分布式存储系统 HBase 原理解析

郝树魁
（北京邮电大学信息与通信工程学院，北京 100876）

摘要：随着数据量的膨胀和计算机硬件价格的下降，快速处理海量数据的需求促使了分布式计算的兴起和发展。Hadoop 除了提供分布式文件系统和支持 MapReduce 的计算框架，还在这一计算框架上提供了可扩展的结构化数据的分布式存储系统 HBase 等。本文在简要介绍 HDFS 体系结构和 MapReduce 逻辑数据流的基础上，解析了分布式存储系统 HBase 的原理并对分布式存储系统的未来发展做出展望。

关键词： 分布式计算，分布式存储，HBase，Hadoop，HDFS，MapReduce
中图分类号：TP392

Analysis of HBase Distributed Storage System

Hao Shu Kui
(School of Information and Communications Engineering, Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract: With the expansion of the amount of data and computer hardware prices fall, the needs of rapid processing mass data is prompting the development the distributed computing. The Hadoop framework provides distributed file system and supports MapReduce calculation, besides the extensibility of distributed storage system for structured data HBase. On the basis of giving the architecture of HDFS and the logical data streams of MapReduce, this paper briefly introduces the principle of HBase distributed storage system and proposes the future development of distributed storage system.

Key words: Distributed computing;Distributed storage;HBase;Hadoop;HDFS;MapReduce

0 引言

我们生活在一个数据爆炸的时代，并且每天都会新增巨大的数据，例如来自互联网、传媒和科学实验数据等。来自互联网数据中心(IDC)的报告，“数字地球”的数据量已经达到 Zattebytes 的数量级。(其中 1ZB = 1012byte)。对于电子商务、科学计算等行业来说，合理的存储和管理海量数据，并且从中提取数据的价值显得尤为重要。

1 数据存储和分析

随着数据的膨胀，使用传统的磁盘和 RDBMS（关系数据库管理系统）面临着越来越多的挑战。得益计算机硬件的发展，磁盘造价迅速下降的同时磁盘容量变的越来越大。同时，磁盘容量的变大提升存储量的同时也带来了一个问题，即磁盘寻址的时间通常远大于磁盘读取数据的时间。并且，磁盘寻址性能的提升速度要慢于读取数据性能提高的速度。RDBMS 适用于点查询和更新，对于提供索引的数据集提供低延迟的检索和更新小数据量的数据。RDBMS 管理符合预定于的 Scheme 的特定格式的结构化数据。但对于非结构化半结构化数据，松散的数据，RDBMS 并不高效。

作者简介：郝树魁（1986），男，无职称，计算机应用技术. E-mail: haosk86@yahoo.com.cn

2 技术创新

Google 公司三篇云计算的经典论文改变了人们处理海量数据的思索方式：即 BigTable--一个分布式的结构化数据存储系统；The Google File System；Google MapReduce。它们对 Hadoop 的发展起到了很大启发作用。谈到 Hadoop，就不得不提到 Lucene 和 Notch。Lucene 不是一个应用程序，而是提供一个纯 Java 的高性能全文索引引擎工具包，它可以方便的嵌入到各种实际应用中来实现全文索引/搜索功能。Nutch 是一个应用程序，是一个以 Lucene 为基础实现的搜索引擎应用，Lucene 为 Nutch 提供文本索引和搜索的 API，Nutch 不仅有搜索功能，还有数据抓起的功能。在 Nutch0.8.0 版本之前，Hadoop 还属于 Nutch 的一部分，而从 Nutch0.8.0 版本开始，将其中实现的 NDFS 和 MapReduce 剥离出来成立了一个新的 Apache 开源项目，这就是 Hadoop。Nutch0.8.0 版本较之前的 Nutch 在架构上有了根本性的变化，那就是完全构建在 Hadoop 基础之上。[1]在 Hadoop 中实现了 Google 的 GFS(Google File System)、MapReduce 和 BigTable 算法，并相应的取名为 HDFS、MapReduce 和 HBase，使 Hadoop 成为了一个分布式的计算平台。

其实，Hadoop 并不仅仅是一个用于存储的分布式文件系统，而是设计用来在由通用计算设备组成的大型集群上执行分布式应用的框架。Apache 的 Hadoop 的项目中包含了下列产品，如图所示：

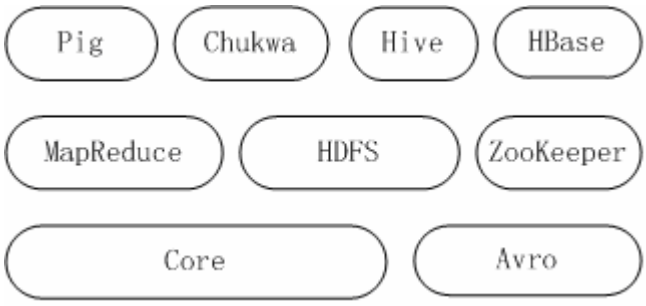

表 1　Hadoop 基本组成

Pig 是在 MapReduce 上构建的查询语言，适用于大量并行计算。Chukwa 是基于 Hadoop 集群中监控系统，简单来说就是一个“看门狗”(WatchDog)。Hive 是 DataWareHouse 和 MapReduce 交集，适用于 ETL 方面的工作。[2]HBase 是一个面向列的分布式存储系统。MapReduce 是 Google 提出的一种算法，用于超大型数据集的并行运算。HDFS 可以支持千万级的大型分布式文件系统。Zookeeper 提供的功能包括：配置维护、名字服务、分布式同步、组服务等，用于分布式系统的可靠协调系统。Avro 是一个数据序列化系统，设计用于支持大批量数据交换的应用。

3 Hadoop 的主要两个部分：HDFS 和 MapReduce

3.1 HDFS

HDFS 即 Hadoop Distributed File System（Hadoop 分布式文件系统)。HDFS 是 GFS 的开源实现。HDFS 很适合大数据集的应用，并且提供了对数据读写的高吞吐率。HDFS 是以流式数据访问模式存储超大文件而设计的文件系统，在商用硬件的集群上运行。

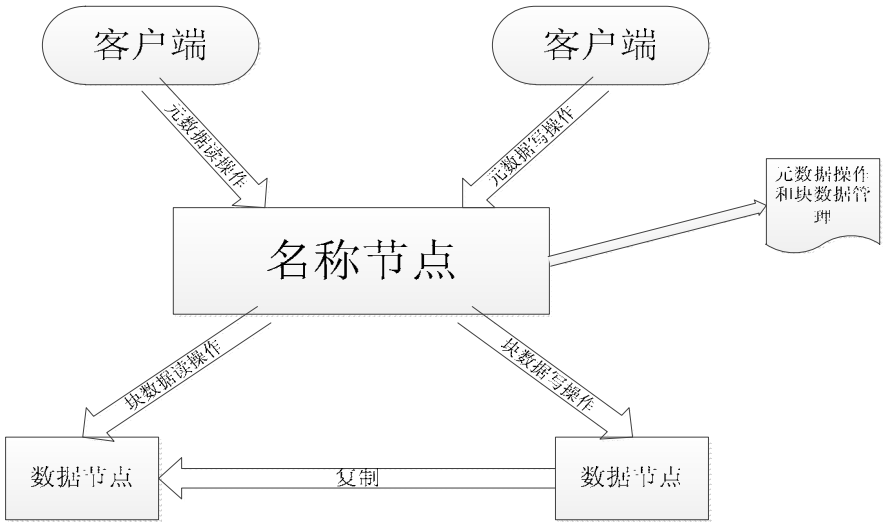

图 2　HDFS 体系结构

HDFS 是一个 master/slave 结构的文件系统，HDFS 集群有两种节点，以管理者-工作者的模式运行，即一个 NameNode（命名节点)和多个 DataNode(数据节点)。NameNode 管理文件系统的命名空间。它维护着这个文件系统树及这个树内所有的文件和索引目录。客户端通过与 NameNode 和 DataNode 交互来访问整个文件系统。

3.2 MapReduce

MapReduce 是一种用于数据处理的编程模型，是 Google MapReduce 的开源实现。MapReduce 程序本质上是一种简化并行计算的编程模型，所以其优势在于处理大型数据集。MapReduce 的工作过程分为两个阶段：map 阶段和 reduce 阶段。

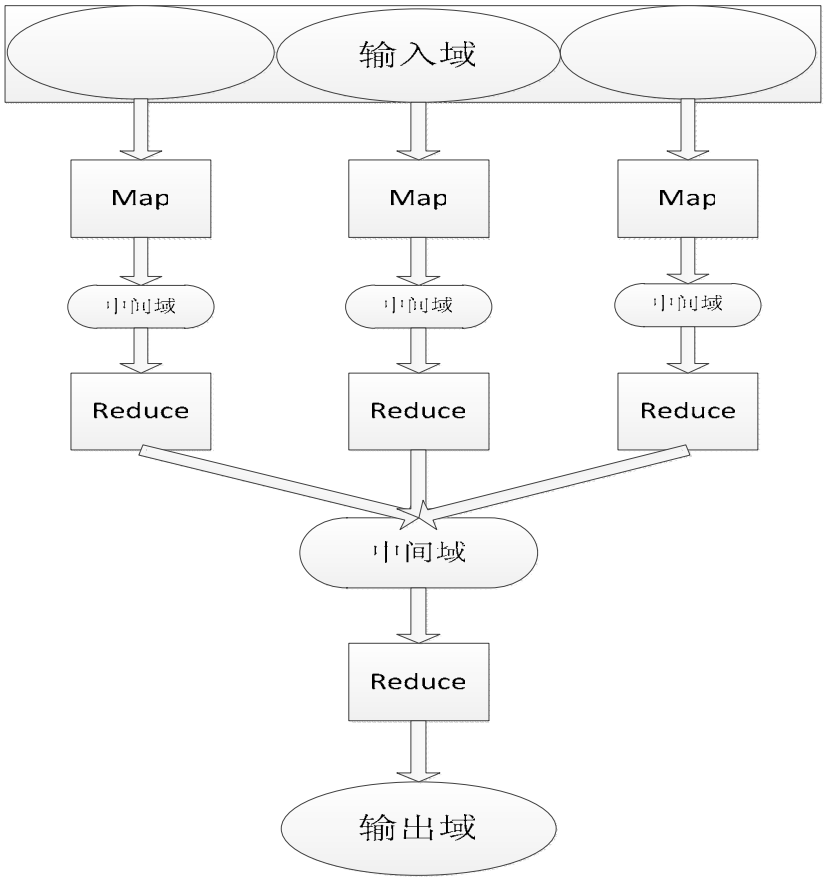

图 3　MapReduce 的逻辑数据流

每个阶段都有键/值对作为输入和输出，简单说来，map 阶段把一组键/值对映射为另一个键/值对，其映射的规则由 map 函数来指定。Reduce 阶段是把一组键/值对进行归约，这个归约的规则由 reduce 函数指定。客户端启动的 MapReduce 应用程序成为 JobTracker。类似于 NameNode，它是 Hadoop 集群中唯一负责控制 MapReduce 应用程序的系统。JobTracker 使用文件块信息确定如何创建和管理 TaskTracker，为每个 DataNode 创建唯一的 TaskTracker。每个 TaskTracker 将状态和任务进度报告回传给 JobTracker。[3]

4 HBase

4.1 HBase 简介

HBase 是 Apache 的 Hadoop 项目的子项目。HBase 是一个开源的、分布式的、面向列的存储系统，该技术来源于 Google 的论文：“BigTable--一个分布式的结构化数据存储系统”。HBase 是 Google BigTable 的开源实现。就像 BigTable 利用了 GFS 所提供的分布式数据存储一样，HBase 在 Hadoop 平台上提供了类似于 BigTable 的能力，依托 HDFS 作为最基本存储基本单元，通过使用 Hadoop 的 DFS 工具可以查看这些数据及其存储结构，还可以通过 MapReduce 对 HBase 进行操作。如图所示：

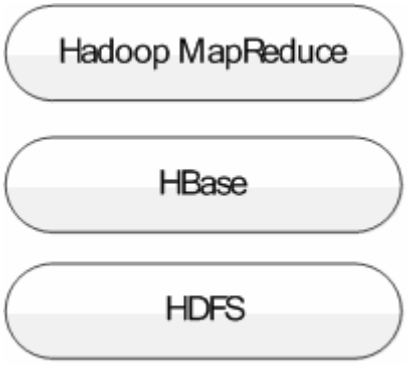

图 4　HBase 于 HDFS、MapReduce

HBase 的需求起源于需要一个类似于 BigTable 的数据存储系统来保存 Web 表格，一个存放 Web 文档及其以 URL 作为关键字的属性宽泛的表。HBase 不同于传统的关系数据库，它是适合于非结构化数据存储的数据库。所谓非结构化数据存储就是说 HBase 是基于列的而不是基于行的模式。这里简要介绍一下列存储。基于列存储的数据库中的数据表的每列单独存放；数据即使索引；查询时只访问涉及的列，大量降低系统 I/O；每列可以由一个线程来处理，支持了查询时的并发处理。

HBase 是介于 Map Entry(key & value)和 DB Row 之间的一种数据存储方式。不仅是一个 key 对应一个 value，还可能需要存储多个属性的数据结构，但没有传统数据库表中那么多的关联关系，即所谓松散数据。简单来说，HBase 中创建的表的属性可以根据需求去动态增加，在 HBase 中没有表与表之间关联查询。Apache HBase 的一个数据行拥有一个可选择的键和任意数量的列。表是疏松存储的，因此用户可以给行定义各种不同的列，这样的功能在实际中很实用。[4]

HBase 表中存储的行记录有三个基本类型的定义：Row Key，Time Stamp，Column。Row Key 是行在 HBase 中的唯一标识，Time Stamp 是每次数据操作对应关联的时间戳，可以看作类似于版本信息，Column 定义为：<family>:<label>，通过这两部分可以唯一的指定一个数据的存储列。一个表格的 family 集合是固定的，label 集合是可以动态增删的。family 的定义和修改需要对 HBase 作类似于 DB 的 DDL 操作，而对于 label 的使用，则不需要定义直接可以使用，这也为动态定制列提供了一种手段。family 另一个作用在于物理存储优化读写操作，同 family 的数据物理上保存的会比较临近。

看一下逻辑数据模型：

表 1　逻辑数据模型

Row Key	*Time Stamp*	*Column "content:"*	*Column "anchor:"*		*Column "mime:"*
"cn.edu.byr.forum"	t5		"anchor:bupt.edu.cn"	"Forum"	
	t4		"anchor:bt.byr.cn"	"Forum"	
	t3	"<html>…"			"text/html"
	t2	"<html>…"			"text/css"
	t1	"<html>…"			"image/gif"

上表中有一列，列的唯一标识为 cn.edu.byr.forum，每一次逻辑修改都有一个 timestamp 关联对应，一共有四个列定义：<contents:>,<anchor:bupt.edu.cn>,<anchor:bt.byr.cn>,<mime:>。

如果用传统的概念来将 HBase 作解释，那么 HBase 可以看作一个 DB Schema，每一个 Row 就是一个表，Row key 就是表名，这个表根据列的不同可以划分为多个版本，同时每个版本的操作都会有时间戳关联到操作的行。

再看一下 HBASE 的物理数据模型：

表 2　物理数据模型(1)

Row Key	Time Stamp	Column “mime:”
“cn.edu.byr.forum”	t3	“text/html”
	t2	“text/css”
	t1	“image/gif”

表 3　物理数据模型(2)

Row Key	Time Stamp	Column “contents:”
“cn.edu.byr.forum”	t3	“<html>…”
	t2	“<html>…”
	t1	“<html>…”

表 4　物理数据模型(3)

Row Key	Time Stamp	Column “anchor:”	
“cn.edu.byr.forum”	t5	“anchor:bupt.edu.cn”	“Forum”
	t4	“anchor:bt.byr.cn”	“Forum”

物理数据模型其实就是将逻辑模型中的一个 Row 分割成为根据 Column family 存储的物理模型。

4.2 HBase 原理

HBase 依托于 Hadoop 的 HDFS 作为存储基础，结构类似于 Hadoop 的 Master-Slave 模式，HBase 中有且仅有一个 Master Server，负责管理所有的 HRegion Server，但 HBase Master Server 本身并不存储 HBase 中的任何数据。HBase 逻辑上的 Table 被定义成为一个 Region 存储在某一台 HRegion Server 上，HRegion Server 与 Region 的对应关系是一对多的关系，负责多个 Region 向客户端提供服务。[5]每一个 HRegion 在物理上会被分为三个部分：HMemcache、HLog、HStore，分别代表了缓存，日志，持久层。

提交更新操作涉及到两部分实体：HMemcache 和 HLog，HMemcache 就是为了提高效率在内存中建立缓存，HLog 是作为同步 HMemcache 和 HStore 的事务管理日志，在 HRegion Server 周期性的发起 Flush Cache 命令的时候，就会将 HMemcache 中的数据持久化到 HStore 中，同时会清空 HMemcache 中的数据。采用这种策略来做数据缓存和同步。

在读取 Region 信息的时候，优先读取 HMemcache 中的内容，如果未取到再去读取 HStore 中的数据。

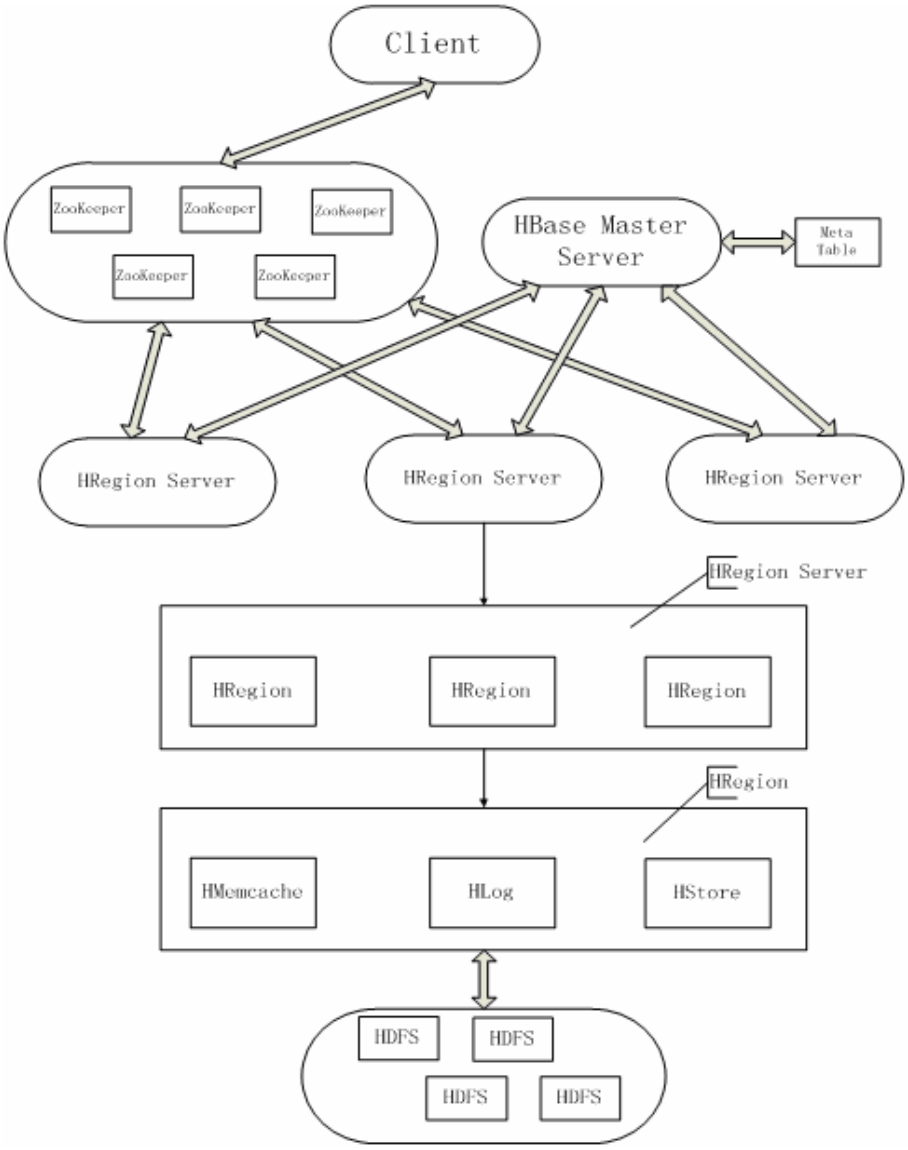

图 5　HBase 原理

客户端通过 Zookeeper 组件能获知哪个 Server 管理-ROOT- Region。那么客户端就去访问管理-ROOT-的 Server，在.META.中记录了 HBase 中所有表信息，从而获取 Region 分布的信息。客户端获取 Region 位置信息后，将会缓存这个信息并直接访问 HRegion Server。重复如上查找缓存之后，即使不访问.META.表也能知道去访问哪个 HRegion Server。HBase 中包含两种基本类型的文件，一种用于存储 WAL 的 log，另一种用于存储具体的数据，通过 DFS 客户端于 HDFS 进行交互实现存储。

4.3 HBase 分布式安装

软件环境：JDK 版本：jdk1.6；操作系统：Ubuntu 10.04；Hadoop 0.20.2 版本；/etc/hosts 文件内容为：

```
192.168.0.101    m1   # NameNode
192.168.0.102    s2   # DataNode
192.168.0.103    s3   # DataNode
```

m1 主机上解压下载的 hbase-0.20.6。修改 hbase-0.20.6/conf/hbase-env.sh 文件。[6]指定本

地 JDK 安装路径：export JAVA_HOME=/usr/java/jdk1.6。修改 hbase-0.20.6/conf/hbase-site.xml。内容如下：

```
<configuration>
<property>
    <name>hbase.rootdir</name>
    <value>hdfs://m1:9000/hbase</value>
</property>
<property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
</property>
<property>
    <name>hbase.master.port</name>
    <value>60000</value>
</property>
<property>
      <name>hbase.zookeeper.quorum</name>
      <value>m1,s2,s3</value>
</property>
</configuration>
```

修改 hbase-0.20.6/conf/regionservers 文件（同 Hadoop 的 slaves 文件内容相同），内容如下：

```
s2
s3
```

将 hbase-0.20.6/目录拷贝到 s2 和 s3 主机上。

通过 hbase-0.20.6/bin/start-hbase.sh 脚本启动 HBase。两种方式查看 HBase 数据信息：在 hbase-0.20.6/bin 目录下执行 hbase shell 命令进入 HBase 控制台查看数据存储信息；通过 Web 查看 HBase。如下所示：

查看 Master http://192.168.0.101:60010/master.jsp

查看 Region Server http://192.168.0.102:60030/regionserver.jsp

http://192.168.0.103:60030/regionserver.jsp

5 总结

HBase 为海量数据的实时响应提供了一个很好的开源解决方案。它一方面借鉴了 HDFS 的高可靠性和可伸缩性，另一方面又借鉴了 BigTable 的高效数据组织形式。HBase 提供了一个类似于 MySQL 关系数据库的 shell，可对 HBase 内的相关表和 family 进行交互。亦可通过 API 以程序的方式访问 HBase 的数据。HBase 是一个新兴的 Apache 项目，但它的确展示了成为主流解决方案的前景。

[参考文献] (References)

[1] Chuck Lam, James Warren. Hadoop in action, Manning Publications [2009].
[2] http://wiki.apache.org/hadoop/ , [2010, November 12].
[3] Google Research Publications. MapReduce: Simplified Data Processing on Large Clusters [December, 2004].
[4] Google Research Publications. Bigtable: A Distributed Storage System for Structured Data [November, 2006].
[5] Tom White. Hadoop: The Definitive Guide, Second Edition O'Reilly | YAHOO! PRESS [October, 2010].
[6] http://hbase.apache.org/ , [2010, October 6].