大规模分布式系统第四次实验报告
员司雨 17307110448

PartI NoSQL 数据库与关系数据库的差异和各自特点

1. NoSQL 数据库与关系数据库差异

（1） 存储方式

NoSQL 数据库	关系数据库
NoSQL 数据库中的数据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。	表格式存储。数据存储在表的行和列中，因此数据之间很容易关联协作存储，提取数据很方便。

（2） 存储结构

NoSQL 数据库	关系数据库
Nosql 数据库基于动态结构，使用非结构化数据。因为 Nosql 数据库是动态结构，可以很容易适应数据类型和结构的变化。	关系数据库存储的是结构化数据。数据表都预先定义了结构（列的定义），结构描述了数据的形式和内容。这一点对数据建模至关重要，虽然预定义结构带来了可靠性和稳定性，但是修改这些数据比较困难。

（3） 存储规范

NoSQL 数据库	关系数据库
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。	在关系数据库当中，数据存储为了实现更高的规范性，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

（4） 存储扩展

NoSQL 数据库	关系数据库

Nosql 数据库是横向扩展的,它的存储天然就是分布式的，可以通过给资源池添加更多的普通数据库服务器来分担负载。	关系数据库是纵向扩展，也就是说想要提高处理能力，要使用速度更快的计算机。因为数据存储在关系表中，操作的性能瓶颈可能涉及到多个表，需要通过提升计算机性能来克服。虽然有很大的扩展空间，但是最终会达到纵向扩展的上限

（5） 查询方式

NoSQL 数据库	关系数据库
以块为单元操作数据，使用的是非结构化查询语言（UnQl），它是没有标准的。	结构化查询语言来操作数据库（就是我们通常说的 SQL）

（6） 事务

NoSQL 数据库	关系数据库
遵循 BASE 原则(基本可用(Basically Availble)、软/柔性事务（Soft-state ）、最终一致性（Eventual Consistency)) Nosql 数据库是在 CAP（一致性、可用性、分区容忍度）中任选两项，因为基于节点的分布式系统中，很难全部满足，所以对事务的支持不是很好，虽然也可以使用事务，但是并不是 Nosql 的闪光点。	遵循 ACID 规则(原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)、持久性(Durability))支持对事务原子性细粒度控制，并且易于回滚事务。

（7） 性能

NoSQL 数据库	关系数据库
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。	关系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。

（8） 授权方式

NoSQL 数据库	关系数据库
Nosql 数据库通常都是开源的,成本较小。	多数的关系型数据库都是付费的并且价格昂贵，成本较大。

2. NoSQL 数据库特点

特点	1. 易扩展。NoSQL 数据库种类繁多，但是一个共同的特点都是去掉关系数据库的关系型特性。数据之间无关系，这样就非常容易扩展。无形之间，在架构的层面上带来了可扩展的能力。 2. 大数据量，高性能。NoSQL 数据库都具有非常高的读写性能，尤其在大数据量下，同样表现优秀。这得益于它的无关系性，数据库的结构简单。一般 MySQL 使用 Query Cache。NoSQL 的 Cache 是记录级的，是一种细粒度的 Cache，所以 NoSQL 在这个层面上来说性能就要高很多。 3. 灵活的数据模型。NoSQL 无须事先为要存储的数据建立字段，随时可以存储自定义的数据格式。而在关系数据库里，增删字段是一件非常麻烦的事情。如果是非常大数据量的表，增加字段简直就是——个噩梦。这点在大数据量的 Web 2.0 时代尤其明显。 4. 高可用。NoSQL 在不太影响性能的情况，就可以方便地实现高可用的架构。比如 Cassandra、HBase 模型，通过复制模型也能实现高可用。
优势	1. 海量数据下，读写性能优异 2. 数据模型灵活 3. 数据间无关系，易于扩展
劣势	1. 不支持 sql 这样的工业标准查询，所以学习成本就比较高； 2. 大多都是初创产品，不够成熟，和传统数据库几十年的完善不可同日而语； 3. 大多数 nosql 都不支持事务（redis 支持，MongoDB 不支持）； 4. nosql 只能保证数据相对一致性，尤其是在数据同步的时候，主从服务器的状态是不一致的。

3. 关系数据库特点

特点	1.数据集中控制。在文件管理方法中，文件是分散的，每个用户或每种处理都有各自的文件，这些文件之间一般是没有联系的，因此，不能按照统一的方法来控制、维护和管理。而数据库则很好地克服了这一缺点，可以集中控制、维护和管理有关数据。 2.数据独立。数据库中的数据独立于应用程序，包括数据的物理独立性和逻辑独立性，给数据库的使用、调整、优化和进一步扩充提供了方便，提高了数据库应用系统的稳定性。 3.数据共享。数据库中的数据可以供多个用户使用，每个用户只与库中的一部分数据发生联系；用户数据可以重叠，用户可以同时存取数据而互不影响，大大提高了数据库的使用效率。 4.减少数据冗余。数据库中的数据不是面向应用，而是面向系统。数据统一定义、组织和存储，集中管理，避免了不必要的数据冗余，也提高了数据的一致性。 5.数据结构化。整个数据库按一定的结构形式构成，数据在记录内部和记录类型之间相互关联，用户可通过不同的路径存取数据。
优势	1、容易理解：二维表结构是非常贴近逻辑世界一个概念，关系模型相对网状、层次等其他模型来说更容易理解 2、使用方便：通用的 SQL 语言使得操作关系型数据库非常方便； 3、易于维护：丰富的完整性(实体完整性、参照完整性和用户定义的完整性)大大减低了数据冗余和数据不一致的概率； 4、支持 SQL，可用于复杂的查询。
劣势	1.为了维护一致性所付出的巨大代价就是其读写性能比较差 2.当为有数据更新的表做索引或对表结构进行变更时，性能差 3.难以实现对简单查询需要快速返回结果的处理；难以满足高并发读写需求； 4.当字段不固定时，关系型数据库处理起来很麻烦。 5.无法满足海量数据的管理需求。 6.无法满足高可扩展性和高可用性的需求

PartII 两种不同类型的NoSQL数据库系统

一. Redis数据库系统

（1） 简介

redis(REmote DIctionary Server)是一个由 Salvatore Sanfilippo 写key-value存储系统，它由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类型的数据库，并提供多种语言的API。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）。这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis 支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步，redis在3.0版本推出集群模式。

Redis 的外围由一个键、值映射的字典构成。与其他非关系型数据库主要不同在于：Redis中值的类型不仅限于字符串，还支持如下抽象数据类型：

1. 字符串列表
2. 无序不重复的字符串集合
3. 有序不重复的字符串集合
4. 键、值都为字符串的哈希表

值的类型决定了值本身支持的操作。Redis支持不同无序、有序的列表，无序、有序的集合间的交集、并集等高级服务器端原子操作。

redis提供五种数据类型：string，hash，list，set及zset(sorted set)。

1. string（字符串）

string是最简单的类型，你可以理解成与Memcached一模一样的类型，一个key对应一个value，其上支持的操作与Memcached的操作类似。但它的功能更丰富。redis采用结构sdshdr和sds封装了字符串，字符串相关的操作实现在源文件sds.h/sds.c中。

2. list(双向链表)

list是一个链表结构，主要功能是push、pop、获取一个范围的所有值等等。操作中key理解为链表的名字。对list的定义和实现在源文件adlist.h/adlist.c.

3. dict(hash表)

set 是集合，和我们数学中的集合概念相似，对集合的操作有添加删除元素，有对多个集合求交并差等操作。操作中key理解为集合的名字。在源文件dict.h/dict.c中实现了hashtable的操作.

4. set

set 是在 dict 的基础上实现的，指定了 key 的比较函数为dictEncObjKeyCompare，若key相等则不再插入。

5. zset(排序set)

zset是set的一个升级版本，他在set的基础上增加了一个顺序属性，这一属性在添加修改元素的时候可以指定，每次指定后，zset会自动重新按新的

值调整顺序。可以理解了有两列的 mysql 表，一列存 value，一列存顺序。操作中 key 理解为 zset 的名字。

（2） 组织架构

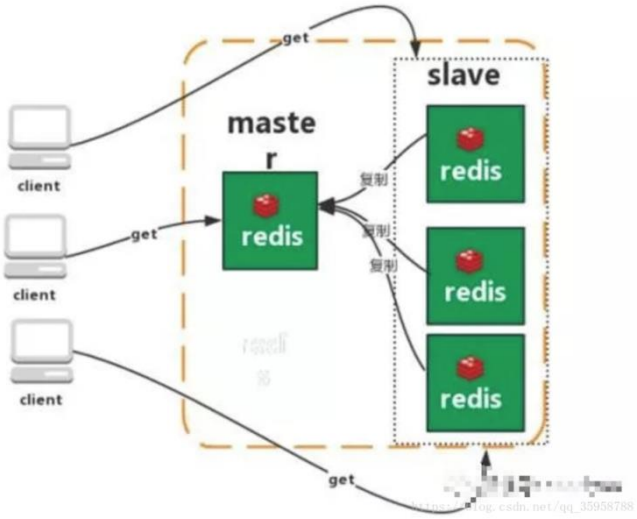
Redis 集群方式共有三种：主从模式，哨兵模式，cluster(集群)模式

1. 主从模式

主从模式主要是基于 Redis 的主从复制特性架构的。通常我们会设置一个主节点,N 个从节点。默认情况下,主节点负责处理使用者的 I/O 操作，而从节点则会对主节点的数据进行备份，并且也会对外提供读操作的处理。组从模式主要的特点如下：

- 主从模式下，当某一节点损坏时，因为其会将数据备份到其它 Redis 实例上，这样做在很大程度上可以恢复丢失的数据。
- 主从模式下，可以保证负载均衡。
- 主从模式下,主节点和从节点是读写分离的。使用者不仅可以从主节点上读取数据,还可以很方便的从从节点上读取到数据,这在一定程度上缓解了主机的压力。

从以上，我们不难看出 Redis 在主从模式下，必须保证主节点不会宕机——一旦主节点宕机，其它节点不会竞争称为主节点，此时，Redis 将丧失写的能力。这点在生产环境中，是致命的。



主从模式示意图

2. 哨兵模式

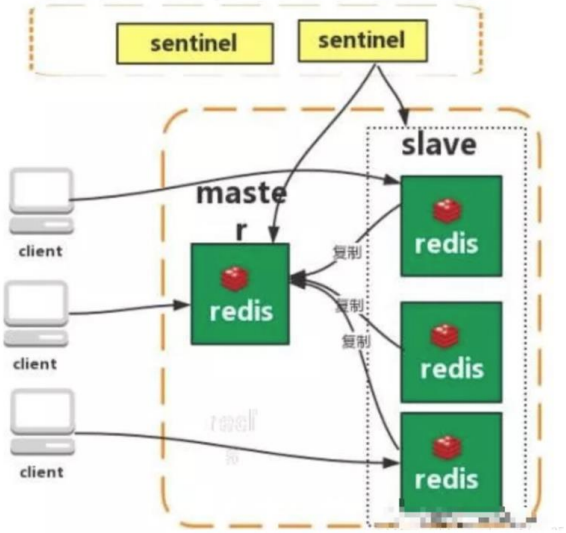
哨兵模式是基于主从模式做的一定变化，它能够为 Redis 提供了高可用性。在实际生产中，服务器难免不会遇到一些突发状况：服务器宕机，停电，硬件损坏等。这些情况一旦发生，其后果往往是不可估量的。而哨兵模式在一定程度上能够帮我们规避掉这些意外导致的灾难性后果。其实，哨兵模式的核心还是主从复制。只不过相对于主从模式在主节点宕机导致不可写的情况下，多了一个竞选机制——从所有的从节点竞选出新的主节点。竞选机制的实现，是依赖于在系统中启动一个 sentinel 进程。sentinel 特点：

- 监控：它会监听主服务器和从服务器之间是否在正常工作。
- 通知：它能够通过 API 告诉系统管理员或者程序，集群中某个实

例出了问题。

- 故障转移：它在主节点出了问题的情况下，会在所有的从节点中竞选出一个节点，并将其作为新的主节点。
- 提供主服务器地址：它还能够向使用者提供当前主节点的地址。这在故障转移后，使用者不用做任何修改就可以知道当前主节点地址。

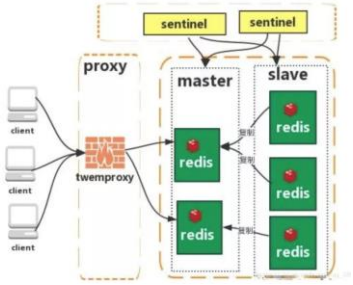
sentinel 也可以集群部署多个哨兵，sentinel 可以通过发布与订阅来自动发现 Redis 集群上的其它 sentinel。sentinel 在发现其它 sentinel 进程后，会将其放入一个列表中，这个列表存储了所有已被发现的 sentinel。集群中的所有 sentinel 不会并发着去对同一个主节点进行故障转移。故障转移只会从第一个 sentinel 开始，当第一个故障转移失败后，才会尝试下一个。当选择一个从节点作为新的主节点后，故障转移即成功了(而不会等到所有的从节点配置了新的主节点后)。这过程中，如果重启了旧的主节点，那么就会出现无主节点的情况，这种情况下，只能重启集群。当竞选出新的主节点后，被选为新的主节点的从节点的配置信息会被 sentinel 改写为旧的主节点的配置信息。完成改写后，再将新主节点的配置广播给所有的从节点。



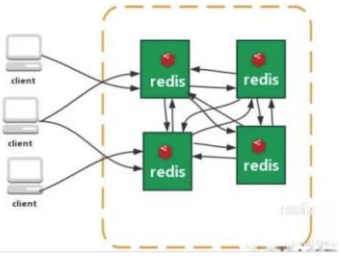
哨兵模式示意图

3. 集群模式

使用集群，只需要将每个数据库节点的 cluster-enable 配置打开即可。每个集群中至少需要三个主数据库才能正常运行。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，可以采用集群，就是分布式存储。即每台 redis 存储不同的内容。集群至少需要 3 主 3 从，且每个实例使用不同的配置文件，主从不用配置，集群会自己选。



集群模式（proxy）



集群模式（直连型）

（3） 特点

1、Redis 不仅仅只支持简单的 K-V 形式的数据存储，还支持 list、set、hash、zset 等等集合类数据的存储；

2、Redis 支持实时的数据备份，及时宕机，也可以把数据恢复过来；

3、Redis 支持数据的持久化，可以存放在内存 memory 中的数据直接保存在磁盘上；

（4） 应用场景

1. 查找最新的回复。

   如果在传统的关系型数据库，这就需要使用 select * from table where name="" order by time desc limit 100；这十分消耗数据库性能，但是通过 Redis，就可以直接在 Redis 里面通过 Id 创建一个 List，指定长度 1w，当需要查找时，直接输出该 list 的后 100 条记录。

2. 排行问题

   常见的排行问题，例如最热话题、游戏排名等等，这些都可以通过 Redis 来轻松实现，直接使用 ZRank 即可得到。

3. 删除过期数据

   Redis 不是真正意义上的可持久化数据库，可以给数据加上一个有效时间，在有效时间超过时，Redis 会自动删除对应数据。

二．MongoDB 数据库系统

（1） 简介

MongoDB 是一个基于分布式文件存储的数据库。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。

MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。它支持的数据结构非常松散，是类似 json 的 bson 格式，因此可以存储比较复杂的数据类型。Mongo 最大的特点是它支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

（2） 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，单节点没问题，研发和测试环境，这个架构比较常见，也比较容易部署，这里就不再多说了。

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

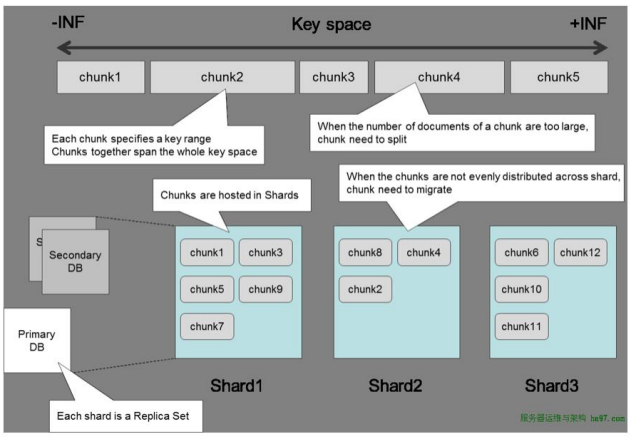
需要两台服务器配置成一主一备，也可以用多台服务器配置成一主多备，当然，也可以通过同一服务器上的多个实例配置成一主一备或一主多备，结果就是，同一服务器上的多个实例实现的这种架构，无论在其灵活性、备份、高可用还是读扩展等方面，其作用都会大打折扣，生产环境中还是多个硬件服务器配置成的主备库架构更常见。

3. 复制集：复制集基本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 一样成功连接它就可以了。



分片机制

（3） 特点

1. **易于使用**

与关系型数据库相比，面向文档的数据库不再有“行”（row）的概念，取而代之的是更为灵活的“文档”（document）模型。通过在文档中嵌入文档和数组，面向文档的方法能够仅使用一条记录来表现复杂的层次关系，这与使用现代面向对象语言的开发者对数据的看法一致。另外，不再有预定义模式（predefined schema）：文档的键（key）和值（value）不再是固定的类型和大小。由于没有固定的模式，根据需要添加或删除字段变得更容易了。通常，由于开发者能够进行快速迭代，所以开发进程得以加快。而且，实验更容易进行。开发者能尝试大量的数据模型，从中选择一个最好的。

2. **易于扩展**

MongoDB 的设计采用横向扩展。面向文档的数据模型使它能很容易地在多台服务器之间进行数据分割。MongoDB 能自动处理跨集群的数据和负载，自动重新分配文档，以及将用户请求路由到正确的机器上。这样，开发者能够集中精力编写应用程序，而不需要考虑如何扩展的问题。如果一个集群需要更大的容量，只需要向集群添加新服务器，MongoDB 就会自动将现有数据向新服务器传送。

3. **丰富的功能**

MongoDB 作为一款通用型数据库，除了能够创建、读取、更新和删除数据之外，还提供一系列不断扩展的独特功能。

- 索引（indexing）

MongoDB 支持通用二级索引，允许多种快速查询，且提供唯一索引、复合索引、地理空间索引，以及全文索引。

- 聚合（aggregation）

MongoDB 支持“聚合管道”（aggregation pipeline）。用户能通过简单的片段创建复杂的聚合，并通过数据库自动优化。

- 特殊的集合类型

MongoDB 支持存在时间有限的集合，适用于那些将在某个时刻过期的数据，如会话（session）。类似地，MongoDB 也支持固定大小的集合，用于保存近期数据，如日志。

- 文件存储（file storage）

MongoDB 支持一种非常易用的协议，用于存储大文件和文件元数据。

4. **卓越的性能**

MongoDB 的一个主要目标是提供卓越的性能，这很大程度上决定了 MongoDB 的设计。MongoDB 能对文档进行动态填充（dynamic padding），也能预分配数据文件以利用额外的空间来换取稳定的性能。MongoDB 把尽可能多的内存用作缓存（cache），试图为每次查询自动选择正确的索引。

（4） 应用场景

MongoDB 的主要目标是在键/值存储方式（提供了高性能和高度伸缩性）和传统的 RDBMS 系统（具有丰富的功能）之间架起一座桥梁，它集两者的优势于一身。根据官方网站的描述，Mongo 适用于以下场景。

1. 网站数据：Mongo 非常适合实时的插入，更新与查询，并具备网站实时数据存储所需的复制及高度伸缩性。

2. 缓存：由于性能很高，Mongo 也适合作为信息基础设施的缓存层。在系统重启之后，由 Mongo 搭建的持久化缓存层可以避免下层的数据源过载。

3. 大尺寸、低价值的数据：使用传统的关系型数据库存储一些数据时可能会比较昂贵，在此之前，很多时候程序员往往会选择传统的文件进行存储。

4. 高伸缩性的场景：Mongo 非常适合由数十或数百台服务器组成的数据库，Mongo 的路线图中已经包含对 MapReduce 引擎的内置支持。

5. 用于对象及 JSON 数据的存储：Mongo 的 BSON 数据格式非常适合文档化格式的存储及查询。

参考文献

[1]罗琼主编；杨微副主编；卢青华，张莉娜，袁丽娜，陈孝如参编. 计算机科学导论：北京邮电大学出版社，2016.08：第 167 页

[2]杨旭，汤海京，丁刚毅主编. 数据科学导论：北京理工大学出版社，2014.03：第 129 页

[3]超高性能 key-value 数据库 Redis.开源社区网[引用日期 2012-09-08]

[4] redis 各个版本发布的新特性. it 技术博客. 2019-08-17[引用日期 2019-08-21]

[5] Brad D .The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing[J]. springer ebooks, 2010.

[6] Banker K . MongoDB in Action[J]. pearson schweiz ag, 2011.