

## <<深入学习MongoDB>>

### 图书基本信息

书名：<<深入学习MongoDB>>

13位ISBN编号：9787115272119

10位ISBN编号：7115272115

出版时间：2012-3

出版单位：人民邮电出版社

作者：Kristina Chodorow

页数：121

译者：巨 成,程显峰

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## <<深入学习MongoDB>>

### 内容概要

本书分两部分，分别来自O'Reilly的《MongoDB扩展技术》与《MongoDB开发技巧50例》两书。

前一部分“MongoDB扩展技术”指导大家创建一个不断增长以满足应用程序需求的MongoDB集群，内容简明扼要，指导用户设置和使用集群存储大量数据并高效访问数据。此外，读者还可了解如何让应用程序兼容分布式数据库系统。

具体的主题有：

通过分片设置MongoDB集群；

在集群中查询和更新数据；

操作、监控和备份集群；

从程序设计角度，考虑如何应对分片、配置服务器或者mongos进程停止运行的情况。

遵照其中建议，你很快就可通过MongoDB构建和运行一个高效的、可预测的分布式系统。

对于用户而言，MongoDB上手很容易，但是构建使用MongoDB的应用程序时，一些棘手的问题便会接踵而来。

怎样权衡范式化与反范式化？

怎样处理复制

组失效的情况并进行故障恢复？

本书第二部分“MongoDB开发技巧50例”呈现了一系列的MongoDB提示和技巧，可帮助用户解决与应用程序设计与实

现、数据安全和监控有关的各种问题。

内容涵盖10gen公司工程师的实际指导，并通过以下5个话题展开了论述。

应用设计技巧：模式设计阶段应注意的问题

实现技巧：基于MongoDB编写应用程序

优化技巧：为应用提速

数据安全技巧：在不牺牲太多性能的情况下，利用复制和日志保证数据安全

管理技巧：配置MongoDB并确保其平滑运行

## <<深入学习MongoDB>>

### 作者简介

Kristina Chodorow

10gen公司的软件工程师，MongoDB项目的核心成员，从事与数据库服务器、PHP驱动、Perl驱动等相关的工作。

她常在世界级技术大会上作报告，包括OSCON、LinuxCon、FOSDEM和Latinoware。

## &lt;&lt;深入学习MongoDB&gt;&gt;

## 书籍目录

## MongoDB 扩展技术

## 第1章 欢迎来到分布式计算的世界

## 第2章 理解分片

## 2.1 分割数据

## 2.1.1 分配数据

## 2.1.2 如何创建块

## 2.2 平衡

## 2.3 mongos

## 2.4 配置服务器

## 2.5 集群的构造

## 第3章 建立集群

## 3.1 选择片键

## 3.1.1 小基数片键

## 3.1.2 升序片键

## 3.1.3 随机片键

## 3.1.4 好片键

## 3.2 新老集合分片

## 3.2.1 快速起步

## 3.2.2 配置服务器

## 3.2.3 mongos

## 3.2.4 分片

## 3.2.5 数据库和集合

## 3.3 增减容量

## 3.3.1 移除分片

## 3.3.2 修改分片中的服务器

## 第4章 使用集群

## 4.1 查询

## 4.2 为什么会这样

## 4.2.1 计数

## 4.2.2 唯一索引

## 4.2.3 更新

## 4.3 MapReduce

## 第5章 管理

## 5.1 使用命令行

## 5.1.1 了解概况

## 5.1.2 配置集合

## 5.1.3 应该连接什么

## 5.2 监控

## 5.2.1 mongostat

## 5.2.2 Web 管理界面

## 5.3 备份

## 5.4 关于架构的建议

## 5.4.1 创建应急站点

## 5.4.2 挖护城河

## 5.5 错误处理

## &lt;&lt;深入学习MongoDB&gt;&gt;

- 5.5.1 分片停机
- 5.5.2 多数分片停机
- 5.5.3 配置服务器停机
- 5.5.4 mongos 进程死掉
- 5.5.5 其他注意事项

## 第6章 学习资源

## MongoDB 开发技巧50例

## 第1章 应用设计技巧

- 1.1 技巧1：速度和完整性的折中
  - 1.1.1 示例：网上购物车
  - 1.1.2 考虑因素
- 1.2 技巧2：适应未来的数据要范式化
- 1.3 技巧3：尽量单个查询获取数据
  - 1.3.1 示例：博客
  - 1.3.2 示例：相册
- 1.4 技巧4：嵌入关联数据
- 1.5 技巧5：嵌入时间点数据
- 1.6 技巧6：不要嵌入不断增加的数据
- 1.7 技巧7：预填充数据
- 1.8 技巧8：尽可能预先分配空间
- 1.9 技巧9：用数组存放要匿名访问的内嵌数据
- 1.10 技巧10：文档要自给自足
- 1.11 技巧11：优先使用\$ 操作符
  - 1.11.1 深入了解
  - 1.11.2 提高性能
- 1.12 技巧12：随时聚合
- 1.13 技巧13：编写代码处理数据完整性问题

## 第2章 实现技巧

- 2.1 技巧14：使用正确的类型
- 2.2 技巧15：用简单唯一的id 替换\_id
- 2.3 技巧16：不要用文档做\_id
- 2.4 技巧17：不要用数据库引用
- 2.5 技巧18：不要用GridFS 处理小的二进制数据
- 2.6 技巧19：处理“无缝”故障切换
- 2.7 技巧20：处理复制组失效及故障恢复

## 第3章 优化技巧

- 3.1 技巧21：尽可能减少磁盘访问
- 3.2 技巧22：使用索引减少内存占用
- 3.3 技巧23：不要到处使用索引
- 3.4 技巧24：索引覆盖查询
- 3.5 技巧25：使用复合索引加快多个查询
- 3.6 技巧26：通过建立分级文档加速扫描
- 3.7 技巧27：AND 型查询要点
- 3.8 技巧28：OR 型查询要点

## 第4章 数据安全性和一致性

- 4.1 技巧29：单机做日志，多机则复制
- 4.2 技巧30：坚持使用复制或日志，或两者兼用

## <<深入学习MongoDB>>

- 4.3 技巧31：不要信任repair 恢复的数据
  - 4.4 技巧32：getlasterror
  - 4.5 技巧33：开发过程中一定要使用安全写入
  - 4.6 技巧34：使用w 参数
  - 4.7 技巧35：一定要给w 设置超时
  - 4.8 技巧36：不要每次写入都调用fsync
  - 4.9 技巧37：崩溃之后正常启动
  - 4.10 技巧38：持久性服务器的瞬时备份
- 第5章 管理技巧
- 5.1 技巧39：手工清理块集合
  - 5.2 技巧40：用repair 压缩数据库
  - 5.3 技巧41：不要改变复制组成员投票的权值
  - 5.4 技巧42：无活跃节点时可重置复制组
  - 5.5 技巧43：不必指定--shardsvr 和--configsvr 参数
  - 5.6 技巧44：开发时才用--notablesan
  - 5.7 技巧45：学习JavaScript
  - 5.8 技巧46：在shell 中管理所有服务器和数据库
  - 5.9 技巧47：获得帮助
  - 5.10 技巧48：创建启动文件
  - 5.11 技巧49：自定义函数
  - 5.12 技巧50：使用单个连接读取自身写入

## &lt;&lt;深入学习MongoDB&gt;&gt;

## 章节摘录

版权页：插图：1.适用的键 这条规则适用于任何升序排列的键值，而并不必须是时间戳。其他例子包括objecrid、日期、自增主键（很可能是从其他数据库导入的）。

只要键值趋向于无穷大，你就会面临这个问题。

2.例外 基本上，这种片键总是一个坏主意，因为它导致热点必然存在。

如果访问量不大且用一个分片就能承受所有读写，那还行得通。

当然，如果遇到一个访问量尖峰或者应用开始变得更受欢迎，那它终会停止工作并且难以修复。

除非你非常清楚自己在干什么，否则不要使用升序片键。

肯定还有更好的片键存在，应该避免使用这一个。

3.1.3随机片键 有时为了避免热点，人们会选择一个取值随机的字段来分片。

采用这种片键一开始还不错，但是随着数据量越变越大，它会变得越来越慢。

假设我们在分片集合中存储照片的缩略图。

每个文档都包含了照片的二进制数据、二进制数据的MD5散列值，以及一段描述、拍照时间和拍照人

。

我们决定在MD5散列值上做分片。

随着集合的增长，我们最终会得到一组均匀分布于各分片的数据块。

目前为止一切顺利。

现在，假设我们非常忙而分片2上的一个块填满并分裂了。

配置服务器注意到分片2比分片1多出了10个块并判定应该抹平分片间的差距。

这样MongoDB就需要随机加载5个块的数据到内存中并将其发送给分片1。

考虑到数据序列的随机性，一般情况下这些数据可能不会出现在内存中。

所以此时的MongoDB会给RAM带来更大压力，而且还会引发大量磁盘IO（磁盘IO总是非常慢）。

除此以外，片键上必须有索引，因此如果选择了从不依据它进行查询的随机键，基本上可以说是“浪费”了一个索引。

另外，考虑到每增加一个索引都会让写操作变得更慢，所以保持索引数量尽可能低也是非常重要的。

3.1.4好片键 我们真正需要的是一种将访问模式也考虑进去的方案。

如果应用会规律性地访问25GB的数据，我们就希望所有的分割和迁移都发生在这25 GB数据上，而不是随机访问数据以至于不断地有新数据被从磁盘中复制到内存里。

因此我们希望能找到这样一个片键，它具备良好的数据局部性（data locality）特征，但又不会因太局部而导致热点出现。

## <<深入学习MongoDB>>

### 编辑推荐

《深入学习MongoDB》适合所有MongoDB用户阅读参考。



## <<深入学习MongoDB>>

### 版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>