
Mongo
Mongodb数据库调研
designed by Sun


目录
Index
Nosql背景
Mongodb介绍
数据操作
数据组织
索引


01
Part One
Nosql背景

1-1大数据时代

大交易数据，
大交互数据，
两类数据的有效融合
将是大势所趋。

大数据时代下的系统
需求3H——High
performance 、 Huge
Storage 、 High
Scalability && High
Availability

随着互联网产品和应
用的程度越来越深，
互联网将更加智能，
数据也将爆炸式增长

大数据的特征
3V——Volume、
Variety、Velocity

1-1大数据时代

1-2关系型DB弊端

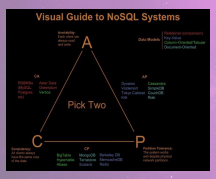
High performance
数据库读写压力巨大，硬盘IO无法承受
Master-Slave，主从分离

Huge Storage
存储记录数量有限，SQL查询效率极低
分库、分表，缓解数据增长压力

High Scalability && High Availability
横向扩展艰难，无法通过快速增加服务
器节点实现，系统升级和维护造成服务
不可用
Master -Master Replication Manager for
MySQL

各解决方案依然有着各自的缺点

1-3Nosql的关键



Consistency、 Availability、 Partition tolerance

- CAP原理：在分布式系统中，这三个要素最多只能同时实现两点，不可能三者兼顾

02 Part Two

Mongodb介绍

2-1简介

MongoDB的名称取自“humongous的中间部分，足见mongodb的宗旨在处理大量数据上面

面向文档存储
完整的索引支持
支持复制和故障恢复
易扩展

MongoDB是一个可扩展、高性能的下一代数据库，由C++语言编写，旨在为web应用提供可扩展的高性能数据存储解决方案。

模式自由
可自由更新数据结构
支持Map / Reduce
采用GridFS

2-2适用范围

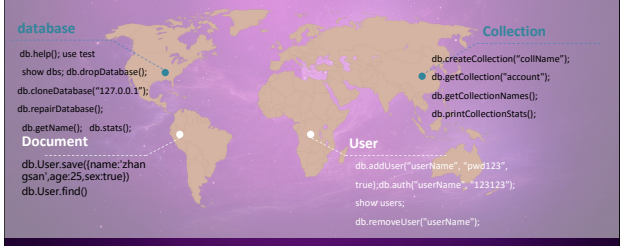
适用环境

- •网站数据、缓存、大尺寸，低价值的数据、
- •高伸缩性的场景（内涵mapreduce支持）
- •用于对象及JSON数据的存储

不适用环境

高度事务性的系统
传统的商业智能应用
复杂的跨文档(表)级联查询

2-3层次结构



database

```
db.help(); use test
show dbs; db.dropDatabase();
db.cloneDatabase("127.0.0.1");
db.repairDatabase();
db.getName();  db.stats();
```

Collection

```
db.createCollection("collName");
db.getCollection("account");
db.getCollectionNames();
db.printCollectionStats();
```

Document

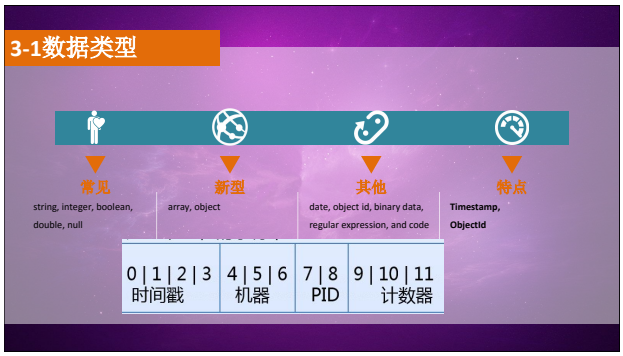
```
db.User.save({name:'zhangsan',age:25,sex:true})
db.User.find()
```

User

```
db.addUser("userName", "pwd123", true);db.auth("userName", "123123");
show users;
db.removeUser("userName");
```

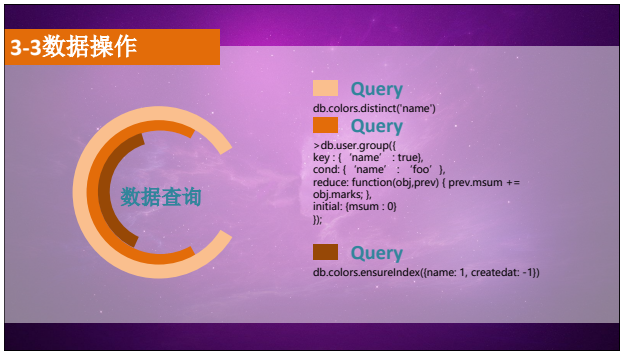
03 Part Three 数据操作

3-1数据类型



3-2数据操作

3-3数据操作



3-4数据操作

One to Many

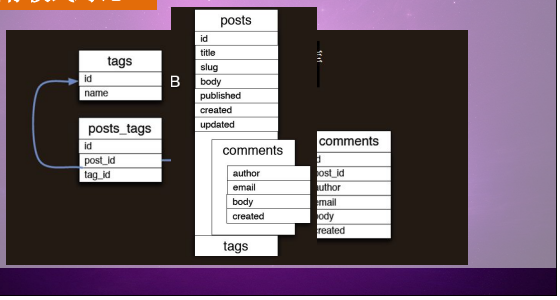
```
doc = { "_id" : 1, "author" : "sam", "title" : "i love you", "text" : "this is
a test", "tags" : [ "love", "test" ], "comments" : [ { "author" : "jim",
"comment" : "yes" }, { "author" : "tom", "comment" : "no" } ] }
db.posts.insert(doc);
```

Many to many

```
db.users.insert({name: 'John',authorizations: [
ObjectId('4bee280f9e89db4e12bf78e2'),ObjectId('4bee283
c9e89db4e12bf78e3')]})
var john = db.users.findOne({name:'John'})
db.sites.find({_id:{$in: john.authorizations}})
```

04 Part Four 数据组织

4-1储存模式对比



4-2储存引擎

使用内存映射文件mmap实现

分配策略

32位机器受地址空间限制，所以单个实例最大数据空间仅为2.5G左右，64位机器基本无限制（128T），故建议使用64位机器部署。

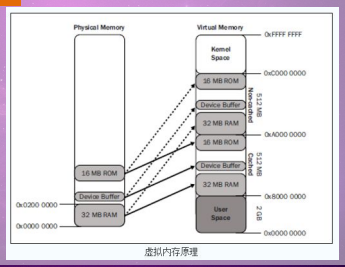
每个数据库由一个.ns元数据文件，以及多个数据文件组成（dbname.{0,1,2,.....}），以自增的数字为扩展名）。为了防止小数据库浪费空间，MongoDB的数据文件默认从16M开始，倍数级别增加，2G为单个数据文件的大小上限。
16M>32M>64M>128M>256M>512M>1G>2G>2G

GridFS

GridFS是一种将大型文件存储在MongoDB的文件规范。所有官方支持的驱动均实现了GridFS规范。

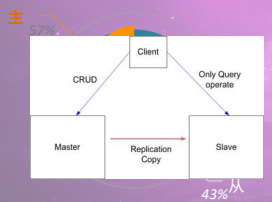
数据库支持以BSON格式保存二进制对象。但是MongoDB中BSON对象最大不能超过16MB。GridFS 规范提供了一种透明的机制，可以将一个大文件分割成为多个较小的文档。这将容许我们有效的保存大的文件对象，特别对于那些巨大的文件，比如视频。

4-2储存引擎



虚拟内存原理

4-3复制模式



Master-Slave

需要启动的两个MongoDB文档数据库，一个是以主模式启动，另一个属于从模式启动。因此，主服务器进程将创建一个local.oplog，将通过这个“交易记录”同步到Slave服务器中。
只需要在某一个服务启动时加上–master参数，而另一个服务加上–slave 与–source 参数，即可实现同步
slave服务器只可以执行读操作，不可以执行写操作。

4-3复制模式

Replica Sets

MongoDB 在 1.6 版本对开发了新功能replica set，这比之前的replication 功能要强大一些，增加了故障自动切换和自动修复成员节点，各个DB 之间数据完
全 一致，大大降低了维护成本。auto shard 已经明确说明不支持replication paris,建议使用replica set，replica set故障切换完全自动。

4-3复制模式

Replica Sets

Replica Set 架构是通过一个日志(oplog)来存储写操作的。
oplog.rs 是一个固定长度的 capped collection，它存在于"local"数据库中，用于记录 Replica Sets 操作日志。

```
rs1:PRIMARY> db.oplog.rs.find()
{ "ts" : { "t" : 1338459114000, "i" : 1 }, "h" :
NumberLong("5493127699725549585"), "op" :
"i","ns" : "test.c1", "o" :
{ "_id" : ObjectId("4fc743e9aea289af709ac6b5"),
"age" : 29, "name" :"Tony" } }
Ts、 op ( i、d、u )、 ns、o
```

4-4分片集群

1.一分片多区间
假设：把shard1的分片划分为二个区间([“a” ,” d”],[“d” , “f”]),一个区间的数据称为块(chunk),把一个块的区间一分为二时，一个块也就变成了二个块。

2.MongoDB的数据分配：当一个块变得越来越大时，MongoDB会自动将其分割成二个较小的块。如果分片间比例失调，则MongoDB会通过迁移块来确保均衡。

3.块的创建取决你选择的片键(shrad key),片键可以是任意字段和字段的组合。

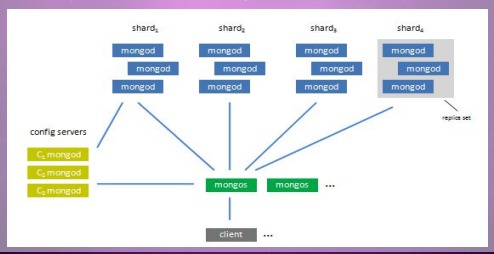
4.如果存在多个可用的分片，只要块的数量足够多，MongoDB就会把数据迁移到其他分片上。这个迁移的过程叫做平衡(balancing),由均衡器(balancer)的进程负 责执行。

5.文件使用MMAP进行内存映射，会将所有数据文件映射到内存中，但是只是虚拟内存，只有访问到这块数据时才会交换到物理内存。如果是读操作，内存中 的数据起到缓存的作用，如果是写操作，内存还可以把随机的写操作转换成顺 序的写操作。

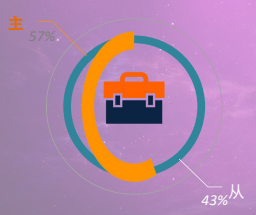
6.对每一个块来说，其头部包含了一些块的元数据，比如自己的位置，上一个和下一个块的位置以及块中第一条和最后一条记录的位置指针。剩下的部分用于存储具体的数据，具体数据之间也是通过双向链接来进行连接。

4-4分片集群

4-4分片集群



4-5高级分片集群



高级分片实例

MongoDB Auto-Sharding 解决了海量存储和动态扩容的问题，但离实际生产环境所需的高可靠(high reliability)、高可用(high availability)还有些距离。解决方案：

Shard: 使用 Replica Sets，确保每个数据节点都具有备份、自动容错转移、自动恢复能力。

Config: 使用 3 个配置服务器，确保元数据完整性(two-phase commit)。

Route: 配合 LVS，实现负载平衡，提高接入性能(high performance)。

05 Part Four
索引

5-1索引

MongoDB的索引跟传统数据库的索引相似，一般的如果在传统数据库中需要建立索引的字段，在MongoDB中也可以建立索引。 MongoDB中_id字段默认已经建立了索引，这个索引特殊，并且不可删除，不过Capped Collections例外。

1. 建立索引
2. 使用索引
3. 查看索引
4. 删除索引
5. 重建索引

5-2索引

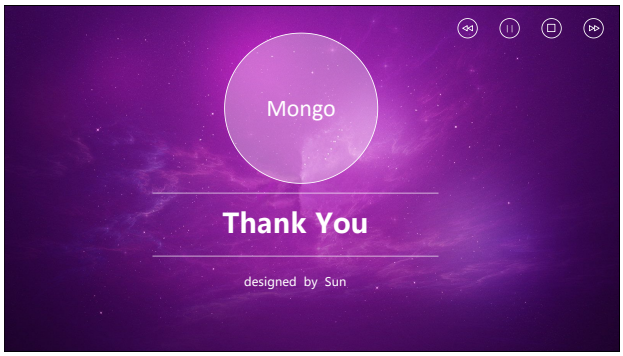
建立索引
db.persons.ensureIndex({name:1});
使用索引
db.persons.find({name :'sam'});

文档索引
db.factories.insert({ name: "xyz",
metro: { city: "New York", state:
"NY" } });

唯一性索引
db.things.ensureIndex({firstname: 1,
lastname: 1},

返回索引
db.persons.getIndexes();

删除索引
db.runCommand({dropIndexes:'foo',
index : {y:1}})
db.collection.dropIndex({x: 1, y: -1})


Mongo
Thank You
designed by Sun