链滴

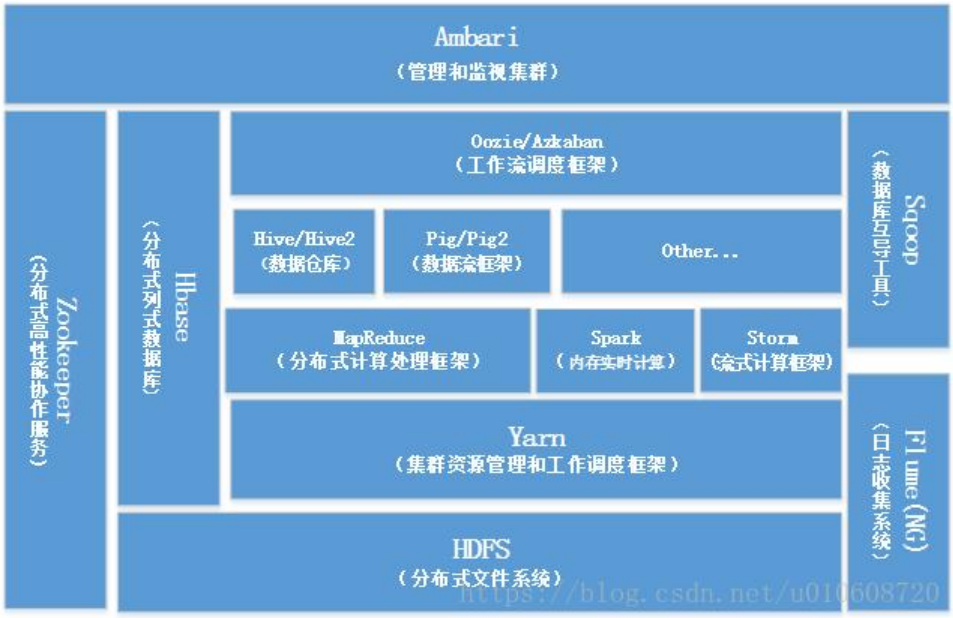
HBase - 分布式存储系统

作者：xjtushilei

原文链接：https://ld246.com/article/1526469455892

来源网站：链滴

HBase(Hadoop Database)来自于Apache组织，早起隶属于Hadoop项目，是其下的一个开源子项，到目前成长为一个独立的顶级开源项目。本文从相对底层的角度讲一下HBase。



Hbase在Hadoop生态系统中的位置（图片来源：https://blog.csdn.net/u010608720/article/detail/80092260)

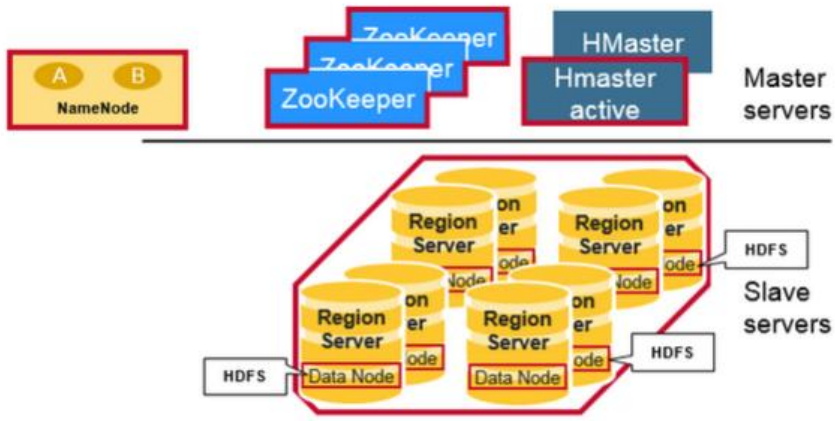
Hbase是一个分布式的、面向列的开源数据库,是对基于GFS的Bigtable数据库的开源实现。类似 Google Bigtable利用GFS作为其文件存储系统, Hbase利用 Hadoop HDFS作为其文件存储系统;Google运 Mapreduce来处理Bigtable中的海量数据, Hbase同样利用 Hadoop MaReduce来处理 Hbase中的量数据; Google Bigtable利用 Chubby作为协同服务, Hbase利用Zookeeper作为协同服务。

Hbase底层使用Hadoop的HDFS作为文件存储系统。借助于此,Hbase首先实现了海量数据存储能力,时能显著提高数据存储的可靠性。Hbase的上层可以使用 Mapreduce并行计算框架,这使它对于海量据处理又具有了明显的效率优势。 Hbase融合了HDFS的海量数据存储能力和 Mapreduce的并行计能力,能够实现海量数据的高效处理,并提供高可靠性保证。

Hbase作为列数据库,存储的数据介于键值(key-value)映射和关系型数据之间。保存在Hbase中的海数据在逻辑上被组织成一张很大的表,支持动态添加数据列,并且每个数据值又以时间戳区分,保留了数的多个版本。

Hbase的体系结构

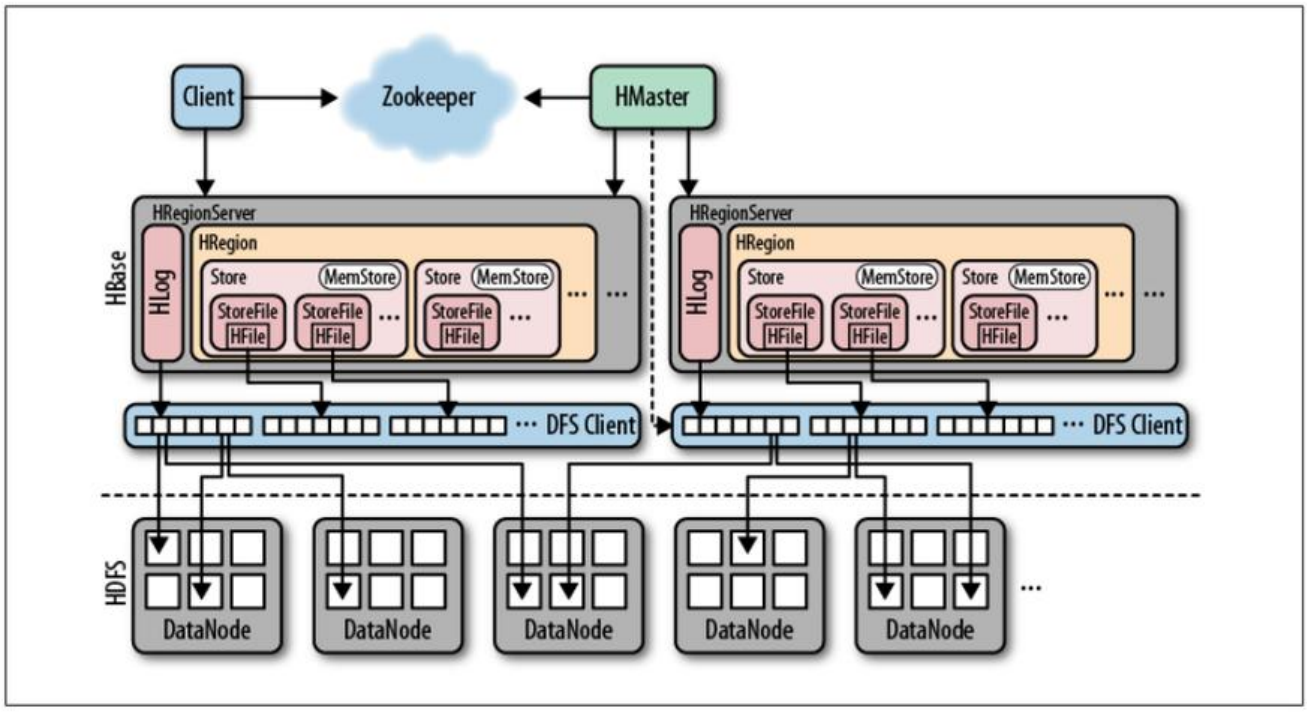
与HDFS类似, Hbase也遵从简单的 Master-Slave 体系结构。它是由 HMasterServer(Hbase Master Server) 和 HRegionServer(Hbase Region Server)构成的服务器集群。其中, HMasterServer充当master角色，主要负责将Region分配给RegionServer、动态加载或卸载RegionServer、对Regionserve实现负载均衡、管理全局数据模式定义等功能; Hregionserver充当 Slave角色,负责与 Client进行交和数据的读写操作。体系结构的相似性使得 Hbase与Hadoop一样,可以方便地进行横向扩展,通过不增加廉价的机器来增加计算和存储能力。



Hbase的体系结构（图片来源：http://www.blogjava.net/DLevin/archive/2015/08/22/426877.html）

HBase的组件结构

从互联网上搜到的组件结构图中，最经典的就是下图所示的结构图。我们可以看到，在hbase的集群，主要有四大功能组件。分别是HMasterServer，HRegionSever，Zookeeper和Client Library。



Hbase的组件结构（图片来源：http://www.aichengxu.com/other/6456119.htm）

主服务器(Hmasterserver)

Hmasterserver是Hbase整个集群的管理者,它主要负责管理数据表(Table)和区域(Region)以及响应户的数据请求,保证用户对数据的访问。它的具体任务包括：

- 保存和管理关于存取数据信息的元数据信息。

- 管理用户对表的增加、删除、修改和查询。
- 调整Region的分布,管理Hregionserver的数据和负载均衡。
- 负责新 Region的分配。
- 在 Hregionserver停机后,负责失效Hregionserver上的Region迁移。
- 处理对 Hbase schema的更新请求。

虽然 Master是 Hbase集群中的中心点,但是 Hbase并不存在单点失效问题,因为 Hbase中可以启动多个 Master,并通过 Zookeeper的 Master Election机制保证总是只有一个 master运行。当正在运行的 master机器出现故障的时候,系统会转移到其他 master来接管。

数据服务器(Hregionserver)

Hregionserver是每个 Region(区域)的管理者和用户服务的提供者。它管理本地数据,并响应用户的数读取请求。一般情况下,在 Hbase集群中,每台机器上只运行了一个 Region-Server。

Region的存储与读取

Hbase是 Bigtable的开源实现,也是列存储数据库。Hbase以表的形式存储数据。当Hbase数据表的小超过设置值时, Hbase会把数据表在行的方向上分隔为多个区域(Region),每个 Region包含全部数的一个子集。从物理上来说，一张表被拆分了多块,每块就是一个 Region。Region是 Hbase中分布存储和负载均衡的最小单元。不同的 Region可以分别位于不同的 Hregionserver上,但同一个 Regio不会被分拆存储到多个 Hregionserver之上。

进一步, Region由多个Store组成, Store是 Hbase的物理存储核心。每个 Store存储表中的1个列簇。Sore由两部分组成: Memstore和 StoreFile。 Memstore是有序的主存缓冲区(Sorted Memory Buffer,用户写入的数据首先暂存于 Memstore,当 Memstore写满后会执行 Flush操作，写入外存的一个 Stoefile(其底层实现是 Hfile)中。这样, Store file文件数量会持续增长。当Storefile文件数量增长到设定值时,会触发合并(compact)操作,将多个 Store File合并成一个Storefile。合并过程中会进行版本合并数据删除。

Region Server响应用户的数据读取请求。它首先会访问本地的 HMemcache缓存,其中保存着最近更的数据。如果缓存里面没有该数据,才会到 Store中进行磁盘查找。

恢复管理

Hregionserver使用本地磁盘上的 Hlog文件进行恢复管理。HLog文件记录着所有更新操作。在 Hregonserver启动的时候,会检查本地的Hog文件,查看最近一次执行缓冲区写出操作后是否有更新操作:果没有更新,就表示数据缓冲区中的所有更新都已经写入磁盘文件中了;如果有更新, Hregionserver会把这些更新写入高速缓存,然后调用 flush cache过程写入外存磁盘。最后, Hregionserver会删除旧的Log文件,并开始让用户访问数据。

Region的合并与分裂

Hregionserver通过合并操作将多而小的 StoreFile合并成一个越来越大的 Store File。当单个StoreFil大小超过一定的阈值后,会触发 Hregionserver的Split操作,把当前 Region分裂成两个Region,并且报给 Master,让它来决定由哪个 Hregionserver存放新拆分而得的 Region。当新的 Region拆分完成并把引用也删除后, Hregionserver负责删除旧的Region。另外,当两个Region足够小时,Hbase负责将们合并。

Hbase集群中的分布式协调器（Zookeeper）

Zookeeper是 Hbase集群的协调者。总体来说, Zookeeper的功能有：

1. 保证在任何时于刻,集群中只有一个 master

2. 存储所有 Region的入口地址。 Hmasterserver启动时会将 Hbase系统表Root加载到 Zookeeper上并通过 Zookeeper cluster获取当前系统表的Meta的存储所对应的 Regionserver信息。

3. 实时监控 Region Server的状态,将 Hregionserver的上线和下线信息实时通知给 HMasterServer。Hregionserver会以短暂的方式向 Zookeeper注册,使得 Hmasterberver可以随时感知各个 Hregionsrver是否在线的状态信息。

4. 存储 Hbase的 schema,包括有哪些表,每个表有哪些列族。

客户端类库(Client Library)

Client Library是封装的用于支持 Hbase数据操作和客户端开发的API集合。

Hbase/Bigtable 与 RDBMS

Hbase是 Bigtable的开源实现,在此我们以 Hbase为例,总结 Hbase、 Bigtable与关系型数据库管理统 RDBMS的主要不同。Hbase是一个分布式的、基于列模式的映射数据库,它只能表示很简单的键值射关系。它有如下特点:

1. **数据类型**，HBase只有简单的字符类型，所有的类型都是交由用户自己处理，它只保存字符串。而系数据库有丰富的类型和存储方式。

2. **数据操作**：HBase只有很简单的插入、查询、删除、清空等操作，表和表之间是分离的，没有复杂表和表之间的关系，而传统数据库通常有各式各样的函数和连接操作。

3. **存储模式**：HBase是基于列存储的，每个列族都由几个文件保存，不同的列族的文件是分离的;传统关系数据库是基于表结构和模式保存的。

4. **数据索引**: Hbase没有真正的索引,由于行是顺序存储的,每行中的列也是顺序存储的,因此不存在索膨胀问题,插入性能和表的大小无关;关系数据库支持多级索引技术,索引页面相对于数据页面来说要小多,利用索引可以加快查找和连接的速度,但是,在数据更新频繁的应用中,索引维护代价不容忽视。

5. **数据维护**: Hbase进行更新操作,插入新版本数据时,旧版本的数据仍然会保留,所以实际上是插入了的数据,属于典型的异地更新策略;传统的关系型数据库则采用原地替换与修改策略,辅以基于日志的恢策略实现数据库的弹性。

6. **可伸缩性**: Hbase支持线性扩展,能够方便地增减节点数量,从而修改集群规模和系统容量,新节点加集群后,运行 Hregionserver, Hbase能够自动实现 Region的负载均衡;传统的关系数据库通常需要通中间层来实现受控的节点加入或者删除。

7. **系统目标**: Hbase强调灵活的可伸缩性以支持海量数据存储,强调并行处理、数据的“最终一致性”思想以支持面向海量数据的实时查询处理;传统的关系型数据库强调事务的ACID特性,强调数据的“实一致性”,强调基于SQL语言的复杂查询支持能力。

从上面的对比可以看出, Bigtable和Hbase之类的基于列模式的分布式数据库,更适合海量存储和实时询处理。另外,互联网上的内容是以字符为基础的, Bigtable和 Hbase正是针对互联网应用而开发出来数据库。 Hbase从设计之初,就强调灵活的分布式架构,用户可以基于异构、廉价的硬件设备组建 Hbas大数据存储和处理集群。

其他

推荐了解的还有：

- **Cassandra**：Facebook开发的nosql，在集群架构上采用P2P环的方式，即一致性散列，节点通讯用Gossip协议。
- **OceanBase**：淘宝自己的nosql，注重CAP中的CA。架构采用Master-Slave模式。
- **Etcd**和**zookeeper**: 特点是raft和paxos的实现。