大规模分布式系统 作业七

钟诚 16307110259

Q：分析总结 NoSQL 数据库与关系数据库的差异和各自特点。请具体介绍两种不同类型的 NoSQL 数据库系统的架构，特点和应用场景

1. NoSQL 数据库与关系数据库的差异和各自特点

1.1 NoSQL 数据库与关系数据库的差异

对 NoSQL 数据库和关系数据库进行比较，差异如下表所示：[1]

比较标准	RDBMS	NoSQL	备注
数据库原理	完全支持	部分支持	RDBMS 有关系代数理论作为基础 NoSQL 没有统一的理论基础
数据规模	大	超大	RDBMS 很难实现横向扩展，纵向扩展的空间也比较有限，性能会随着数据规模的增大而降低 NoSQL 可以很容易通过添加更多设备来支持更大规模的数据
数据库模式	固定	灵活	RDBMS 需要定义数据库模式，严格遵守数据定义和相关约束条件 NoSQL 不存在数据库模式，可以自由灵活定义并存储各种不同类型的数据
查询效率	快	可以实现高效的简单查询，但是不具备高度结构化查询等特性，复杂查询的性能不尽人意	RDBMS 借助于索引机制可以实现快速查询（包括记录查询和范围查询） 很多 NoSQL 数据库没有面向复杂查询的索引，虽然 NoSQL 可以使用 MapReduce 来加速查询，但是，在复杂查询方面的性能仍然不如 RDBMS
一致性	强一致性	弱一致性	RDBMS 严格遵守事务 ACID 模型，可以保证事务强一致性 很多 NoSQL 数据库放松了对事务 ACID 四性的要求，而是遵守 BASE 模型，只能保证最终一致性

[1] 《大数据技术原理与应用》相关课件 林子雨

数据完整性	容易实现	很难实现	任何一个 RDBMS 都可以很容易实现数据完整性，比如通过主键或者非空约束来实现实体完整性，通过主键、外键来实现参照完整性，通过约束或者触发器来实现用户自定义完整性 但是，在 NoSQL 数据库却无法实现
扩展性	一般	好	RDBMS 很难实现横向扩展，纵向扩展的空间也比较有限 NoSQL 在设计之初就充分考虑了横向扩展的需求，可以很容易通过添加廉价设备实现扩展
可用性	好	很好	RDBMS 在任何时候都以保证数据一致性为优先目标，其次才是优化系统性能，随着数据规模的增大，RDBMS 为了保证严格的一致性，只能提供相对较弱的可用性 大多数 NoSQL 都能提供较高的可用性
标准化	是	否	RDBMS 已经标准化（SQL） NoSQL 还没有行业标准，不同的 NoSQL 数据库都有自己的查询语言，很难规范应用程序接口
技术支持	高	低	RDBMS 经过几十年的发展，已经非常成熟，Oracle 等大型厂商都可以提供很好的技术支持 NoSQL 在技术支持方面仍然处于起步阶段，还不成熟，缺乏有力的技术支持
可维护性	复杂	复杂	RDBMS 需要专门的数据库管理员(DBA)维护 NoSQL 数据库虽然没有 DBMS 复杂，也难以维护
存储方式	表格式	大块组合	关系型数据库是表格式的，因此存储在表的行和列中。他们之间很容易关联协作存储，提取数据很方便。 NoSQL 数据库则与其相反，他是大块的组合在一起。通常存储在数据集中，就像文档、键值对或者图结构。

存储规范	关系表	平面数据集	关系型数据库的数据存储为了更高的规范性，把数据分割为最小的关系表以避免重复，获得精简的空间利用。虽然管理起来很清晰，但是单个操作设计到多张表的时候，数据管理就显得有点麻烦。 Nosql 数据存储在平面数据集中，数据经常可能会重复。单个数据库很少被分隔开，而是存储成了一个整体，这样整块数据更加便于读写
高并发读写	效率低	性能高	关系型数据库为了维护数据的一致性付出了巨大的代价，读写性能比较差。在面对高并发读写性能非常差，面对海量数据的时候效率非常低。 NoSQL 存储的格式都是 key-value 类型的，并且存储在内存中，非常容易存储，而且对于数据的 一致性是 弱要求。NoSQL 无需 SQL 的解析，提高了读写性能。
成本	高	低	关系型数据库通常有 SQL Server，Mysql，Oracle。大多数的关系型数据库都是付费的并且价格昂贵，成本较大。 主流的 NoSQL 数据库有 redis，memcache，MongoDb。而 NoSQL 数据库通常都是开源的。

1.2 NoSQL 数据库的特点

NoSQL 是一种不同于关系数据库的数据库管理系统设计方式，是对非关系型数据库的一类统称，它采用的数据模型并非传统关系数据库的关系模型，而是类似键/值、列族、文档等非关系模型。它打破了长久以来关系型数据库与 ACID 理论大一统的局面，NoSQL 数据存储不需要固定的表结构，通常也不存在连接操作。NoSQL 数据库不适用传统的关系数据库模型，而是使用文档型的、列存储、图形数据库等方式存储数据模型。
NoSQL 数据库的特点主要有以下三种:

（1） 灵活的可扩展性

传统的关系型数据库由于自身设计机理的原因，通常很难实现“横向扩展”，在面对数据库负载大规模增加时，往往需要通过升级硬件来实现“纵向扩展”。但是，当前的计算机硬件制造工艺已经达到一个限度，性能提升的速度开始趋缓，已经远远赶不上数据库系统负载的增加速度，而且配置高端的高性能服务器价格不菲，因此寄希望于通过“纵向扩展”满足实际业务需求，已经变得越来越不现实。相反，“横向扩展”仅需要非常普通廉价的标准化刀片服务器，不仅具有较高的性价比，也提供了理论上近乎无限的扩展空间。NoSQL 数据库在设

计之初就是为了满足“横向扩展”的需求，因此天生具备良好的水平扩展能力。[2]

（2） 灵活的数据模型

关系模型是关系数据库的基石，它以完备的关系代数理论为基础，具有规范的定义，遵守各种严格的约束条件。这种做法虽然保证了业务系统对数据一致性的需求，但是过于死板的数据模型，也意味着无法满足各种新兴的业务需求。相反，NoSQL 数据库天生就旨在摆脱关系数据库的各种束缚条件，摈弃了流行多年的关系数据模型，转而采用键/值、列族等非关系模型，允许在一个数据元素里存储不同类型的数据。

以下是四种主流 NoSQL 数据库：

键值对存储（key-value）：相关产品有 Tokyo Cabinet、Redis 等，数据模型为一系列键值对，应用于处理大量数据的高访问负载。优势为快速查询，缺点为存储数据缺少结构化，

列存储数据库： 相关产品有 Cassandra, HBase，数据模型为列模式，将同一列数据存在一起，应用于分布式文件系统；优点是查找速度快、可扩展性强，更容易进行分布式扩展；缺点是功能相对局限

文档型数据库：相关产品有 MongoDB，数据模型为一系列键值对，应用于 Web，优点是数据结构要求不严格，缺点是查询性能不高，缺乏统一的查询语法

图形数据库：相关产品有 Neo4J，InfoGrid，数据模型为图结构，应用于社交网络、推荐系统筹和构建关系图谱，优点是利用了图结构的相关算法，缺点是需要整个图的计算而不容易进行分布式的集群方案。[3]

（3）与云计算紧密融合

云计算具有很好的水平扩展能力，可以根据资源使用情况进行自由伸缩，各种资源可以动态加入或退出，NoSQL 数据库可以凭借自身良好的横向扩展能力，充分自由利用云计算基础设施，很好地融入到云计算环境中，构建基于 NoSQL 的云数据库服务。

1.3 关系型数据库的特点

关系型数据库，是指采用了关系模型来组织数据的数据库，其以行和列的形式存储数据，以便于用户理解，关系型数据库这一系列的行和列被称为表，一组表组成了数据库。用户通过查询来检索数据库中的数据，而查询是一个用于限定数据库中某些区域的执行代码。关系模型可以简单理解为二维表格模型，而一个关系型数据库就是由二维表及其之间的关系组成的一个数据组织。[4]

关系型数据库有以下特点：

1. 存储方式：传统的关系型数据库采用表格的储存方式，数据以行和列的方式进行存储，要读取和查询都十分方便。

2. 存储结构：关系型数据库按照结构化的方法存储数据，每个数据表都必须对各个字段定义好（也就是先定义好表的结构），再根据表的结构存入数据，这样做的好处就是由于数据的形式和内容在存入数据之前就已经定义好了，所以整个数据表的可靠性和稳定性都比较高，但带来的问题就是一旦存入数据后，如果需要修改数据表的结构就会十分困难。

3. 存储规范：关系型数据库为了避免重复、规范化数据以及充分利用好存储空间，把

---

[2] https://blog.csdn.net/linjiayina/article/details/105378589

[3] https://blog.csdn.net/qq_20042935/article/details/89949078

[4] 周乾. 关系型数据库的特殊应用[J]. 大东方, 2016, 000(005):P.208-208.

数据按照最小关系表的形式进行存储，这样数据管理的就可以变得很清晰、一目了然，当然这主要是一张数据表的情况。如果是多张表情况就不一样了，由于数据涉及到多张数据表，数据表之间存在着复杂的关系，随着数据表数量的增加，数据管理会越来越复杂。

4.扩展方式：由于关系型数据库将数据存储在数据表中，数据操作的瓶颈出现在多张数据表的操作中，而且数据表越多这个问题越严重，如果要缓解这个问题，只能提高处理能力，也就是选择速度更快性能更高的计算机，这样的方法虽然可以一定的拓展空间，但这样的拓展空间一定有非常有限的，也就是关系型数据库只具备纵向扩展能力。

5.查询方式：关系型数据库采用结构化查询语言（即 SQL）来对数据库进行查询，SQL 早已获得了各个数据库厂商的支持，成为数据库行业的标准，它能够支持数据库的 CRUD（增加，查询，更新，删除）操作，具有非常强大的功能，SQL 可以采用类似索引的方法来加快查询操作。

6.规范化：在数据库的设计开发过程中开发人员通常会面对同时需要对一个或者多个数据实体（包括数组、列表和嵌套数据）进行操作，这样在关系型数据库中，一个数据实体一般首先要分割成多个部分，然后再对分割的部分进行规范化，规范化以后再分别存入到多张关系型数据表中，这是一个复杂的过程。好消息是随着软件技术的发展，相当多的软件开发平台都提供一些简单的解决方法，例如，可以利用 ORM 层（也就是对象关系映射）来将数据库中对象模型映射到基于 SQL 的关系型数据库中去，以及进行不同类型系统的数据之间的转换。

7.事务性：关系型数据库强调 ACID 规则（原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）），可以满足对事务性要求较高或者需要进行复杂数据查询的数据操作，而且可以充分满足数据库操作的高性能和操作稳定性的要求。并且关系型数据库十分强调数据的强一致性，对于事务的操作有很好的支持。关系型数据库可以控制事务原子性细粒度，并且一旦操作有误或者有需要，可以马上回滚事务。

8.读写性能：关系型数据库十分强调数据的一致性，并为此降低读写性能付出了巨大的代价，虽然关系型数据库存储数据和处理数据的可靠性很不错，但一旦面对海量数据的处理的时候效率就会变得很差，特别是遇到高并发读写的时候性能就会下降的非常厉害。

9.授权方式：关系型数据库常见的有 Oracle，SQLServer，DB2，Mysql，除了 Mysql 大多数的关系型数据库如果要使用都需要支付一笔价格高昂的费用，即使是免费的 Mysql 性能也受到了诸多的限制。[5]

2. 介绍两种不同类型的 NoSQL 数据库系统的架构，特点和应用场景

2.1 Redis

REmote DIctionary Server(Redis) 是一个由 Salvatore Sanfilippo 写的 key-value 存储系统。

Redis 是一个开源的使用 ANSI C 语言编写、遵守 BSD 协议、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库，并提供多种语言的 API。

它通常被称为数据结构服务器，因为值（value）可以是 字符串(String), 哈希(Hash), 列表(list), 集合(sets) 和 有序集合(sorted sets)等类型。

2.1.1 Redis 的架构[6]

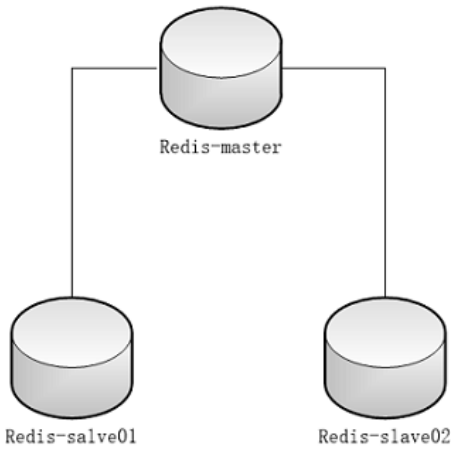
[5] 《NoSQL 数据库与关系型数据库对比》郎云海 CNKI:SUN:DTSJ.0.2019-05-195

[6] https://blog.csdn.net/sf131097/article/details/98589761

2.1.1.1 Redis 普通主从模式

通过持久化功能，Redis 保证了即使在服务器重启的情况下也不会损失（或少量损失）数据，因为持久化会把内存中数据保存到硬盘上，重启会从硬盘上加载数据。但是由于数据是存储在一台服务器上的，如果这台服务器出现硬盘故障等问题，也会导致数据丢失。为了避免单点故障，通常的做法是将数据库复制多个副本以部署在不同的服务器上，这样即使有一台服务器出现故障，其他服务器依然可以继续提供服务。为此，Redis 提供了复制（replication）功能，可以实现当一台数据库中的数据更新后，自动将更新的数据同步到其他数据库上。

在复制的概念中，数据库分为两类，一类是主数据库（master），另一类是从数据库（slave）。主数据库可以进行读写操作，当写操作导致数据变化时会自动将数据同步给从数据库。而从数据库一般是只读的，并接受主数据库同步过来的数据。一个主数据库可以拥有多个从数据库，而一个从数据库只能拥有一个主数据库。



Redis 的普通主从模式，能较好地避免单独故障问题，以及提出了读写分离，降低了 Master 节点的压力。互联网上大多数的对 redis 读写分离的教程，都是基于这一模式或架构下进行的。但实际上这一架构并非是目前最好的 redis 高可用架构。

2.1.1.2 Redis 哨兵模式高可用架构

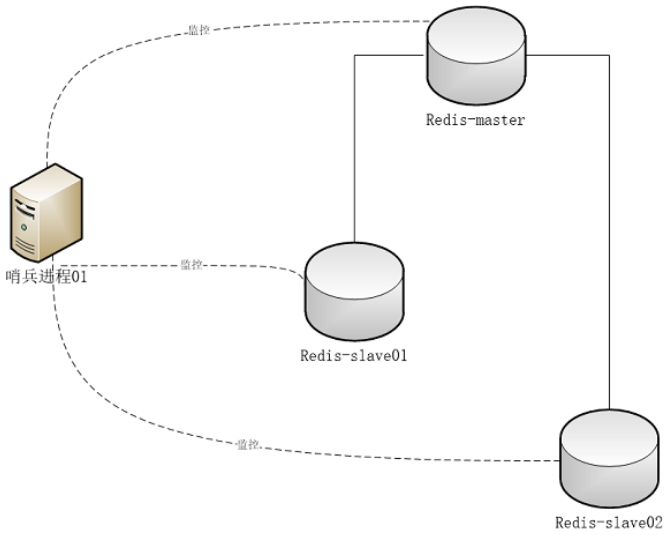
当主数据库遇到异常中断服务后，开发者可以通过手动的方式选择一个从数据库来升格为主数据库，以使得系统能够继续提供服务。然而整个过程相对麻烦且需要人工介入，难以实现自动化。

为此，Redis 2.8 开始提供了哨兵工具来实现自动化的系统监控和故障恢复功能。 哨兵的作用就是监控 redis 主、从数据库是否正常运行，主出现故障自动将从数据库转换为主数据库。

顾名思义，哨兵的作用就是监控 Redis 系统的运行状况。它的功能包括以下两个。

（1）监控主数据库和从数据库是否正常运行。

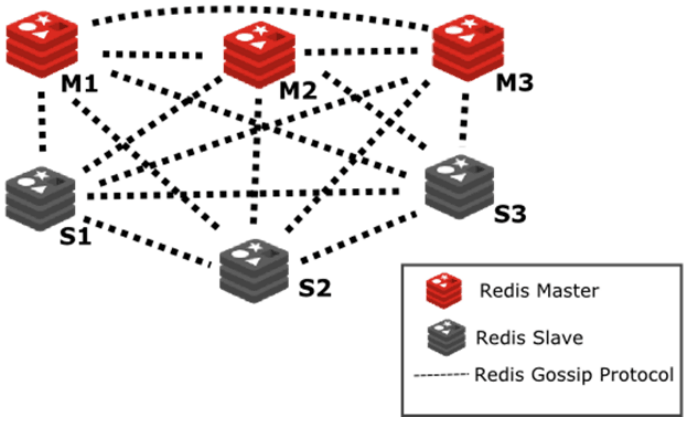
（2）主数据库出现故障时自动将从数据库转换为主数据库。



2.1.1.3 Redis-cluster 群集高可用架构

即使使用哨兵，redis 每个实例也是全量存储，每个 redis 存储的内容都是完整的数据，浪费内存且有木桶效应。为了最大化利用内存，可以采用 cluster 群集，就是分布式存储。即每台 redis 存储不同的内容。

采用 redis-cluster 架构正是满足这种分布式存储要求的集群的一种体现。redis-cluster 架构中，被设计成共有 16384 个 hash slot。每个 master 分得一部分 slot，其算法为：hash_slot = crc16(key) mod 16384 ，这就找到对应 slot。采用 hash slot 的算法，实际上是解决了 redis-cluster 架构下，有多个 master 节点的时候，数据如何分布到这些节点上去。key 是可用 key，如果有{}则取{}内的作为可用 key，否则整个可以是可用 key。群集至少需要 3 主 3 从，且每个实例使用不同的配置文件。



2.1.2 Redis 的特点

(1) Key-value 型数据库
(2) 支持内存存储，速度快，查找和操作的时间复杂度都是 0(1)
(3) 支持丰富的数据类型，支持 string，list，set，sorted set，hash 等
(4) 简单，使用 C 编写
(5) 内存消耗较大，不能用作海量数据的高性能读写
(6) 支持持久性操作
(7) 功能丰富，可用于缓存，消息，按 key 设置过期时间，过期后将会自动删除

2.1.3 Redis 的应用场景

所有数据 in memory 的场景，作为 Mem cached 的替代者使用的场景，需要除了 key/value 外的更多数据类型支持的场景，不适用于需要大容量数据集的应用、可能占用大量 CPU 或磁盘 IO 的场合，如新浪微博，Instagram 等

2.2 MongDB

MongDB 是一个强大、灵活而可扩展的数据存储系统，其将强大的可扩展特性与关系库最有用的特性进行了整合，像：次级索引，范围查询和排序等特性。而 MongDB 也内建了类似 MapReduce 汇聚和地理空间索引等有用特性。

历经艰苦的努力，MongDB 也拥有了易操作和用户友好等特性，同时，其还具备开发友好的模型、管理友好的配置选项及用起来感觉轻松自然的 API 和数据库 shell。MongDB 尽力让使用者省心，从而可以专注于编程而不用担心数据存储方面的问题。

2.2.1 MongDB 架构

1）单库：如果单个 MongDB 库能承担所有数据和负载，且不考虑数据冗余和高可用，单节点问题，研发和测试环境。则使用单库架构

2）主从复制：主从复制架构是 MongDB 支持的最常见的复制环境。这种架构非常灵活，且能被用于备份、故障切换、读扩展及更多其他用途，这种架构至少需要两台服务器配置成一主一备，也可以用多台服务器配置成一主多备，当然，也可以通过同一服务器上的多个实例配置成一主一备或一主多备，结果就是，同一服务器上的多个实例实现的这种架构，无论在其灵活性、备份、高可用还是读扩展等方面，其作用都会大打折扣，生产环境中多个硬件服务器配置成的主备库架构更常见。

3）复制集：复制集就是一个可以自动故障切换的主备库集群。其和前述的主备库架构之间的最大差别就是复制集并没有一个固定的主库，而是由集群选出一个主库，当前的主库宕掉后，另外一个服务器上的备库会转变为主库。另外，两种架构都总是有一个主库和一个或多个从库。复制集的好处是一切都是自动的，集群自己会自己完成很多管理任务，将从库提升为主库并确保数据的一致性。对开发者来说，这种架构的易用性也很好，只需确定集群中的服务器，驱动程序将自动发现所有的服务器，如果当前服务器宕掉后会自动完成故障切换。

4）分片：所谓分片，就是将数据拆分并将拆分后的不同部分数据存储到不同的服务器，这样，在没有升级为更强大硬件服务器的情况下，才可能存储更多数据和处理更多负载。分片分为手工分片和自动分片。

i）手工分片：几乎针对任何数据库系统，都可以进行手工分片。其中，每个应用都保持到几个不同数据库服务器的连接，这些数据库服务器是相互完全独立的，应用代码负责管理将不同的数据存储到不同的服务器上，以及查询相应的服务器以获取到正确的结果数据。手工分片会运行的很好，但服务器的增减以及数据的分布和负载平衡模式发生改变时，相应工作人工维护起来会越来越困难。

ii）自动分片：MongDB 支持自动分片，这消除了人工分片中管理维护工作的困难，MongDB 负责维护数据的自动拆分和再平衡，同时，开发者也不用关心数据分布和负载均衡问题。MongDB 自动分片时，会将其集合分散成小的数据块，这些数据块分布到不同的分片（服务器），其中的每个分片负责总数据集合中的一部分子集。通过 mongos 路由进程，MongDB 实现了数据分片对用户应用程序的透明，应用程序无需关心什么分片存储什么数据，甚至数据被分散到多个分片服务器上。所有数据分片工作，都是通过所有分片前端的 mongos 路由进程来完成。应用的工作只是知道 mongos 进程的连接方式，并像连接 MongDB

一样成功连接它即可。

2.2.2 MongDB 特点[7]

1）丰富的数据模型

MongDB 是一个面向文档的数据库，而不是关系数据库。其之所以放弃关系模型的原因是为了易于扩展，当然，还有其他的优势方面的考虑。MongDB 的理念是用一个更灵活的模型——“文档”，去替代关系库中行的概念。通过文档中嵌入文档和数组，其可以实现通过一条记录来表示复杂的层级关系，这点非常适合使用现代面向对象语言的开发者考虑数据的方式。

MongDB 中没有模式的概念，一个文档的键无需通过任何方式预先定义或固定。通常，没有模式发生改变，就无需进行大量数据的迁移。新增或删除键可以在应用层面进行处理，而无需强制他们都适应固定且同样的格式。这在处理数据模型演化时给予了开发者许多灵活性。

2）易扩展性

现代应用的数据规模正以难以置信的速度增长。高级传感技术、可用带宽的增长以及互联网上手持设备的流行，这都将创建一个场景，其中哪怕小规模的应用都将产生前所未有的大量数据，TB 级的数据，是之前闻所未闻的数据量，而现在却随处可见。

开发者数据量增长的同时，导致其将会面临一个困难的决定，他们将如何扩展他们的数据库？如果扩展数据库，将会面临纵向扩展（服务器硬件升级的更为强大）或横向扩展（将数据分布到多台服务器上）。纵向扩展通常是最容易的方法，但它也带来诸多缺陷，比如：更强大的服务器通常会很昂贵，当买不起更强大的服务器时，这种方法将到达一个物理上的硬性限制。对于人们最想建的大型 web 应用来说，通过一台服务器来实现不太可能，其成本也是难以承受的。

而横向扩展则兼具扩展性和经济性，为了增加存储空间和提升性能，你可以另外购买一台服务器，并将其加入你目前的集群。MongDB 一开始就设计为横向扩展，其面向文档的数据模型允许它将数据在多个服务器间自动拆分，数据和负载能在集群内进行平衡，文档可以自动重新分布，这使得开发者专注应用开发，而非扩展数据库。当需要更多容量时，只需将一台新服务器加入现有集群，并让数据库自己想办法完成后续的一切。

3）维持高性能

卓越性能是 MongDB 的主要目标，并且，这也已经导致了有关它的很多设计决定。MongDB 用二进制线性协议作为和服务器交互的主模式，其动态填补文档并预先分配数据文件以额外空间换取稳定的性能。MongDB 默认存储引擎中使用内存映射文件，并将内存管理的任务交给操作系统。此外，其还具有动态查询优化器的特点，这使得它能记住执行查询的最快方式。简而言之 MongDB 几乎每个方面都是为维持高性能而设计。

虽然，MongDB 足够强大且尽力试图保留关系库的很多特性，但其并非被设计来完成关系库的所有事情。任何时候，MongDB 都尽量将处理和逻辑交给客户端的驱动或代码，保持这种流线设计是 MongDB 能获得如此高性能的原因之一。

4）易于管理

MongDB 通过服务器自我管理来尽可能的简化数据库的管理。除了启动数据库服务器，很少需要人工管理。如果主库宕掉了，MongDB 能自动切换到备库并将这个备库提升为主库。分布式环境中，集群只需要被告知一个新节点的加入，然后，MongDB 将自动对其进行集成和配置。

5）索引

---

[7] https://www.cnblogs.com/lhdz_bj/p/11543161.html

MongDB 支持通常的次级索引，允许各种快速查询，且支持唯一、复合及地理空间索引等。

6）存储 java 脚本

开发者能在服务器端存储和使用 java 函数，以替代存储过程功能。

7）汇聚

MongDB 支持 MapReduce 和其他汇聚工具。

8）固定大小集合

封顶集合大小固定，且其适用于类似日志的某些数据类型。

9）文件存储

MongDB 支持存储大文件和文件元数据的易于使用的协议。

此外，MongDB 并不支持关系库的某些常见特点，特别是连接（join）和复杂的多行事务。这些也是允许强大扩展特性的架构所决定的，因为分布式系统中难以支持关系库的那些特点。

2.2.3 MongDB 的应用场景

需要动态查询支持的场景，需要使用索引而不是 MapReduce 的场景，对大数据库有性能要求的场景，需要平滑扩展的场景；不适用于高度事务性的系统场景和传统的商业智能应用；如大众点评等