能力有限
    * 单盘 IOPS <100 (远低于sata)
* LVM
* Soft RAID
    * mdadm

EBS RAID

两种可选方案

* RAID0 （我目前暂时选用）

    * 8 disks (chunk可调到256k）

* RAID 10（可靠性高）

    * 8x2 disks

EBS RAID

文件系统

* XFS，个人推荐

    * 可用空间比高,大文件性能佳,恢复速度快

        * xfs_growfs 动态扩展空间方便至极

* 微调:（raid0, 8 disks，256k chunks）

mkfs.xfs -d sunit=512,swidth=3072 /dev/vg1/mongo

/dev/vg1/db1 /mongo
sunit=512,swidth=3072,noatime,nodiratime,nobarrier

方案设计

设计时考虑

* 审慎考虑是否真的需要auto-sharding
* 从业务场景出发, 考虑 单库 vs 多库
* 常用数据能否可预知，方便区分

经典模式

单库 + auto-sharding

* 简单

* 做好不易，特别注意sharding key的优选

* 要克服的问题：shard 不均衡，数据波动

* mongos 可靠性已经加强,诡异案例减少

个人的方案

我目前使用的方式：

多库／多RS， no-sharding

* 不纠结mongoS/mongoC的可靠性问题
* 冷热数据分库，部署不同的ReplicaSet
  * 后台处理进程和前台进程所需的数据不同库
* RS粒度可控，延展性可控，规避EBS缺陷

个人的方案

* 后台处理进程和前台进程所需的数据不同库

* 常用集中的数据由后台数据生产出来

  * 控制在～10m 数据量

  * 使用Gearman队列触发数据的后台更新

    * 用户登录 => 通知 => 刷新数据

    * 用户注销 => 活跃用户 => 后台定时处理

个人的方案

缺陷:

* 针对我们的情况非通用，局限性很大

* 由于EBS最佳配置的容量有限，扩容时要确保业务数据能够顺利迁移

  * 确认单表容量 < 2T

* App层的逻辑稍微复杂点，不简单使用ORM

部署

充分压榨EC2的资源，提高性价比

* 每个Instance可部署多个mongodb instance

* 不同的mongodb 绑定到不同的core

    * 如部署数量最大可EC2 cores – 1

* Primary/Secondary 尽量交叉部署

* Arbitor 等混合部署到app ec2上

* RS data nodes >= 3(规避Raid0风险)

数据库备份

1. 常规备份

* 使用EBS的snaptshot 机制

  * 在Slave端进行

  * 辅助工具：ec2-consistent-snapshot

* 每周执行一次

数据库备份

2. 增量备份（OpLogBackup，Perl 工具）

* 执行常规备份后，启动备份daemon

* 从Slave pull oplog 转储到本地文件

* 每1分钟1个文件，时间戳（间隔可控）

* 用xz压缩

* rotate后传送到S3 bucket

* 执行常规备份后清理旧文件

数据库备份

自制工具的初衷 =》 省钱!

* 效果： 最坏可能丢失1～3分钟的数据

* 省钱， S3和EC2 之间无传输费用

* xz 压缩， S3存储成本可忽略不计， EBS贵!

* 通过oplog replay 可恢复到分钟之间的数据

MongoDB 2.1 的oplog， 可实现类似功能:

```
> oplog --from host --seconds times
```

数据库备份

3. 偶尔使用MongoDump备份

* 方便迁移

  * 速度最慢，数据量多或gridfs

    * 通过fork mongodump 并行dump db/collection 加快dump进程

    * 压缩bson文件节约费用

注意事项

务必预分配数据文件!

* 运行中分配? EBS反应太慢，查询超时

  * 一次性将db的data file创建达到预设

    * head -c 2146435072 /dev/zero > db.02

    * cp db.02 db.03 ...

注意事项

不使用GridFS

* AWS 环境使用GridFS 非常不经济

* 仅管理文件meta

* 文件本身存储在S3,用ObjectId做文件名

* 只创建/删除，不做更新操作

    * Delete请求无费用

一些必要提示

时刻关注数据碎片情况

* db.collection.stats() => paddingFactor

* update/remove 导致碎片产生

* 定期reIndex可改善减少碎片

一些必要提示

repair vs compact

* 都能对数据文件碎片处理

compact：

* 速度快，可单独collection, 但无法释放多余空间，实用性有限

一些必要提示

db.repairDatabase

* 速度缓慢，但处理彻底， paddingFactor 重置，更新速度会下降

* repair 临时目录可放在instance storage上

* 定期要对Primary进行repair

  P -> stepDown -> repair -> 切回

一些必要提示

工具

* mongostat => 关注 locked, faults
    * faults > 100, 内存非常不足
    * locked > 30, 写锁问题严重
* iostat => 检查EBS延迟

一些必要提示

打开DB Profileing （检查>200ms)

* db.setProfilingLevel(1,200)

* db.system.profile.find().sort({$natural:-1})

简单排查

查询 => nscanned（无索引)

更新：moved（doc需要重写，碎片)

一些必要提示

应用端需要考虑

* 审慎使用ORM，简单的封装driver更易于调控

* 使用短字段 ts 优于 login_time_stamp

    * app端做个字段map: t => timestamp

    * 省空间就是省钱 + 最简单的调优

一些必要提示

* 创建新文档时做padding
* 字段占位
* 变长字段放置最后
* 注意Ordered Hash
* BinData可用于占位padding

一些必要提示

* 只做相等比较的字段可用Binary data代替UTF8 String类型

  * 如可用BinData存储MD5而不是Md5 hex encoding string

* 结合denormalized 使用复合_id或字段索引，减少索引的使用

  _id: { uid, aid } // article comment

一些必要提示

FindAndModify要注意

* Write-lock, query切忌可以yield

使用Read-Update模式

* 提高索引命中，加热索引纪录

更新多个文档

* Loop and update-one优于使用update更新多文档，减少writelock伤害

其他

* GridFS 和业务数据分库(如果要用)
* 做Queue队列分库（Capped Collection）
* 用Perl写日常维护的脚本，少敲命令

项目环境

项目：创意图片分享交流，专业工具

一期目标： 初始存储 ~5千万张图片，每日更新5 – 10万张

* MongoDb 2.0.2
    * 4 x RS, ～30 db， 5 EC2 Large/xLarge
    * 1.5T
* mongodb主用途：
    用户数据，抓取队列，数据分析

项目环境

其他开发工具

* Padrino 0.10.5 / Mongoid（前台）

    > 计划开始向OpenResty迁移

* Redis / Gearman

* Perl (后台和维护)

Todo

Resty-Lua-Mongo

* Lua driver for OpenResty

* MIT License (3月底）

OpenResty的好处

* Nginx + Lua

* non-blocking IO，同步调用，比Node.js更易于迁移代码

* 高性能，低消耗 = 省钱!

Todo

写一本去年应该完成的书

* 《MongoDB实践》的开放出版

* 希望半年内完成 ;-(

* Sorry, 图灵的编辑同学们

也希望能够听取大家的建议

>> nightsailer@gmail.com

谢谢大家