NoSQL 非关系型数据库学习报告

徐鸣飞、吴珂非

2023 年 6 月 10 日

摘要

本文总体介绍了我们小组在学习 NoSQL 数据库时所，并就 MongoDB 进行了更深一层的学习。此外列举了国内外较为重要的 NoSQL 数据库实现。

关键词：NoSQL 数据库，高并发读写，高效率存储和访问，高扩展性和可用性，基于 Key 值的数据模型，MongoDB，文档型数据库，CouchDB，OrientDB，Terrastore，时序数据库。

1 NoSQL 的起源

NoSQL 数据库的发展最早可以追溯到 1991 年 Berkeley DB 第一版的发布。Berkeley DB 是一个 Key/Value (键/值) 类型的 Hush 数据库。这种类型的数据库适用于数据类型相对简单, 但需要极高的插入和读取速度的嵌入式场合。[1]

NoSQL 一词最早出现于 1998 年，是 Carlo Strozzi 开发的一个轻量、开源、不提供 SQL 功能的关系数据库。2009 年，Last.fm 的 Johan Oskarsson 发起了一次关于分布式开源数据库的讨论，来自 Rackspace 的 Eric Evans 再次提出了 NoSQL 的概念，这时的 NoSQL 主要指非关系型、分布式、不提供 ACID 的数据库设计模式。2009 年在亚特兰大举行的"no:sql(east)"讨论会是一个里程碑，其口号是"select fun, profit from real_world where relational=false;"。因此，对 NoSQL 最普遍的解释是" 非关联型的"，强调 Key-Value Stores 和文档数据库的优点，而不是单纯的反对 RDBMS。

2 NoSQL 数据库的发展原因

自上世纪 90 年代以来，互联网应用经历了用户通过互联网获取信息的 Web1.0 时代，现在已发展到更注重用户交互作用的 Web 2.0 时代。期间互联网应用经历了用户从至多 2000 人到现在百万级别以上的用户群。应用内容从严谨的业务流程，例如航班预订、股票交易，发展到如今的通信、购物、娱乐、社交等各领域川。数据量也从以前的 TB 级升至 PB 级，并仍在持续爆炸式增长，互联网应用进入大数据时代。

在这期间，计算机网络和硬件水平得到了飞速的发展。针对互联网应用的转变，应用架构也从集中处理、向上扩展 (Scale-Up) 的交互式系统架构，衍变为现代 Web 应用的分布式、横向扩展 (Scale-Out) 的系统架构，它可以通过添加更多的 Web 服务器来支持更多. 的用户。为应对大数据时代海量数据的分析处理，出现了基于大规模廉价计算平台的云计算和分布式大数据集处理模型 MapReducel。

与快速转换的应用架构和数据处理技术相比，关系型数据库技术在这几

十年间的发展缓慢。面对日新月异的互联网应用，传统关系型数据库面临以下几个难题[2]:

1. 高并发读写

   Web2.0 时代的互联网需要根据用户的个性化信息实时生成动态页面，提供动态信息, 因此数据库的并发负载非常高，可以达到每秒上万次的读写请求问。

2. 海量数据的高效率存储和访问

   像人人网、新浪微博这样的社交网站，每天用户产生海量的用户动态。以新浪微博为例，其公开的社交网络数据有 2.6 亿条，对于关系数据库来说，在一个 2.6 亿条记录的表里进行 SQL 查询效率极低 [7]。再例如像盛大网络、腾讯这样的大型网站，其用户登录系统有上亿的账号，关系型数据库的管理查询性能很低 [8]。

3. 高扩展性和可用性

   关系型数据库技术是为前期集中化计算模型设计的技术。为适应更多用户与负载，它采用向上扩展的方式，即采用更大型的服务器，升级 CPU、内存和硬盘 I/O, 这样导致硬件成本直线上升。关系型数据库很难像 Web 服务器那样简单的通过添加更多的硬件和服务器节点来扩展性能和负载能力 7。数据库的升级和扩展需要停机维护和数据迁移，这对于很多需要提供 24 小时不间断服务的网站来说降低了可用性。

关系型数据库很难满足以上三方面的互联网需求，而对于 Web 2.0 时代的互联网应用和大数据时代的海量数据管理，关系型数据库的很多主要特性往往并不适用:

1. 事务一致性

   很多互联网实时系统并不要求严格的数据库事务，对读一致性要求很低，有些场合对写一致性要求也不高。关系型数据库严格的事务管理反而成为数据库高负载下的一个沉重的负担。

2. 读写实时性

   关系型数据库保证在插入一条数据后查询可以立刻读出这条数据。但很多互联网应用并不要求这么高的实时性。例如新浪微博中用户发布了一条微博，过十几秒甚至几分钟后才被关注者看到是完全可以接受的。

3. 复杂 SQL 查询关系型数据库中的复杂 SQL 查询，尤其是多表关联查询，需要耗费较长的时间。大数据量的 Web 系统在设计阶段就会避免这样的情况，更多的是单表的主键查询以及单表的简单条件分页查询，并不需要复杂的 SQL 功能。

关系型数据库在当前越来越多的互联网应用下不再适合, 而 NoSQL 就是在解决这样的应用需求下产生的一种非关系型数据库技术的总称，自其概念提出以来发展迅速，已出现十多种流行的数据库产品，广泛使用在互联网应用中。

3 NoSQL 技术简介

3.1 NoSQL 概念与特性

NoSQL 非关系型数据库技术现在还没有一个公认的权威定义。InfoSys Technologies 的首席技术架构师 SouravMazumder 提出了“非关系数据库”的一个较为严谨的描述[3]:

(1) 使用可扩展的松耦合类型数据模式来对数据进行逻辑建模;

(2) 为遵循 CAP 定理的跨多节点数据分布模型而设计，支持水平伸缩;

(3) 拥有在磁盘和 (或) 内存中的数据持久化能力;

(4) 支持多种“Non-SQL” 接口来进行数据访问。

从这个描述中可以看出，NoSQL 相对于传统的关系型数据库在两个方面做出了重大变革。一是数据模式，NoSQL 使用松耦合类型、可扩展的数据模式，例如 key-value 键值对、列、文档、图表等等，而不是关系型数据库的固定的二维表元组。NoSQL 的数据模式没有严格的定义，不要求在存

储数据前就确定数据模式，在系统运行中也可以动态更改。这非常有利于存储现代 Web 应用中占绝大部分的半结构化和非结构化数据。二是水平伸缩，NoSQL 本质. 上就是为分布式系统设计的，支持横向扩展，能够适应现代 Web 应用飞速增长的海量数据，并且在分布式架构下可以达到很好的性能。传统关系型数据库通常都支持 ACID 的强事务机制。而与之不同的是，NoSQL 系统通常注重性能和扩展性，而非事务机制。对很多 NoSQL 系统来说，对性能的考虑远在 ACID 的保证之上。通常 NoSQL 系统仅提供对行级别的原子性保证，即同时对同一个 Key 下的数据进行的两个操作，在实际执行中是串行执行的，保证了每一个 Key-Value 对不会被破坏。对绝大多数应用场景来说，这样的保证并不会引起多大的问题，但其换来的执行效率却是非常可观的。当然使用这样的系统可能需要在应用层的设计上多做容错性和修正机制的考虑。

NoSQL 中通常有两个层次的一致性: 第一种是强一致性，即集群中的所有机器状态同步保持一致; 第二种是最终一致性，即可以允许短暂的数据不一致，但数据最终会保持致。NoSQL 会削弱数据一致性的原因是 CAP 理论。这个理论最早被 EricBrewer 教授提出, 后由 SethGilbert 和 NancyLynch 进行了证明。CAP 理论首先把分布式系统中的三个特性进行了如下归纳:

- 一致性 (Consistency): 系统中的所有数据备份，在同一时刻都是同一值。
- 可用性 (Availability): 每个操作总能在确定的时间内返回，即系统随时都是可用的。
- 分区容错性 (Tolerance to network Partitions): 在出现网络分区的情况下，例如断网，分离的系统也能正常运行。

CAP 理论指出，在分布式存储系统中最多只能同时满足以上两个特性[4]。

3.2 基于 Key 值的数据模型

数据库的数据模型指的是数据在数据库中的组织方式，数据库的操作模型指的是存取这些数据的方式。通常数据模型包括关系模型、键值模型以及

各种图结构模型。数据操作语言可能包括 SQL、键值查询及 MapReduce 等。NoSQL 数据库之间的架构通常结合了多种数据模型和操作模型，互不相同，这里仅介绍基于 Key 值的数据模型。

在键值型系统中，键值查找的优势是: 对数据库的操作模式是固定的，这些操作所产生的负载也是相对固定且可预知的。这样分析整个应用的性能瓶颈就变得简单，因为复杂的逻辑操作并没有放在数据库里面封装操作。但另一方面，复杂的联合操作以及满足多个条件的取数据操作并不容易实现，因此会造成业务逻辑和数据逻辑不容易分清。当前 NoSQL 的数据模型大致包括 Key-Value 存储、Key-结构化存储、Key-文档存储等以下几种不同键值模型的数据结构。

1. Key-Value 存储

   Key-Value 存储是最简单的 NoSQL 存储，每个 key 值对应一个任意的数据值。NoSQL 系统并不关心这个任意的数据值是什么。单纯的 Key-Value 存储不提供针对数据中特定的某个属性值的操作，通常它只提供像 set、get 和 delete 这样的简单操作[5]。以 Dynamo 为原型的 Voldemort 数据库，就只提供了分布式的 Key-Value 存储功能。Berkeley DB 则是一个提供 Key-Value 操作的持久化数据存储引擎。

2. Key-结构化数据存储

   Key-结构化数据存储的典型代表是 Redis, 它将 Key-Value 存储中的 Value 扩展为结构化的数据类型，包括数字、字符串、列表、集合以及有序集合。除了 set/get/delete 操作以外，Redis 还提供了很多针对以上数据类型的特殊操作，比如针对数字可以执行增减操作，对 list 可以执行 push/pop 操作，而这些对特定数据类型的特定操作并没有对性能造成多大的影响[6]。通过提供这种针对单个 Value 的特定类型的操作，Redis 可以说实现了功能与性能的平衡。

3. Key-文档存储

   Key-文档存储的代表有 CouchDB、MongoDB 和 Riak。这种存储方式下 Key-Value 的 Value 是结构化文档，通常这些文档是被转换成

JSON 或类似 JSON 的结构进行存储。文档可以存储列表、键值对以及层次结构复杂的文档。文档型存储的灵活性和复杂性是一把双刃剑: 一方面，开发者可以任意组织文档结构，另一方面，应用层的查询需求会变得比较复杂。

4. BigTable 的列簇式存储

HBase 和 Cassandra 的数据模型都借鉴自 Google 的 BigTable。这种数据模型的特点是列式存储，即每一行数据的各项被存储到不同的列中，这些列的集合称为列簇。而每一列中每一个数据都包含一个时间戳属性, 这样列中的同一个数据项的多个版本都能保存下来。

列式存储可以这样理解，将行 ID、列簇号、列号以及时间戳一起组成一个 Key，然后将 Value 按 Key 的顺序进行存储[6]。这种数据结构可以天然地进行高效的松散列数据存储, 即在很多行中并没有某列的值。当然另一个方面，对那些很少有某一列是 Null 值的行，由于每一个数据必须包含列标识，会造成空间浪费。这些 NoSQL 系统对 BigTable 数据模型的实现互有一些差别，其中以 Cassandra 的变更最为明显。Cassandra 引入了 supercolumn 的概念，通过将多个列组织到相应的 supercolumn 中，可以在更高层级上进行数据的组织、索引等。

4 MongoDB 简介

4.1 MongoDB 特性

1. 文档数据模型

MongoDB 以二进制 JSON 格式存储文档数据，或者叫做 BSON。BSON 有相似的数据结构，它是专门为文档存储设计。当查询 MongoDB 并返回结果时，这些数据就会转换为易于阅读的数据格式。

关系型数据库包含表，MongoDB 拥有集合。即关系型数据库在表的行里保存数据，而 MongoDB 在集合的文档里保存数据。集合是 MongoDB 中非常重要的概念，集合中的数据存储在磁盘上，而且大部分查

询需要指定查询的目标集合。面向文档的数据模型非常适合表示集中形式的数据，允许我们处理整个数据，而这些数据只包含在一个数据库对象中。

MongoDB 在支持丰富的数据结构外，文档不需要遵守严格的数据定义 schema。我们在关系型数据库存储数据时，每个表都有严格的 schema，指定每个列的数据类型。如果某行需要扩展字段，就必须修改表结构。MongoDB 的文档放在集合中，集合不需要定义 schema，理论上，每个集合中的文档都可以拥有不同的数据结构，实际上，集合中的文档都是相对一致的。

2. 查询

系统支持主动查询模式（ad hoc queries）是指不需要事先定义系统接收何种查询，关系型数据库忠实地执行格式正确的包含各种条件的 SQL 查询，而键值存储的查询只支持一个领域的查询：键 Key，键-值存储数据库牺牲丰富的查询功能来换取更简单的伸缩模型。MongoDB 的设计目标之一是保留大部分关系型数据库的功能。

3. 索引

ad hoc 查询的一个关键元素就是查找在创建数据库时还不知道的值，随着数据库中添加的文档数据越来越多，查询的成本变得越来越高。

MongoDB 的索引是使用 B-树（平衡树）数据结构。B-树索引也大量使用于许多关系型数据库中，对于不同的查询做了优化，包括范围扫描和条件子句查询。MongoDB 能为单个的文档建立索引，这种叫做辅助索引，通过允许多个辅助索引，MongoDB 可以允许用户优化不同的查询，这也是 MongoDB 重要的功能特性。

4. 复制

MongoDB 提供了数据库复制特性，叫做可复制集合（replica set）。可复制集合在多个机器上分布式存储数据，在服务器或者网络出错时，实现数据冗余存储和自动灾备。此外，复制还可用于伸缩数据库读操作。

可复制集合由多台服务器组成集群。通常，每个服务器有独立的物理机。我们调用这些节点，一个节点作为主节点，其他的作为次节点。与其他数据库中的主从复制类似，可复制集合的主节点可以同时接受读/写操作，但是从服务器只能进行读操作。

可复制集合最重要的一个特点是支持自动化灾备，如果主节点失败，则集群中会选择一个从节点，并自动提升为主节点。当之前的主节点回归时，会作为从节点。

5. 加速与持久化

在数据库系统领域，写入速度和持久性之间存在矛盾的关系，写入速度（write speed）可以理解为数据库在给定时间内插入、更新和删除的容量。持久性（durability）指的是这些写操作被永久保存的保证级别。

从 2.0 版本开始，MongoDB 默认启动事务日志，默认 100 毫秒就会写一次日志文件，如果服务器意外关机，日志会通过重启服务器来确保 MongoDB 数据文件恢复为一致状态。

对于写入压力，可以通过关闭日志功能以提高性能，坏处是意外关机之后可能导致数据文件冲突。

6. 伸缩

伸缩数据库最简单的方式就是升级服务器硬件。如果应用是运行在单个节点上，则通常可行的方案就是通过组合添加更快的磁盘、更多内存以及更强的 CPU 来解决数据库性能瓶颈。提升单节点参数的做法通常也称垂直扩展（vertical scaling 或 scaling up）。垂直扩展非常简单、可靠，但是达到某个点后成本很高，最终会达到一个无法低成本垂直扩展的临界点。

然后可以考虑水平扩展（horizontally 或 scaling out），水平扩展指的是在多台机器上分布式存储数据库，而不是提升单个节点的配置。水平扩展架构可以运行在许多台很小的、廉价的机器上，通常可以减少

硬件的成本。而且，跨机器分布式存储数据可以降低宕机带来的丢失数据的后果。

MongoDB 通过基于范围的分区机制来实现水平扩展，称为分片机制，它可以自动化管理每个分布式节点存储的数据。

分片系统处理额外的分片节点，而且它还会处理自动化灾备。每个独立的节点是一个可复制集合，至少由 2 台机器组成，确保节点失败的时候可以自动恢复。

4.2 MongoDB 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）以及传统的 RDBMS 系统（丰富的功能）架起一座桥梁，集两者的优势于一身。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：使用传统的关系数据库存储一些数据时可能会比较贵，在此之前，很多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 非常适合由数十或者数百台服务器组成的数据库。
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格式化的存储及查询。

使用行业：

- 游戏场景，使用 MongoDB 存储游戏用户信息，用户的装备、积分等直接以内嵌文档的形式存储，方便查询、更新

- 物流场景，使用 MongoDB 存储订单信息，订单状态在运送过程中会不断更新，以 MongoDB 内嵌数组的形式来存储，一次查询就能将订单所有的变更读取出来。
- 社交场景，使用 MongoDB 存储存储用户信息，以及用户发表的朋友圈信息，通过地理位置索引实现附近的人、地点等功能
- 物联网场景，使用 MongoDB 存储所有接入的智能设备信息，以及设备汇报的日志信息，并对这些信息进行多维度的分析
- 视频直播，使用 MongoDB 存储用户信息、礼物信息等

公司应用案例：

- Craiglist 上使用 MongoDB 的存档数十亿条记录。
- FourSquare，基于位置的社交网站，在 Amazon EC2 的服务器上使用 MongoDB 分享数据。
- Shutterfly，以互联网为基础的社会和个人出版服务，使用 MongoDB 的各种持久性数据存储的要求。
- bit.ly, 一个基于 Web 的网址缩短服务，使用 MongoDB 的存储自己的数据。
- spike.com，一个 MTV 网络的联营公司，spike.com 使用 MongoDB 的。
- Intuit 公司，一个为小企业和个人的软件和服务提供商，为小型企业使用 MongoDB 的跟踪用户的数据。
- sourceforge.net，资源网站查找，创建和发布开源软件免费，使用 MongoDB 的后端存储。
- etsy.com ，一个购买和出售手工制作物品网站，使用 MongoDB。
- 纽约时报，领先的在线新闻门户网站之一，使用 MongoDB。

- CERN，著名的粒子物理研究所，欧洲核子研究中心大型强子对撞机的数据使用 MongoDB。

不适合的场景：

- 高度事物性的系统：例如银行或会计系统。传统的关系型数据库目前还是更适用于需要大量原子性复杂事务的应用程序。
- 传统的商业智能应用：针对特定问题的 BI 数据库会对产生高度优化的查询方式。对于此类应用，数据仓库可能是更合适的选择。
- 需要 SQL 的问题。

4.3 MongoDB 学习心得

mongodb 是一个非关系型的文档数据库。数据库中有集合这一概念，集合由一组文档构成，这一个集合相当于 RDBMS 中的一张表。而一个集合中的文档是动态的键值对，统一集合下的文档不必具有相同的文档字段以及结构，不同文件中的相同字段数据类型可以不同。

MongoDB 是一种 NoSQL 的数据库，是面向对象的数据库。与一般的 RDBMS 不同，MongoDB 没有表与表之间的关系的概念，MongoDB 的操作语法非常适用于 JavaScript 的语法糖。BSON 的格式可以看做是 JSON 的扩展，因此 JavaScript 环境非常适合作为运行使用 MongoDB 的理想的宿主环境。

MongoDB 的数据模型：相比于 RDBMS 中的数据，MongoDB 的数据模型。比如说在 RDBMS 中要建立多对多的关系。比如老师，课程这一个一对多的关系。采用 RDBMS 数据库会设计两张表，一张老师信息表，一张课程表。但是在 MongoDB 中只需要一个集合然后集合中文档采用如下数据模型实现。

```
{
        '_id':ObjectId(),
        'name':'alex',
        'course':
```

```
    [
        {
            'name':'english',
            'describ':'blabla'
        },
        {
            'name':'math',
            'describ':'blabla'
        }
    ]
}
```

可以将一对多的关系采用一个数据结构的文档表示，非常简单。集合是没有固定的模式的。

创建数据库的语法：use DB_NAME

这个语法有两个作用：如果数据库本身不存在那么使用这个指令会创建一个数据库，如果数据库存在那么使用这个指令会切换到选择的数据库。

当然要查询当前数据库的名字，使用 db 指令，如果要看当前系统中所有的数据库那么使用 show dbs 查看，但是在创建数据库后，如果并未往数据库中添加数据，使用 show dbs 是查询不到数据库的，必须往数据库中写入至少一个文件才行。所以需要创建集合 (collections)，插入数据。一般情况下创建了 collection 就可以使用 show dbs 查询到数据库了。

MongoDB 的数据类型:分析 MongoDB 的数据类型,可以看到与 JavaScript 的数据类型有联系也有较大的区别。MongoDB 的文档存储格式类型为 BSON (Binary JSON)。是对 json 的扩展 (一般 json 只是支持 6 中数据类型 null, number, string, boolean, array, object)，加入了更多的在数据库中大量使用的数据类型。MongoDB 支持的整数有 32 位以及 64 位的有符号整数类型。由于 Mongo 的 shell 是 JavaScript 的 shell。因此处理数字的时候会出问题。js 的 shell 只有一种数字类型。因此在用 shell 进行数据存储时，可能造成更改原有数据类型的情况。

经过这一段时间的学习，我发现 mongoDB 的难点不在具体实现上，其基本使用是非常简单的，正如 java 操作 mongodb 文档和文件上传下载中介绍的，只是对几个封装类进行一些操作而已，再加上一份 api 文档的帮助就足够了。我认为难点在于理解 mongoDB 的一些特性，体会其存储形式的好处，以及确定应用场景这几方面。

mongoDB(非结构化数据库) 不仅可以处理结构化数据，而且更适合处理非结构化数据 (文本、图像、超媒体等信息)。它突破了关系型数据库结构定义不易改变而且数据定长的限制，在处理连续信息和非结构化信息中有着关系型数据库无法比拟的优势。

上面说的是 mongoDB 存储形式上的优势，紧接着就是这种存储形式给它带来的性能上的优势：由于数据结构松散，数据之间没有 join 操作，因此可以将数据写到多台服务器上，也就是自动分片技术，分片增强了写扩展性，以应对面对大数据量的写入。另外，mongoDB 的复制技术还提供了数据备份、故障转移，同时也减轻了数据读取压力。

开始接触时，觉得好多陌生的词：模式自由、复制、高可用性、自动分片、信息基础设施等等。但是按照网上的教程不断地学习如何使用 mongodb，越能感到其优点，而这些优点是与上述名词有着密不可分地关系的，于是对这些名词就有了更深刻地了解，当我初步学会使用后，我又自行搜索论文以及相关文章进一步地理解，这些陌生的名词也就不陌生了。

5 其他常见的 NoSQL 数据库实现

5.1 文档型数据库

文档型数据库主要用来存储、索引并管理面向文档的数据或者类似的半结构化数据。文档型数据库（面向文档数据库）的关键核心概念即文档（Document），它是数据库中最小的单位。每一种文档型数据库的存储放肆都有所不同，一般都假定文档以某种标准化格式封装并加密数据，并用多种格式进行解码，包括 XML、YAML、JSON 和 BSON，当然也包括二进制格式如 PDF、微软 Office 文档等。文档型数据库存储并检索文档数据，用

户在选用的时候应考虑数据访问的模式和用例，以便创建一个高效实用的文档模型。

5.1.1 CouchDB

CouchDB 是一个面向文档的、分布式的数据库，支持 REST 接口访问。面向文档是指 CouchDB 中存储的是半结构化的 JSON 文档，而且可以方便的将文档对象映射为具体编程语言的对象。由于 JSON 在 Ajax 技术及社交网络应用中广泛应用，因此 CouchDB 在这类应用的数据存储和处理中具有良好的应用前景。CouchDB 是分布式的数据库系统，源于 Erlang 极好的并发特性，CouchDB 存储系统可以分布到多台计算机之上，每台计算机称为存储系统的一个节点。CouchDB 能够很好的协调和同步多个节点之间的数据一致性和完整性，有效的应用系统应用中可能出现的各种错误。CouchDB 支持 REST 接口访问，即可以通过 GET、PUT、POST、HEAD 和 DELETE 等标准的 Http 请求对数据库进行写入和查询分析等操作。因此，任何一种编程语言只要具备 Http 请求模块就可以方便的与 CouchDB 进行对接，甚至只要利用正确的 URL 地址只要运用浏览器就可以访问 CouchDB 得到所需要的数据。

与其他的非关系数据库一样，CouchDB 利用 Map/Reduce 对数据进行插入、搜索等操作。CouchDB 将键值对存储在 B-树引擎之上，并根据键值进行排序，因此具有高效的查询和操作性能。需要注意的是，CouchDB 是一个基于版本的数据库系统，即只能添加数据不能删除和修改数据，当数据需要更新时 CouchDB 只是增加了新的版本，所有的版本数据依旧保存在数据库中，且可以方便的得到。如果要删除数据则需要将整个数据库删除。[8]

5.1.2 OrientDB

OrientDB 是一个基于 Java 实现的多模型数据库，它将键值模型、文档模型、图模型和对象模型融合到一个数据库引擎中，同时支持无模式、全模式和混合模式。OrientDB 提供社区版和企业版两个版本，其中社区版是开源的，可以免费使用（Apache 2 许可）。OrientDB 中存储的最小单位是记录

（Record），记录可以存储在四种类型中：文档、二进制大对象（Binary Large Object，BLOB）、顶点和边。在 OrientDB 中，使用类（Class）定义记录。OrientDB 中类的概念最接近关系数据库中表的概念。类可以是无模式、全模式或混合模式的。它们可以从其他类继承，继承表示子类继承父类，并继承其所有属性。每个类都有自己的簇（Cluster），簇是存储一组记录的地方。一个类可以支持多个簇，当对一个类执行查询时，它会自动传播到属于该类的所有簇，当然，也可以从单个簇中查询特定对象。类提供了用于组织数据的逻辑框架，簇提供了实际的存储数据的物理或内存空间。当 OrientDB 生成记录时，会自动为每一个记录自动分配一个唯一的标识符，称为记录 ID（Record ID，RID）。RID 包含了簇的信息和位置的信息，格式如下：

$\#<cluster>:<position>$

其中，cluster 指示了记录所属的簇，position 标识符指示了记录在簇中的绝对位置。通过 RID 可以直接访问记录，不需要像在关系型数据库中那样创建一个字段作为主键。

OrientDB 支持使用 Gremlin 以及扩展用于图遍历的 SQL 进行查询，作为多模型数据库，OrientDB 可以非常方便、灵活地完成跨不同数据模型的查询，这是多模型数据库 OrientDB 的优势之一。如果不使用多模型数据库，就需要编写一个应用层程序分别调用关系型数据库、键值数据库、文档数据库、图数据库等数据库接口，通过跨数据库查询的方式实现与多模型数据库相同的功能，这样查询就会比较复杂，性能也会受到影响。

5.1.3 Terrastore

Terrastore 是一个基于 Terracotta（一个业界公认的、快速的分布式集群组件）实现的高性能分布式文档数据库。可以动态从运行中的集群添加 / 删除节点，而且不需要停机和修改任何配置。支持通过 http 协议访问 Terrastore。Terrastore 提供了一个基于集合的键 / 值接口来管理 JSON 文档并且不需要预先定义 JSON 文档的架构。

5.2 时序数据库

时序数据库全称为时间序列数据库。时间序列数据库指主要用于处理带时间标签（按照时间的顺序变化，即时间序列化）的数据，带时间标签的数据也称为时间序列数据。

时间序列数据主要由电力行业、化工行业、气象行业、地理信息等各类型实时监测、检查与分析设备所采集、产生的数据，这些工业数据的典型特点是：产生频率快（每一个监测点一秒钟内可产生多条数据）、严重依赖于采集时间（每一条数据均要求对应唯一的时间）、测点多信息量大（常规的实时监测系统均有成千上万的监测点，监测点每秒钟都产生数据，每天产生几十 GB 的数据量）。

5.2.1 Apache Cassandra

Cassandra 是一套开源分布式 NoSQL 数据库系统。它最初由 Facebook 开发，用于储存收件箱等简单格式数据，集 GoogleBigTable 的数据模型与 Amazon Dynamo 的完全分布式的架构于一身 Facebook 于 2008 将 Cassandra 开源，此后，由于 Cassandra 良好的可扩展性，被 Digg、Twitter 等知名 Web 2.0 网站所采纳，成为了一种流行的分布式结构化数据存储方案。

Cassandra 是一个混合型的非关系的数据库,类似于 Google 的 BigTable。其主要功能比 Dynamo（分布式的 Key-Value 存储系统）更丰富，但支持度却不如文档存储 MongoDB（介于关系数据库和非关系数据库之间的开源产品，是非关系数据库当中功能最丰富，最像关系数据库的。支持的数据结构非常松散，是类似 json 的 bjson 格式，因此可以存储比较复杂的数据类型）。Cassandra 最初由 Facebook 开发，后转变成了开源项目。它是一个网络社交云计算方面理想的数据库。以 Amazon 专有的完全分布式的 Dynamo 为基础，结合了 Google BigTable 基于列族（Column Family）的数据模型。

5.3 其他开源数据库

5.3.1 Hypertable

Hypertable 是一个开源、高性能、可伸缩的数据库，它采用与 Google 的 Bigtable 相似的模型。在过去数年中，Google 为在 PC 集群上运行的可伸缩计算基础设施设计建造了三个关键部分。第一个关键的基础设施是 Google File System（GFS），这是一个高可用的文件系统，提供了一个全局的命名空间。它通过跨机器（和跨机架）的文件数据复制来达到高可用性，并因此免受传统文件存储系统无法避免的许多失败的影响，比如电源、内存和网络端口等失败。第二个基础设施是名为 Map-Reduce 的计算框架，它与 GFS 紧密协作，帮助处理收集到的海量数据。第三个基础设施是 Bigtable，它是传统数据库的替代。Bigtable 让你可以通过一些主键来组织海量数据，并实现高效的查询。Hypertable 是 Bigtable 的一个开源实现，并且进行了一些改进。

5.3.2 Redis

redis 是一个 key-value 存储系统。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 是一个高性能的 key-value 数据库。redis 的出现，很大程度补偿了 memcached 这类 key/value 存储的不足，在部分场合可以对关系数据库起到很好的补充作用。它提供了 Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang 等客户端。

Redis 支持主从同步。数据可以从主服务器向任意数量的从服务器上同

步，从服务器可以是关联其他从服务器的主服务器。这使得 Redis 可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。

5.3.3 HBase

HBase –Hadoop Database，是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，利用 HBase 技术可在廉价 PC Server 上搭建起大规模结构化存储集群。

与 FUJITSU Cliq 等商用大数据产品不同，HBase 是 Google Bigtable 的开源实现，类似 Google Bigtable 利用 GFS 作为其文件存储系统，HBase 利用 Hadoop HDFS 作为其文件存储系统；Google 运行 MapReduce 来处理 Bigtable 中的海量数据，HBase 同样利用 Hadoop MapReduce 来处理 HBase 中的海量数据；Google Bigtable 利用 Chubby 作为协同服务，HBase 利用 Zookeeper 作为对应。

6 总结

nosql 作为新时代解决传统数据库问题的解决方法，对数据模式进行了改变，采用了松耦合可拓展的数据模式，更加动态、自由，同时具有水平伸缩，为分布式拓展系统提供了工具，能够很好的处理现代互联网海量的数据。Nosql 系统具有良好的性能与拓展性，但是要在应用层设计上多做容错性和修正机制的考虑。

Nosql 分为 key-value 存储、结构化数据存储以及文档存储，不同的存储方式适用于不同的场景，而本次研究的 mongodb 属于文档存储类型，以二进制 JSON 格式存储文档数据，当查询 MongoDB 并返回结果时，这些数据就会转换为易于阅读的数据格式。在支持丰富的数据结构外，文档不需要遵守严格的数据定义 schema。理论上，每个集合中的文档都可以拥有不同的数据结构，实际上，集合中的文档都是相对一致的。Mongodb 功能丰

富，适用场景广泛，在游戏场景、物流、社交软件都具有应用，前景十分广泛，本次学习中我们学习了如何使用 mongodb，在对其有初步了解后通过搜索资料，阅读论文对 mongodb 的理解进一步加深，在本次学习之后我们会将学习的知识运用到实践中提升自己。

参考文献

[1] 黄贤立. *NoSQL* 非关系型数据库的发展及应用初探 [J]. 福建电脑,2010,26(07):30+45.

[2] Coulter T. Costing: non traditional data stores versus traditional DBMS technologies[C].Technology Management in the Energy Smart World (PICMET), 2011 IEEE Proceeding of PICMET'11: July 31-Aug 4.2011:1-15.

[3] 颜开. *NoSQL* 数据库笔谈 *v2*[EB/OL]. http://www.yankay.com/nosql/. 2012 年 2 月访问.

[4] Brewer E. Pushing the CAP: strategies for consistency and availability[J]. Computer, Feb.2012, Volume 45 :23-29.

[5] Sakr S, Liu A, Batista D, Alomari M. A survey of large scale data management approaches in cloud environments[C]. Communications Survey & Tutorials,IEEE, Volume 13,Issue3:1:3-336.

[6] Redis 官方网站 [EB/OL]. http://www.redis.io

[7] Fay C, Jeffery D, Sanjay G Wilson H, Deborah W, Michael B, Tushar C, Andrew F, Robert G.BigTable: a distributed storage system for structured data[J]. ACM Trans. Comput.Syst.2008(26).

[8] 刘臣. 非关系数据库 *CouchDB* 的应用 [J]. 电脑知识与技术,2013,9(14):3220-3222.