# NoSQL 数据库介绍

ougd . ai . wangjie

## 一、NoSQL 数据

- **简介**

NoSQL(NoSQL = Not Only SQL )，意即反 SQL 运动，指的是非关系型的数据库,是一项全新的数据库革命性运动

随着互联网 web2.0 网站的兴起，传统的关系数据库在应付 web2。0 网站，特别是超大规模和高并发的 SNS 类型的 web2。0 纯动态网站已经显得力不从心,暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。

- **优点**
  - 可以处理超大量的数据
  - 可以运行在便宜的 PC 服务器集群上
  - 打破了性能的瓶颈

    NoSQL 的支持者称，通过 NoSQL 架构可以省去将 Web 或 Java 应用和数据转换成 SQL 友好格式的时间，执行速度变得更快。

    “SQL 并非适用于所有的程序代码，" 对于那些繁重的重复操作的数据，SQL 值得花钱.但是当数据库结构非常简单时,SQL 可能没有太大用处.
  - 没有过多的操作
  - Bootstrap 支持

    因为 NoSQL 项目都是开源的，因此它们缺乏供应商提供的正式支持。这一点它们与大多数开源项目一样，不得不从社区中寻求支持。
- **缺点**
  - 没有正式的官方支持，万一出了差错会是可怕的

    nosql 并未形成一定标准,各种产品层出不穷，内部混乱，各种项目还需时间来检验

## 二、NoSQL 数据库开源软件

### 1. MongoDB：

- **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。他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.Mongo 最大的特点是他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。

- **特点**

它的特点是高性能、易部署、易使用，存储数据非常方便。主要功能特性有:

- 面向集合存储，易存储对象类型的数据

  *“面向集合”（Collenction—Orented）,意思是数据被分组存储在数据集中，被称为一个集合（Collenction)。每个 集合在数据库中都有一个唯一的标识名，并且可以包含无限数目的文档.集合的概念类似关系型数据库（RDBMS）里的表（table)，不同的是它不需要定 义任何模式（schema）。*

- 模式自由

  *模式自由（schema—free)，意味着对于存储在 mongodb 数据库中的文件，我们不需要知道它的任何结构定义。如果需要的话，你完全可以把不同结构的文件存储在同一个数据库里。*

- 支持动态查询
- 支持完全索引，包含内部对象
- 支持查询
- 支持复制和故障恢复
- 使用高效的二进制数据存储，包括大型对象（如视频等)
- 自动处理碎片，以支持云计算层次的扩展性
- 支持 RUBY，PYTHON，JAVA，C++,PHP,C＃等多种语言
- 文件存储格式为 BSON（一种 JSON 的扩展）

  *BSON（Binary Serialized dOcument Format）存储形式是指：存储在集合中的文档，被存储为键-值对的形式。键用于唯一标识一个文档，为字符串类型，而值则可以是各中复杂的文件类型。*

- 可通过网络访问

  *MongoDB 服务端可运行在 Linux、Windows 或 OS X 平台，支持 32 位和 64 位应用，默认端口为 27017。推荐运行在 64 位平台，因为 MongoDB 在 32 位模式运行时支持的最大文件尺寸为 2GB*

## 2. Google BigTable

- **简介**

BigTable 是非关系的数据库,是一个稀疏的、分布式的、持久化存储的多维度排序 Map。Bigtable 的设计目的是可靠的处理 PB 级别的数据，并 且能够部署到上千台机器上。Bigtable 已经实现了下面的几个目标：适用性广泛、可扩展、高性能和高可用性.Bigtable 已经在超过 60 个 Google 的产品和项目上得到了应用，包括 Google Analytics、GoogleFinance、Orkut、Personalized Search、Writely 和 GoogleEarth。这些产品对 Bigtable 提出了迥异的需求,有的需要高吞吐量的批处理，有的则需要及时响应，快速返回数据给最终用户.它们使用的 Bigtable 集群的配置也有很大的差异,有的集群只有几台服务器，而有的则需要上千台服务器、存储几百 TB 的数据.

- **功能**

在很多方面，Bigtable 和数据库很类似：它使用了很多数据库的实现策略。并行数据库和内存数据库已经具备可扩展性和高性能，但是 Bigtable 提供了一个和这些系统完全不同的接口.Bigtable 不支持完整的关系数据模型； 与之相反，Bigtable 为客户提供了

简单的数据模型，利用这个模型,客户可以动态控制数据的分布和格式（alex 注：也就是对 BigTable 而言，数据是没有格式的,用数据库领域的术语说,就是数据没有 Schema，用户自己去定义 Schema），用户也可以自己推测(alex 注:reasonabout）底层存储数据的位置相关性(alex 注：位置相关性可以这样理解，比如树状结构,具有相同前缀的数据的存放位置接近。在读取 的时候，可以把这些数据一次读取出来）。数据的下标是行和列的名字，名字可以是任意的字符串。Bigtable 将存储的数据都视为字符串，但是 Bigtable 本身不去解析这些字符串,客户程序通常会在把各种结构化或者半结构化的数据串行化到这些字符串里。通过仔细选择数据的模式,客户可以控制 数据的位置相关性。最后，可以通过 BigTable 的模式参数来控制数据是存放在内存中、还是硬盘上。

- **特点**
  - 适合大规模海量数据，PB 级数据
  - 分布式、并发数据处理，效率极高
  - 易于扩展，支持动态伸缩
  - 适用于廉价设备
  - 适合于读操作，不适合写操作
  - 不适用于传统关系数据库
- **应用**

BigTable 为谷歌旗下的搜索、地图、财经、打印、以及社交网站 Orkut、视频共享网站 YouTube 和博客网站 Blogger 等业务提供技术支持。

2010 年 9 月,Google 宣布将放弃 MapReduce 新索引系统将迁移至 BigTable 平台。新平台基于 Colossus,也被称为 GFS2.

## 3. HyperTable

- **简介**

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，并且根据我们的想法进行了一些改进。

## 4. Amazon Dynamo

- **简介**

Dynamo 是亚马逊的 key-value 模式的存储平台，可用性和扩展性都很好,性能也不错:读写访问中 99.9%的响应时间都在 300ms 内

- **基本原理**
  - **CAP 原则**

**C**onsistency(一致性）:即数据一致性,简单的说，就是数据复制到了 N 台机器，

如果有更新，要 N 机器的数据是一起更新的.

**Availability**(可用性)：好的响应性能,此项意思主要就是速度。

**Partition tolerance**（分区容错性)：这里是说好的分区方法，体现具体一点,简单地可理解为是节点的可扩展性。

任何分布式系统只可同时满足二点，没法三者兼顾。所以架构师不要将精力浪费在如何设计能满足三者的完美分布式系统，而是应该进行取舍。

- **DHT—分布式哈希表**

**DHT**（**Distributed Hash Table**，分布式哈希表），它是一种分布式存储寻址方法的统称.就像普通的哈希表，里面保存了 key 与 value 的对应关系,一般都能根据一个 key 去对应到相应的节点，从而得到相对应的 value.

在 DHT 算法中，一致性哈希作为第一个实用的算法，在大多数系统中都使用了它. 一致性哈希基本解决了在 P2P 环境中最为关键的问题-如何在动态的网络拓扑中分布存储和路由.每个节点仅需维护少量相邻节点的信息，并且在节点加入/退出系统时，仅有相关的少量节点参与到拓扑的维护中。 至于一致性哈希的细节就不在这里详细说了,要指明的一点是，在 Dynamo 的数据分区方式之后，其实内部已然是一个对一致性哈希的改造了。

- **数据划分**

按分布式系统常用的哈希算法切分数据，分放在不同的节点上。读操作时，也是根据 key 的哈希值寻找对应的节点。Dynamo 使用了 Consistent Hashing 算法，节点对应的不再是一个确定的 hash 值,而是一个 hash 值范围,key 的 hash 值落在这个范围内，则顺时针沿环形查找，碰到的第一个 node 即为所需。

Dynamo 对 Consistent Hashing 算法的改进在于：它放在环上作为一个节点的是一组机器(而不是把一台机器作为节点），这一组机器是通过同步机制保证数据一致的。

如果一个环形内的访问量大了，则可以在两个节点间加入一个新节点以缓解压力，这时 会影响到其后继节点的 hash 范围，需要调整数据。假设一个环形中原本只有 node2、node3、node4，在加入新的 node1 之后，原先 从 node2 查询的部分 key 将改为从 node1 查询，node1 和 node2 中的数据就需要调整，主要是 node1 从 node2 中提取出属于它的数据, 这样做需要选取性能压力不高的时候

*假设我们的数据的 key 的范围是 0 到 2 的 64 次方，然后设置一个常数,比如说 1000，将我们的 key 的范围分成 1000 份.然后再将这 1000 份 key 的范围均匀分配到所有的节点(s 个节点)，这样每个节点负责的分区数就是 1000/s 份分区*

- **数据同步**

Dynamo 的一个节点中的同步是由 client 端来“解决”的，使用所谓的（N, R， W）模型,其中，N 表示节点中机器的总数，R 表示一个读请求需要的机器参与总数，W 代表一个写请求需要的机器参与总数，这些值由 client 端配置。

例如，一个节点有 5 台机器（N=5），client 发出写请求——广播到 5 台机,如果收到 3 个“写完成"的返回消息，即认为写成功（W=3）;client 发出读请求——还是广播到 5 台机,如果收到 2 个“读完成”的返回消息，即认为读成功 （R=2）。对于数据十分重要的应用（如金融），配置可以为(5, 5, 5)，即要求 node 中所有机器的写都成功；而对于数据读写访问量极高的应用，配置可以为（5, 1, 1).

通常 W 不等于 N，于是，在某些情况下一个节点内的机器上的数据可能会有不一致，这时 Dynamo 是通过将多个 Read 的返回结果“合并”来得出最终结果的，使用了所谓 Object Version 和 Vector clock 的技术,即跟踪一个 Object 在不同机器上的版本变化，以确保当多个 Read 请求结果返回不一致时，能够更具其版本信息得出正确的结果。

Dynamo 的这种做法是一种折中，即为了同时保证读和写的效率，写操作不要求绝对同步，而把不同步可能产生的后果推给了读操作。

- **数据恢复**

  Dynamo 的一个节点中一台机器建有一个 Merkle Tree，当两台机器不一致时(如一台机器宕机一段时间)，通过这个 tree 结构，可以快速定位不一致的 Object 来恢复数据。Merkle Tree 又叫 Hash Tree，它把 key 分成几个范围，每个范围算出一个 hash 值，作为叶子，再一层层合并计算上去，这样，从根节点开始比较 hash 值，就 可以快速找到哪几段范围中的 hash 值变化了。

- **可用性的补救(常见问题的解决方法）**

  第一个是 hinted handoff 数据的加入:在一个节点出现临时性故障时，数据会自动进入列表中的下一个节点进行写操作，并标记为 handoff 数据，在收到通知需要原节点恢复时重新把数据推回去。这能使系统的写入成功大大提升.

  第二个是向量时钟来做版本控制:用一个向量（比如说［a,1］表示这个数据在 a 节点第一次写 入）来标记数据的版本,这样在有版本冲突的时候，可以追溯到出现问题的地方。这可以使数据的最终一致成为可能.（Cassandra 未用 vector clock,而只用 client timestamps 也达到了同样效果。）

  第三个是 Merkle tree 来提速数据变动时的查找：使用 Merkle tree 为数据建立索引，只要任意数据有变动,都将快速反馈出来。

  第四个是 Gossip 协议：一种通讯协议，目标是让节点与节点之间通信,省略中心节点的存在,使网络达到去中心化.提高系统的可用性.

## 5. Apache Cassandra

- **简介**

  Apache Cassandra 是一套开源分布式 Key-Value 存储系统.它最初由 Facebook 开发，用于储存特别大的数据。Facebook 目前在使用此系统。

- **主要特征**

  Cassandra 的主要特点就是它不是一个数据库，而是由一堆数据库节点共同构成的一个分布式网络服务，对 Cassandra 的一个写操作，会被复制到其他节点上去，对 Cassandra 的读操作，也会被路由到某个节点上面去读取。对于一个 Cassandra 群集来说，扩展性能 是比较简单的事情，只管在群集里面添加节点就可以了。

  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 Dynomite(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。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。P2P 去中心化的存储。很多方面都可以称之为 Dynamo 2.0.

  - 分布式
  - 基于 column 的结构化
  - 高伸展性

- **和其它数据库相比，其突出的特点**

  - 模式灵活

    *使用 Cassandra,像文档存储，你不必提前解决记录中的字段.你可以在系统运行*

*时随意的添加或移除字段。这是一个惊人的效率提升，特别是在大型部 署上.*

- 真正的可扩展性

  *Cassandra 是纯粹意义上的水平扩展。为给集群添加更多容量,可以指向另一台电脑。你不必重启任何进程，改变应用查询，或手动迁移任何数据。*

- 多数据中心识别

  *你可以调整你的节点布局来避免某一个数据中心起火，一个备用的数据中心将至少有每条记录的完全复制*

- 范围查询

  *如果你不喜欢全部的键值查询，则可以设置键的范围来查询.*

- 列表数据结构

  *在混合模式可以将超级列添加到 5 维。对于每个用户的索引,这是非常方便的。*

- 分布式写操作

  *有可以在任何地方任何时间集中读或写任何数据。并且不会有任何单点失败。*

## 6. HBase

- **简介**

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.

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。

## 7. MemBase

- **简介**

Membase 是开源项目，源代码采用了 Apache2。0 的使用许可。

Membase 容易安装、操作，可以从单节点方便的扩展到集群，而且为 memcached（有线协议的兼容性)实现了即插即用功能,在应用方面为开发者和经营者提供了一个比较低的门槛。做为缓存解决方案，Memcached 已经在不同类型的领域（特别是大容量的 Web 应用）有了广泛的使用，其中 Memcached 的部分基础代码被直接应用到了 Membase 服务器的前端。

通过兼容多种编程语言和框架，Membase 具备了很好的复用性。在安装和配置方面,Membase 提供了有效的图形化界面和编程接口，包括可配置 的告警信息。

Membase 的目标是提供对外的线性扩展能力，包括为了增加集群容量，可以针对统一的节点进行复制. 另外,对存储的数据进行再分配仍然是必要的。

这方面的一个有趣的特性是 NoSQL 解决方案所承诺的可预测的性能，类准确性的延迟和吞吐量。通过如下方式可以获得上面提到的特性：

- 自动将在线数据迁移到低延迟的存储介质的技术（内存，固态硬盘，磁盘）
- 可选的写操作一一异步，同步（基于复制，持久化)
- 反向通道再平衡[未来考虑支持］
- 多线程低锁争用

- 尽可能使用异步处理
- 自动实现重复数据删除
- 动态再平衡现有集群

通过把数据复制到多个集群单元和支持快速失败转移来提供系统的高可用性.

ougd . ai . wangjie