NoSQL

Not Only SQL-非关系型数据库

Outline

- NoSQL基本概念
- NoSQL数据库分类
- NoSQL在系统架构中的应用
- MongoDB 数据库

NoSQL是什么？

- NoSQL概念在2009年被提了出来。NoSQL最常见的解释是“non-relational”，“Not Only SQL”也被很多人接受。
- Wikipedia上定义
  - NoSQL是一种打破了关系型数据库长久以来占主导地位的快速成长起来的非关系松散数据存储类型，这种数据存储不需要事先设计好的表结构，它也不会出现表之间的连接操作和水平分割，学术界称这种数据库为结构化存储[1]。

NoSQL诞生的原因

- 关系型数据库面临的问题
  - 扩展困难：由于存在类似Join这样多表查询机制，使得数据库在扩展方面很艰难；
  - 读写慢：这种情况主要发生在数据量达到一定规模时由于关系型数据库的系统逻辑非常复杂，使得其非常容易发生死锁等的并发问题，所以导致其读写速度下滑非常严重；
  - 成本高：企业级数据库的License价格很惊人，并且随着系统的规模，而不断上升；
  - 有限的支撑容量：现有关系型解决方案还无法支撑Google这样海量的数据存储；
- 数据库访问的新需求
  - 低延迟的读写速度：应用快速地反应能极大地提升用户的满意度；
  - 支撑海量的数据和流量：对于搜索这样大型应用而言，需要利用PB级别的数据和能应对百万级的流量；
  - 大规模集群的管理：系统管理员希望分布式应用能更简单的部署和管理；
  - 庞大运营成本的考量：IT经理们希望在硬件成本、软件成本和人力成本能够有大幅度地降低；

NoSQL数据库的共有原则

- **假设失效是必然发生的**
  - NOSQL实现都建立在硬盘、机器和网络都会失效这些假设之上。
  - 我们不能彻底阻止这些时效，我们需要让我们的系统能够在即使非常极端的条件下也能应付这些失效。
- **对数据进行分区**
  - 最小化了失效带来的影响，也将读写操作的负载分布到了不同的机器上。
- **保存同一数据的多个副本**
  - 大部分 NOSQL 实现都基于数据副本的热备份来保证连续的高可用性。
  - 一些实现提供了 API，可以控制副本的复制，也就是说，当你存储一个对象的时候，你可以在对象级指定你希望保存的副本数。
- **查询支持**
  - 在这个方面，不同的实现有相当 本质的区别。不同实现的一个共性在于哈希表中的 key/value 匹配。

NoSQL数据库的共有原则

- 动态伸缩
  - 要掌控不断增 长的数据，大部分 NOSQL 实现提供了不停机或完全重新分区的扩展集群的方法。
    - 一个已知的处理这个问题的算法称为一致哈希。有很多种不同算法可以实现一致哈希。
    - 另一个（简单很多）的算法使用逻辑分区。在逻辑分区中，分区的数量是固定的，但分区在机器上的分布式动态的。
- 使用 Map/Reduce 处理汇聚
  - Map/Reduce 常常被看作是并行汇聚查询的一个模式。大部分 NOSQL 实现并不提供 map/reduce 的内建支持，需要一个外部的框架来处理这些查询。
  - 在质疑模型中，你可以将代码发送到数据所在地地方，并在该节点上直接运行复杂的查询
- 基于磁盘的和内存中的实现
  - NOSQL 实现分为基于文件的方法和内存中的方法。有些实现提供了混合模型，将内存和磁盘结合使用。两类方法的最主要区别在于每 GB 成本和读写性能。
  - 对于较低性能的实现，磁盘方案的成本远低于基于内存的方法，而对于高性能需求的场合，内存方案则更加廉价。

设计理念：关系数据库 VS NoSQL

- 关系数据库
  - 表都是存储一些格式化的数据结构，
  - 每个元组字段的组成都一样，
  - 即使不是每个元组都需要所有的字段，但数据库会为每个元组分配所有的字段，
  - 这样的结构可以便于表与表之间进行连接等操作。
- NoSQL
  - 以键值对存储，它的结构不固定，
  - 每一个元组可以有不一样的字段，
  - 每个元组可以根据需要增加一些自己的键值对，
  - 不会局限于固定的结构，可以减少一些时间和空间的开销。

设计理念：关系数据库 VS NoSQL

- 关系数据库
  - 分布式关系型数据库中强调的ACID分别是原子性（Atomicity）、一致性Consistency）、隔离性（Isolation）、持久性（Durability）。
  - ACID的目的就是通过事务支持，保证数据的完整性和正确性
- NoSQL
  - 对于许多互联网应用来说，对于一致性要求可以降低，而可用性(Availability)的要求则更为明显，从而产生了弱一致性的理论BASE。
  - BASE分别是英文：Basically，Available，Soft-state，Eventual Consistency的缩写，这个模型是反ACID模型。

NoSQL和关系数据库

- NoSQL数据库仅仅是关系数据库在某些方面（性能，扩展）的一个弥补
  - 单从功能上讲，NoSQL的几乎所有的功能，在关系数据库上都能够满足。
  - 一般会把NoSQL和关系数据库进行结合使用，各取所长，各得其所。
- 在某些应用场合，比如一些配置的关系键值映射存储、用户名和密码的存储、Session会话存储等等，
  - 用NoSQL完全可以替代关系数据库(如：MySQL)存储。不但具有更高的性能，而且开发也更加方便。

NoSQL的优缺点

- 优点
  - 简单的扩展

    典型例子是Cassandra，由于其架构是类似于经典的P2P，所以能通过轻松地添加新的节点来扩展这个集群；
  - 快速的读写

    主要例子有Redis，由于其逻辑简单，而且纯内存操作，使得其性能非常出色，单节点每秒可以处理超过10万次读写操作；
  - 低廉的成本

    这是大多数分布式数据库共有的特点，因为主要都是开源软件，没有昂贵的License成本；

- 不足
  - 不提供对SQL的支持

    如果不支持SQL这样的工业标准，将会对用户产生一定的学习和应用迁移成本；
  - 支持的特性不够丰富

    现有产品所提供的功能都比较有限，大多数NoSQL数据库都不支持事务，也不像MS SQL Server和Oracle那样能提供各种附加功能，比如BI和报表等；
  - 现有产品的不够成熟

    大多数产品都还处于初创期，和关系型数据库几十年的完善不可同日而语；

Outline

- NoSQL基本概念
- NoSQL数据库分类
- NoSQL在系统架构中的应用
- MongoDB 数据库

NoSQL 数据库分类

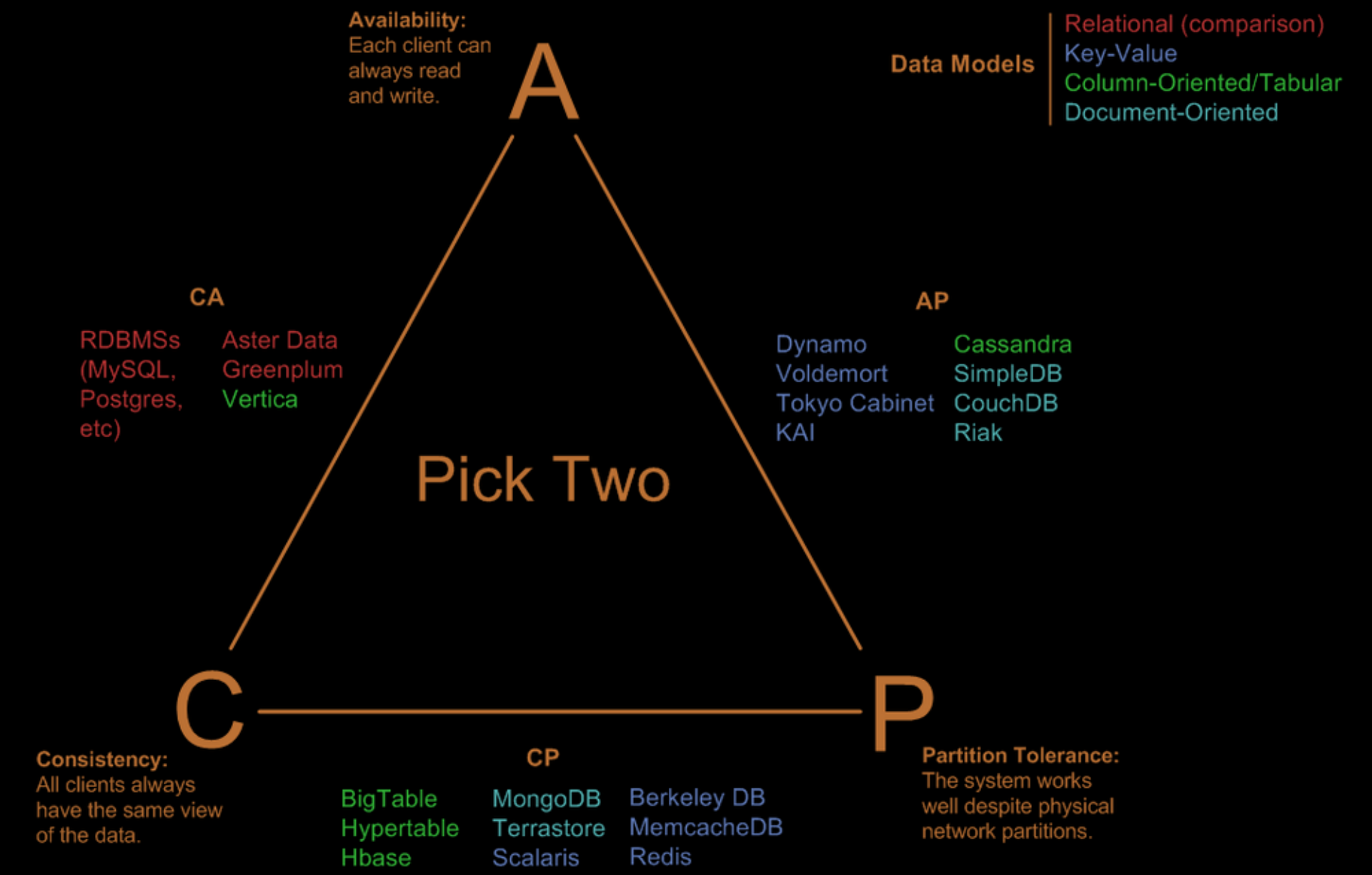
- 按CAP 分类
- 按数据模型分类

CAP理论

一个分布式系统不能同时满足一致性，可用性和分区容错性这三个需求，最多只能同时满足两个。

- 一致性（Consistency）
  - 任何一个读操作总是能读取到之前完成的写操作结果，也就是在分布式环境中，多点的数据是一致的；
- 可用性（Availability）
  - 每一个操作总是能够在确定的时间内返回，也就是系统随时都是可用的。
- 分区容忍性（Partition Tolerance）
  - 在出现网络分区（比如断网）的情况下，分离的系统也能正常运行。

Visual Guide to NoSQL Syst



关注一致性和可用性的 (CA)

- 这些数据库对于分区容忍性方面比较不感冒，主要采用复制（Replication）这种方式来保证数据的安全性，常见的CA系统有：
  - 传统关系型数据库，比如Postgres和MySQL等(Relational)；
  - Vertica (Column-oriented)；
  - Aster Data (Relational)；
  - Greenplum (Relational)；

关注一致性和分区容忍性的(CP)

- 这种系统将数据分布在多个网络分区的节点上，并保证这些数据的一致性，但是对于可用性的支持方面有问题，比如当集群出现问题的话，节点有可能因无法确保数据是一致性的而拒绝提供服务，主要的CP系统有：
  - BigTable (Column-oriented)；
  - Hypertable (Column-oriented)；
  - HBase (Column-oriented)；
  - MongoDB (Document)；
  - Terrastore (Document)；
  - Redis (Key-value)；
  - Scalaris (Key-value)；
  - MemcacheDB (Key-value)；
  - Berkeley DB (Key-value) ；

关于可用性和分区容忍性的(AP)

- 这类系统主要以实现"最终一致性（Eventual Consistency）"来确保可用性和分区容忍性，AP的系统有：
  - Dynamo (Key-value)；
  - Voldemort (Key-value)；
  - Tokyo Cabinet (Key-value)；
  - KAI (Key-value)；
  - Cassandra (Column-oriented)；
  - CouchDB (Document-oriented)；
  - SimpleDB (Document-oriented)；
  - Riak (Document-oriented)；

NoSQL-三种主流的数据模型

- **Column-oriented（列式）**
  - 主要围绕着“列（Column）”，而非“行（Row）”进行数据存储
  - 属于同一列的数据会尽可能地存储在硬盘同一个页（Page）中
  - 大多数列式数据库都支持Column Family这个特性
    （很多类似数据仓库（Data Warehouse）的应用，虽然每次查询都会处理很多数据，但是每次所涉及的列并没有很多）
  - **特点：**比较适合汇总（Aggregation）和数据仓库这类应用。
- **Key-value**
  - 类似常见的HashTable，一个Key对应一个Value，但是其能提供非常快的查询速度、大的数据存放量和高并发操作，
  - 非常适合通过主键对数据进行查询和修改等操作，
    虽然不支持复杂的操作，但可通过上层的开发来弥补这个缺陷。
- **Document（文档）**
  - 类似常见的HashTable，一个Key对应一个Value，
  - 其能提供非常快的查询速度、大的数据存放量和高并发操作，
  - 非常适合通过主键对数据进行查询和修改等操作，
  - 虽然不支持复杂的操作，但是可以通过上层的开发来弥补这个缺陷。

NoSQL按数据模型分类

类型	部分代表	特点
列存储	Hbase Cassandra Hypertable	顾名思义，是按列存储数据的。最大的特点是方便存储结构化和半结构化数据，方便做数据压缩，对针对某一列或者某几列的查询有非常大的IO优势。
文档存储	MongoDB CouchDB	文档存储一般用类似json的格式存储，存储的内容是文档型的。这样也就有有机会对某些字段建立索引，实现关系数据库的某些功能。
key-value存储	Tokyo Cabinet / Tyrant Berkeley DB MemcacheDB Redis	可以通过key快速查询到其value。一般来说，存储不管value的格式，照单全收。（Redis包含了其他功能）
图存储	Neo4J FlockDB	图形关系的最佳存储。使用传统关系数据库来解决的话性能低下，而且设计使用不方便。
对象存储	db4o Versant	通过类似面向对象语言的语法操作数据库，通过对象的方式存取数据。
xml数据库	Berkeley DB XML BaseX	高效的存储XML数据，并支持XML的内部查询语法，比如XQuery,Xpath。

以上NoSQL数据库类型的划分并不是绝对，只是从存储模型上来进行的大体划分，它们之间没有绝对的分界。

选择合适的NoSQL

我们常常需要根据如下情况考虑：

- 数据结构特点
  - 包括结构化、半结构化、字段是否可能变更、是否有大文本字段、数据字段是否可能变化。
- 写入特点
  - 包括insert比例、update比例、是否经常更新数据的某一个小字段、原子更新需求。
- 查询特点
  - 包括查询的条件、查询热点的范围。比如用户信息的查询，可能就是随机的，而新闻的查询就是按照时间，越新的越频繁。

Outline

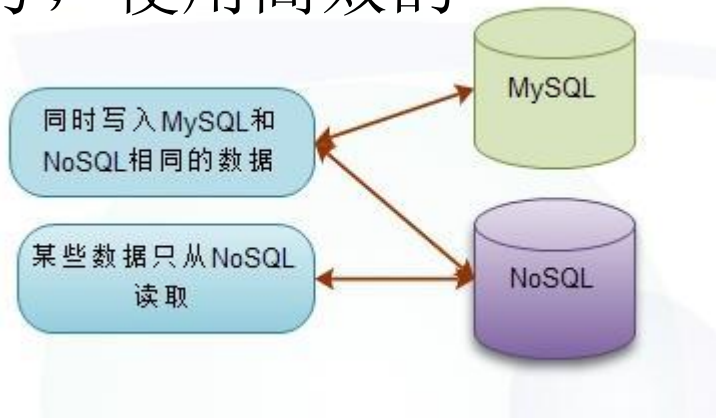
- NoSQL基本概念
- NoSQL 数据库分类
- NoSQL在系统架构中的应用
- MongoDB 数据库

NoSQL在系统架构中的应用

- 以NoSQL为辅
- 以NoSQL为主
- 以NoSQL为缓存

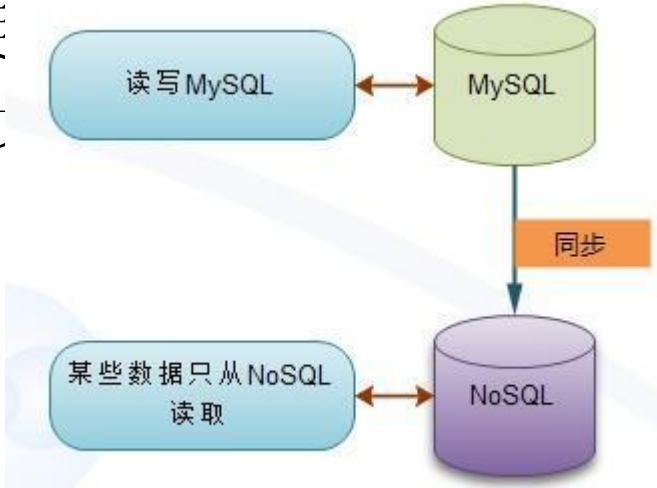
以NoSQL为辅

- NoSQL作为镜像
  - 不改变原有的以MySQL作为存储的架构，使用NoSQL作为辅助镜像存储，用NoSQL的优势辅助提升性能。
  - 在原有基于MySQL数据库的架构上增加了一层辅助的NoSQL存储，
  - 在写入MySQL数据库后，同时写入到NoSQL数据库，让MySQL和NoSQL拥有相同的镜像数据
  - 在某些可以根据主键查询的地方，使用高效的NoSQL数据库查询



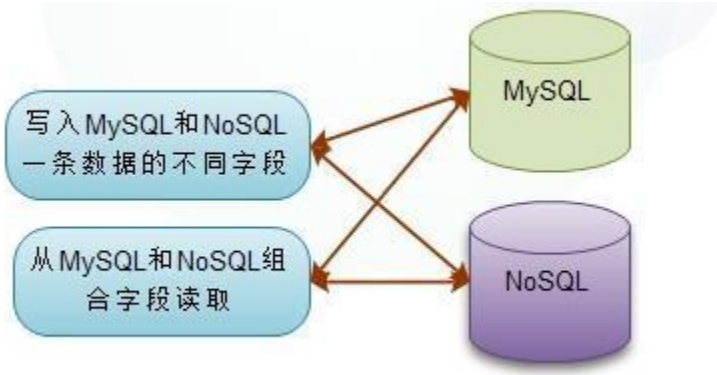
以NoSQL为辅

- **NoSQL为镜像（同步模式）**
  - 通过MySQL把数据同步到NoSQL中,，是一种对写入透明但是具有更高技术难度一种模式
  - 适用于现有的比较复杂的老系统，通过修改代码不易实现，可能引起 时也适用于需要把数据同步 存储中。



以NoSQL为辅

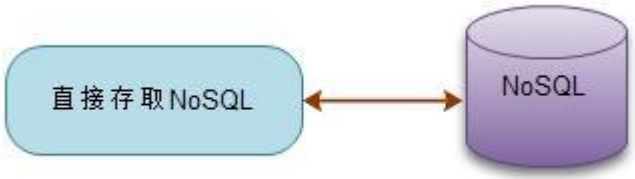
- MySQL和NoSQL组合
  - MySQL中只存储需要查询的小字段，NoSQL存储所有数据。
  - 把需要查询的字段，一般都是数字，时间等类型的小字段存储于MySQL中，根据查询建立相应的索引，
  - 其他不需要的字段，包括大文本字段都存储在NoSQL中。
  - 在查询的时候，我们先从MySQL中查询出数据的主键，然后从NoSQL中直接取出对应的数据即可。



以NoSQL为主

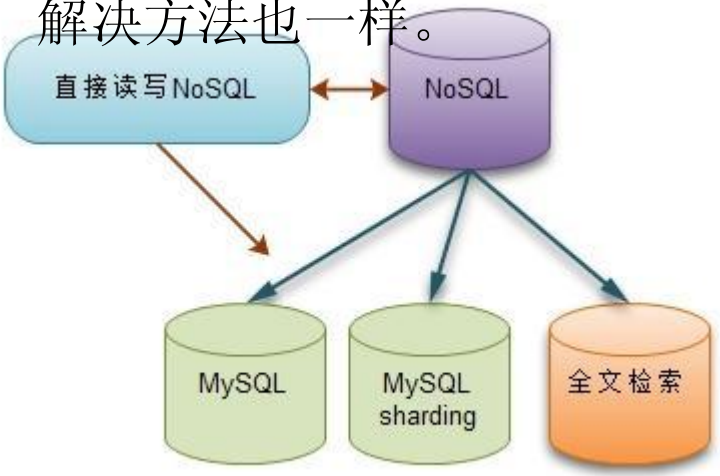
- **纯NoSQL架构**
  - 在一些数据结构、查询关系非常简单的系统中，我们可以只使用NoSQL即可以解决存储问题。
  - 在一些数据库结构经常变化，数据结构不定的系统中，就非常适合使用NoSQL来存储。
    - 比如监控系统中的监控信息的存储，可能每种类型的监控信息都不太一样。
  - 有些NoSQL数据库已经具有部分关系数据库的关系查询特性，他们的功能介于key-value和关系数据库之间，却具有key-value数据库的性能，基本能满足绝大部分web 2.0网站的查询需求。

MongoDB就带有关系查询的功能，能解决常用的关系查询，所以也是一种非常不错的选择。



以NoSQL为主

- **以NoSQL为数据源的架构**
  - 数据直接写入NoSQL，再通过NoSQL同步协议复制到其他存储。
  - 根据应用的逻辑来决定去相应的存储获取数据。
  - 应用程序只负责把数据直接写入到NoSQL数据库，然后通过NoSQL的复制协议，把NoSQL数据的每次写入，更新，删除操作都复制到MySQL数据库中。
  - 同 时，也可以通过复制协议把数据同步复制到全文检索实现强大的检索功能。
  - 这种架构需要考虑数据复制的延迟问题，这跟使用MySQL的master-salve模式的延迟问题是一样的，解决方法也一样。



以NoSQL为缓存

由于NoSQL数据库天生具有高性能、易扩展的特点，所以我们常常结合关系数据库，存储一些高性能的、海量的数据。
从另外一个角度看，根据NoSQL的高性能特点，它同样适合用于缓存数据。用NoSQL缓存数据可以分为内存模式和磁盘持久化模式。

- **内存模式**
  - Memcached提供了相当高的读写性能，在互联网发展过程中，一直是缓存服务器的首选。
  - NoSQL数据库Redis又为我们提供了功能更加强大的内存存储功能。跟Memcached比，Redis的一个key的可以存储多种数据结构Strings、Hashes、Lists、Sets、Sorted sets。
  - Redis不但功能强大，而且它的性能完全超越大名鼎鼎的Memcached。
    - Redis支持List、hashes等多种数据结构的功能，提供了更加易于使用的api和操作性能，比如对缓存的list数据的修改。

以NoSQL为缓存

- **持久化模式**
  - 虽然基于内存的缓存服务器具有高性能，低延迟的特点，但是内存成本高、内存数据易失却不容忽视。
  - 大部分互联网应用的特点都是数据访问有热点，也就是说，只有一部分数据是被频繁访问的。
  - 其实NoSQL数据库内部也是通过内存缓存来提高性能的，通过一些比较好的算法
    - 把热点数据进行内存cache，
    - 非热点数据存储到磁盘
    - 以节省内存占用。
  - 使用NoSQL来做缓存，由于其不受内存大小的限制，我们可以把一些不常访问、不怎么更新的数据也缓存起来。

Outline

- NoSQL基本概念
- NoSQL 数据库分类
- NoSQL在系统架构中的应用
- MongoDB 数据库

MongoDB

- MongoDB的名称取自“humongous”(巨大的) 的中间部分，足见mongodb的宗旨在处理大量数据上面
- 面向文档(Document)存储
- C++实现
- 支持多平台 Windows Linux, Mac OS-X, FreeBSD, Solaris
- 多语言驱动:
  - Ruby / Ruby-on-Rails
  - Java
  - C#
  - JavaScript
  - C / C++
  - Erlang Python, Perl

MongoDB设计理念

- 不求搞定一切
- 正确的工具做正确的事儿
- 性能与功能的取舍
- 放弃事务

MongoDB特性

- 模式自由
  - 支持动态查询、完全索引，可轻易查询文档中内嵌的对象及数组
- 面向文档存储
  - 易存储对象类型的数据, 包括文档内嵌对象及数组
- 高效的数据存储
  - 支持二进制数据及大型对象(如照片和视频)
- 支持复制和故障恢复
  - 提供了主-从、主-主模式的数据复制及服务器之间的数据复制
- 自动分片
  - 以支持云级别的伸缩性，
  - 支持水平的数据库集群，可动态添加额外的服务器

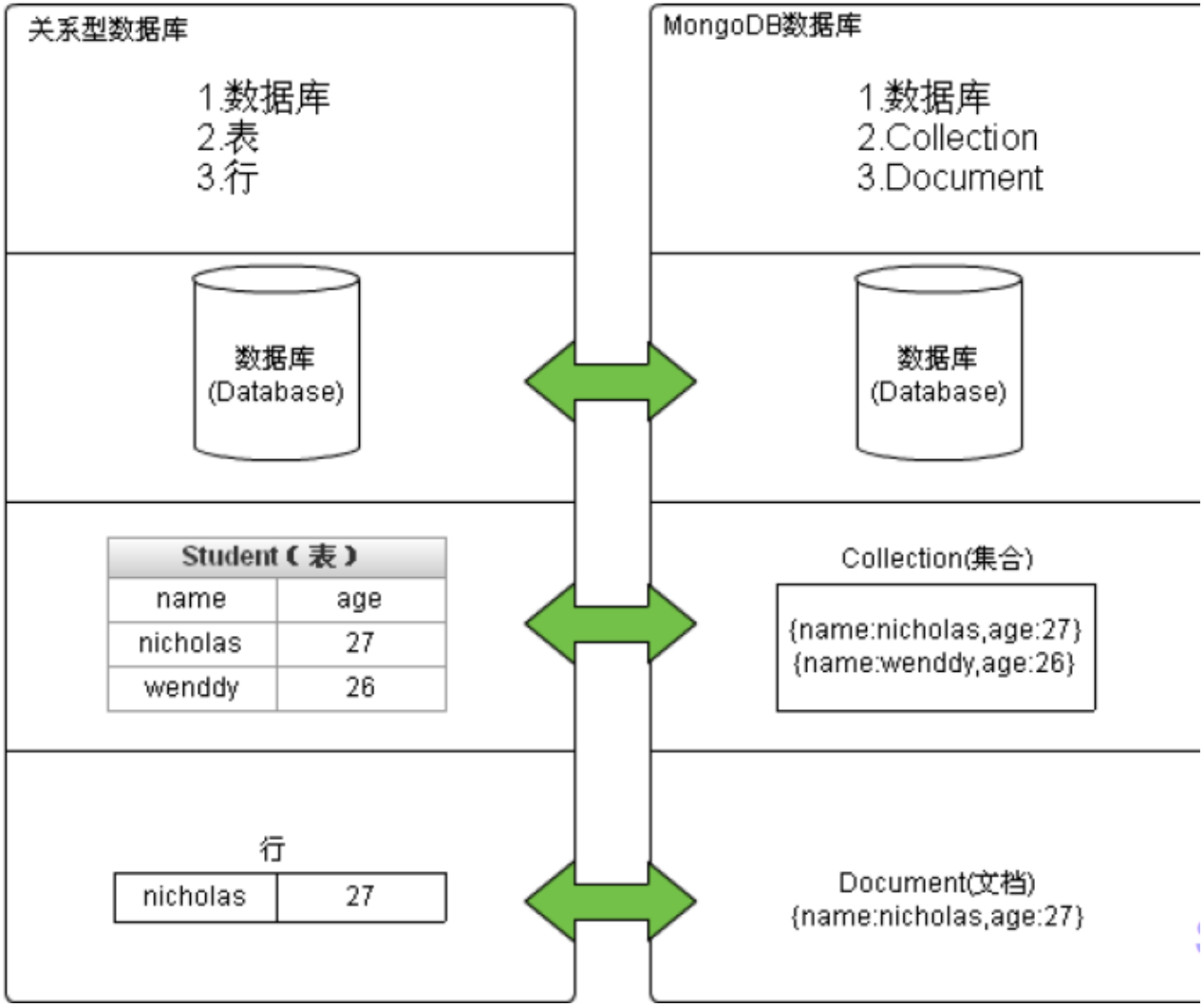
MongoDB适用场景

- 适合作为信息基础设施的持久化缓存层
- 适合实时的插入，更新与查询，并具备应用程序实时数据存储所需的复制及高度伸缩性
- Mongo的BSON数据格式非常适合文档化格式的存储及查询
- 适合由数十或数百台服务器组成的数据库。因为Mongo已经包含了对MapReduce引擎的内置支持

MongoDB不适用场景

- 要求高度事务性的系统
- 传统的商业智能应用
- 复杂的跨文档(表)级联查询

MongoDB数据模型



MongoDB的工作方式

- 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并且最像关系型数据库。
- MongoDB存储的数据格式是key-value对的集合
  - 键是字符串,值可以是数据类型集合里的任意类型,包括数组和文档对象。
- MongoDB是一个免安装的数据库
  - 将它解压后生成一个bin目录,其中包含11个工具命令,除此之外不再需要任何其它的二进制依赖文件。
- 启动数据库只需要关注其中的两个命令：mongod和mongo。
  - 前者是MongoDB数据库进程本身,是核心数据库服务器,
  - 后者是命令行Shell客户端
- MongoDB使用了内存映射文件进行数据管理，把所有空闲内存当缓存使用,且不能指定内存大小。
- 数据空间采用预分配,目的是为了避免形成过多的硬盘碎片。
  - 在32位模式运行时支持的最大文件为2GB。

MongoDB的下载

- MongoDB的官网：
  http://www.mongodb.org/
- MongoDB的下载地址：
  http://www.mongodb.org/downloads