MongoDB 核心贡献者：不是 MongoDB 不行，而是你不懂！

近期MongoDB在Hack News上是频繁中枪。许多人更是声称恨上了MongoDB，David mytton 就在他的博客中揭露了 MongoDB 许多现存问题。然而恨的人有之偏爱的也同样很多，作为回击：Russell Smith 带来了多年工作经验的总结。Russell Smith 曾担任 Ops 和大型网站缩放顾问并且帮助过 Guardian、Experian 等多家公司，MongoDB London User Group 的联合创始人。作为 MongoDB Master（MongoDB 官方认可的 MongoDB 核心贡献者组织，并通过社区分享自己的专业技术），其参与工作的基础设施单服务器每秒查询超过 3 万次，每天活跃数据更在 1TB 以上。

下面来看 Russell 对 MongoDB 一些常见及生僻的问题做出分析：

32 位 vs 64 位

现在大多数的服务器都对 32 位操作系统实现支持，更有许多新型硬件支持着允许更多 RAM 的 64 位操作系统。

MongoDB 也同时发布了 32 位及 64 位两个版本的数据库。归结于 MongoDB 使用的内存映射文件，32 位版本只支持 2G 数据的存储。对于标准的 Replica Set，MongoDB 只拥有单一的处理策略 —— mongod。如果你想在未来储存 2G 以上的数据，请使用 64 位版本的 MongoDB。如果拥有分片安装，那么 32 位版本同样可以使用。

总结：使用 64 位版本或者理解 32 位版本的限制。

文件大小限制

不同于 RDBMS 把数据储存在行与列中，MongoDB 的数据是储存在文件中的。这些文件使用二进制存储形式，其格式为类似 JSON 格式的 BSON 格式。

和其它的数据库一样，单个文件的储存大小是有限制的。在旧版本的 MongoDB 中，单个文件都限制在 4M 以内。而新版本的 MongoDB 单文件已经支持到 16M 大小。这样的限制也许是令人厌烦的，但是 10gen 的意见是：如果这项设置不停的困扰到你，那么是否你的设计模式存在着问题；或者你可以使用 文件无大小限制的 GridFS。

这种情况通常的建议是避免存储过大的文件，不定期的更新数据库中存储的各种对象。而像 Amazon S3 或者 Rackspace Cloudfiles 这样的服务通常可能会是更好的选择，而非必要情况下最好别让基础设施陷入过载。

总结：把每个文件保持在 16M 以下，那么一切都好。

写入失败

MongoDB 在默认的情况下允许高速的写入和更新，而付出的代价就是没有明确的错误通知。默认情况下多数的驱动都在做异步、“不安全”写入 —— 这就意味着驱动程序不能立即反馈错误信息，类似于 MySQL 的 INSERT DELAYED。如果你想知道某个事情是否成功，你必须使用 getLastError 手动的检查错误信息。

某些情况下如果你需要在错误发生后立刻得到错误信息，即：大多数的驱动中都很容易实现同步“安全”查询。这将谋杀掉 MongoDB 不同于传统数据库的优点。

如果对比“完全安全”的同步写入你需要多一点性能，同时还想要一定程度的安全，那么你可以使用 getLastError with ‘j’ 让 MongoDB 只到一份日志提交后再发出错误报告通知。那么日志将以 100 毫秒一次的速度输出到磁盘，而不是 60 秒。

总结：如果必须要写入确认，你可以使用安全写入或 getLastError。

数据结构模型的弱化不等于没有数据结构模型

RDBMS 一般都拥有一个预定义的数据结构模型：表格的行和列，每个字段都拥有名称和数据类型。如果你想给其中一行加一列，那么你必须给整个表格都添加一列。

MongoDB 则是移除了这个设置，对于 Collection 和文件没有强制的模型限定。这有益于快速开发及简易修改。

当然这不意味着你就可以无视结构模型的设计，一个合适的结构模型可以让你获得 MongoDB 的最佳性能。赶快阅读 MongoDB 文档，或者观看这些结构模型设计的相关视频吧！

- Schema Design Basics
- Schema Design at Scale
- Schema Design Principles and Practice

总结：设计结构模型并充分利用 MongoDB 的特色。

默认情况下修改语句修改的只是单个文件

在传统的 RDBMS 中除非使用 LIMIT 子句，修改语句作用的将是所有匹配的地方。然而 MongoDB 每个查询上都默认使用等价“LIMIT 1”的设置。虽然无法做到“LIMIT 5”，但是你可以通过下面的语句整个的移除限制：

```
db.people.update({age: {$gt: 30}}, {$set: {past_it: true}}, false, true)
```

同样在官方的驱动中还有类似的选项 —— ‘multi’。

总结：可以通过指定多个文件的 multi 为 true 来完成多文件修改

查询区分大小写

字符串的查询可能不按预期的那样发展 —— 这归结于 MongoDB 默认区分大小写。

例如：db.people.find({name: ‘Russell’})与 db.people.find({name: ‘russell‘}) 是不同的。在这里最理想的解决方案就是对需要查询数据进行确认。你也可以通过正则表达式进行查询，比 如：db.people.find ({name:/Russell/i})，但是这样会影响到性能。

总结：查询是区分大小写的，在牺牲速度的情况下可以利用正则表达式。

对输入的数据无容错性

当你尝试向传统数据库插入错误类型的数据，传统的数据库一般会把数据转换成预定义的类型。然而这在 MongoDB 中是行不通的，因为 MongoDB 的文件是没有预定义数据模型的。这样的话 MongoDB 会插入你输入的任何数据。

总结：使用准确的数据类型。

关于锁

当资源被代码的多个部分所共享时，需要确信锁必须要确保这处资源只能在一个地方被操作。

旧版本的 MongoDB （pre 2.0）拥有一个全局的写入锁。这就意味贯穿整个服务器中只有一个地方做写操作。这就可能导致数据库因为某个地方锁定超负载而停滞。这个问题在 2.0 版本 中的得到了显著的改善，并且在当前 2.2 版本中得到了进一步的加强。MongoDB 2.2 使用数据库级别的锁在这个问题上迈进了一大步。同样值得期待的 Collection 级别的锁也计划在下一个版本中推出。

尽管如此，Russell 还是认为：大多数受此限制的应用程序于其说是受 MongoDB 影响，还不如说是程序本身的问题来的更直接。

总结：使用最新的稳定版本才能获得最高的性能。

关于包

在类 Ubuntu 和 Debian 系统上安装时，许多人都出现过“过时版本”这样的问题。解决方案很简单：使用 10gen 官方库，那么在 Ubuntu 和 Debian 上安装也会像在 Fedora 和 Centos 上安装一样流畅。

总结：使用拥有大多数最新版本的官方包。

使用偶数个 Replica Set 成员

Replica Set 是增加冗余及提升 MongoDB 数据集群性能的有效途径。数据在所有的节点中被复制，并选出一个作为主节点。假如主节点出故障，那么会在其他的节点中票选一个作为新的主节点。

在同一个 Replica Set 中使用两台机器是很有诱惑的，它比 3 台机器来的便宜并且也是 RDBMS 的标准行事风格。

但是到了 MongoDB 这里，同一个 Replica Set 中的成员数量只能是奇数个。假如你使用了偶数个成员，那么当主节点发生故障时那么其它的节点都会变成只读。发生这种情况是因为剩下待选节点的数目不满足票选主节点的规定。

如果你想节约成本，同时还希望支持故障转移和冗余的增强，那么你可以使用 Arbiter。Arbiter 是一种特殊的 Replica Set 成员，它不储存任何用户数据（这就意味着他们可以使用非常小的服务器）。

总结：只可以使用偶数个 Replica Set 成员，但是可以使用 Arbitter 来削减成本。

没有 join 语句

MongoDB 不支持 join：如果你想在多个 Collection 中检索数据，那么你必须做多次的查询。

如果你觉得你手动做的查询太多了，你可以重设计你的数据模型来减少整体查询的数量。MongoDB 中的文件可以是任何类型，那么可以轻易的对数据进行 De-Normalize。这样就可以让它始终和你的应用程序保持一致。

总结：没有 join 不妨看一下如何设计数据结构模型。

Journaling

MongoDB 使用内存映射文件并且每 60 秒向磁盘输出一次通知，这就意味着最大程度上你可能丢失 60 秒加上向硬盘输出通知这段时间内所有的数据。

为了避免数据丢失，MongoDB 从 2.0 版本起就添加了 Journaling（默认情况下开启）。Journaling 把时间从 60 秒更改为 100ms。如果数据库意外

的停机，在启动之前它将会被重启用以确保数据库处于一致状态。这也是MongoDB 与传统数据库最接近的地方。

当然 Journaling 会轻微的影响到性能，大约 5%。但是对于多数人来说额外带来的安全性肯定是物有所值的。

总结：最好别关闭 Journaling。

默认情况下没有身份认证

MongoDB 在默认设置下并没有身份验证。MongoDB 会认为自身处在一个拥有防火墙的信任网络。但是这不代表它不支持身份验证，如果需要可以轻松的开启。

总结：MongoDB 的安全性可以通过使用防火墙和绑定正确的接口来保证，当然也可以开启身份验证。

Replica Set 中损失的数据

使用 Replica Set 是提高系统可靠性及易维护的有效途径。这样的话，弄清节点间故障的发生及转移机制就变得至关重要。

Replica Set 中的成员一般通过 oplog（记录了数据中发生增、删、改等操作的列表）来传递信息，当其中一个成员发生变化修改 oplog 后，其他的成员也将按照 oplog 来执行。如果你负责处理新数据的节点在出错后恢复运行，它将会被回滚至最后一个 oplog 公共点。然而在这个过程中：丢失的“新数据”已经被 MongoDB 从数据库中转移并存放到你的数据目录‘rollback’里面等待被手动恢复。如果你不知道这个特性，你可能就会认为数据被弄丢了。所以每 当有成员从出错中恢复过来都必须要检查这个目录。而通过 MongoDB 发布的标准工具来恢复这些数据是件很容易的事情。查看官方文档以了解更多相关信息。

总结：故障恢复中丢失的数据将会出现在 rollback 目录里面。

分片太迟

分片是把数据拆分到多台机器上，通常被用于 Replica Set 运行过慢时进行性能提升。MongoDB 支持自动分片。然而如果你让分片进行太迟的话，问题就产生了。因为对数据的拆分和块的迁移需要时间和资源， 所以如果当服务器资源基本上耗尽时很可能会导致在你最需要分片时却分不了片。

解决的方法很简单，使用一个工具对 MongoDB 进行监视。对你的服务器做最准确的评估，并且在占整体性能的 80%前进行分片。类似的监视工具有：MMS、Munin（+Mongo Plugin）和 CloudWatch。

如果你确定从一开始就要分片处理，那么更好的建议会是选用 AWS 或者类似的云服务进行分片。而在小型服务器上，关机或者是调整机器明显比转移成千上万条数据块来的更直接一点。

总结：尽早的分片才能有效的避免问题。

不可以更改文件中的 shard key

对于分片设置，shard key 是 MongoDB 用来识别分块对应文件的凭证。当你插入一个文件后，你就不可以对文件的 shard key 进行更改。而这里的解决方案是把文档删除然后重新建立，这样就允许把它指定到对应的分块了。

总结：shard key 不可以修改，必要的时候可以删除文件重新建立。

不可以对 256G 以上的 Collection 进行分片

重新回到分片太迟的问题上来 —— MongoDB 不允许对增长到 256G 以上的 Collection 进行分片，之前版本的设置还没有 256G。这个限定在以后肯定会被移除，而这里也没有更好的解决方案。只能进行重编译或者把大小控制在 256G 以下。

总结：在 Collection 达到 256G 以前进行分片。

唯一性索引与共享

索引的唯一性约束只能通过 shard key 来保证。

更多详情

选择了错误的 shard key

MongDB 需要你选择一个 shard key 来将数据分片。如果选择了错误的 shard key,更改起来将是件很麻烦的事情。

点击查看如何更改

总结：选择 shard key 之前先阅读这个文档。

与 MongoDB 通信的未经加密

与 MongoDB 的连接默认情况下都是非加密的，这就意味你的数据可能被第三方记录和使用。如果你的 MongoDB 是在自己的非广域网下使用，那么这种情况是不可能发生的。

然而如果你是通过公网访问 MongoDB 的话，那么你肯定会希望你的通信是经过加密的。公版的 MongoDB 是不支持 SSL 的。庆幸的是可以非常简 单的定制自己的版本。10gen 的用户则拥有特别定制的加密版本。幸运的是大部分的官方驱动都支持 SSL，但是小麻烦同样是不可避免的。点击查看文档。

总结：当用公网连接时，要注意和 MongoDB 的通信是未加密的。

事务

不像 MySQL 这些支持多行数据原子操作的传统数据库，MongoDB 只支持单文件的原子性修改。解决这个问题的方法之一是在应用程序中使用异步提交的方式；另一个是：建立一个以上的数据存储。虽然第一种方法并不适用于所有情况，但是很显然比第二个来的要好。

总结：不支持对多文件事务。

日志预分配慢

MongDB 可能会告诉你已经准备就绪，但事实上它还在对日志进行分配。如果你选择了让机器自行分配，而恰巧你的文件系统和磁盘速度又很慢，那么烦恼的事情发生了。通常情况下这不会成为问题，但是一旦出现了可以使用 undocumented flag --noprealloc j 来关闭预分配。

总结：如果机器文件系统和磁盘过慢的话，那么日志的预分配也可能很慢。

NUMA + Linux +MongoDB

Linux、NUMA 与 MongoDB 遇到一起的时候运行总是不会很好。如果你在 NUMA 硬件上运行 MongoDB 的话，这里建议是直接关掉。因为各种奇怪的问题随之而来，比如：速度会阶段性或者在 CPU 占用率很高的时候大幅下降。

总结：禁 NUMA。

Linux 里面的进程限制

如果你在 MongoDB 未满载的时候出过 SEGMENTATION FAULT 错误，你可能会发现这是因为使用了过低或者默认的打开文件或用户进程限制。10gen 建议把限制设置在 4K+，然而设置的大小该取决具体情况。阅读 ulimit 了解更多。

总结：长久的为 MongoDB 在 Linux 加上软或硬的打开文件或用户进程限制。

原文链接：MongoDB Gotchas & How To Avoid Them （编译/仲浩 包研/审校）