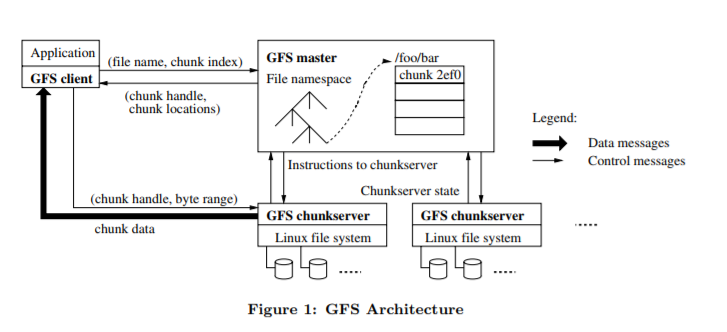
**Paper读后感**

**18301174 许歆笛**

**1.GFS**

**1.设计**

GFS提供了一套类似传统文件系统的API接口函数，虽然并不是严格按照POSIX等标准API的形式实现的。文件以分层目录的形式组织，用路径名来标识。我们支持常用的操作，如创建新文件、删除文件、打开文件、关闭文件、读和写文件。

如Figure2所示，一个GFS集群包含一个单独的Master节点 ，这里的一个单独的Master节点的含义是GFS系统中只存在一个逻辑上的Master组件。后面我们还会提到Master节点复制，因此，为了理解方便，我们把Master节点视为一个逻辑上的概念，一个逻辑的Master节点包括两台物理主机，即两台Master服务器）、多台Chunk服务器，并且同时被多个客户端访问。

**2.系统交互**

在设计这个系统时，一个重要的原则是最小化所有操作和Master节点的交互。带着这样的设计理念，我们现在描述一下客户机、Master服务器和Chunk服务器如何进行交互，以实现数据修改操作、原子的记录追加操作以及快照功能。变更是一个会改变Chunk内容或者元数据的操作，比如写入操作或者记录追加操作。变更操作会在Chunk的所有副本上执行。我们使用租约（lease）机制来保持多个副本间变更顺序的一致性。Master节点为Chunk的一个副本建立一个租约，我们把这个副本叫做主Chunk。主Chunk对Chunk的所有更改操作进行序列化。所有的副本都遵从这个序列进行修改操作。为了提高网络效率，我们采取了把数据流和控制流分开的措施。在控制流从客户机到主Chunk、然后再到所有二级副本的同时，数据以管道的方式，顺序的沿着一个精心选择的Chunk服务器链推送。我们的目标是充分利用每台机器的带宽，避免网络瓶颈和高延时的连接，最小化推送所有数据的延时。

**3.MASTER节点的操作**

Master节点执行所有的名称空间操作。此外，它还管理着整个系统里所有Chunk的副本：它决定Chunk的存储位置，创建新Chunk和它的副本，协调各种各样的系统活动以保证Chunk被完全复制，在所有的Chunk服务器之间的进行负载均衡，回收不再使用的存储空间。不同于许多传统文件系统，GFS没有针对每个目录实现能够列出目录下所有文件的数据结构。GFS也不支持文件或者目录的链接。每个Master节点的操作在开始之前都要获得一系列的锁。

**4.容错和诊断**

在GFS集群的数百个服务器之中，在任何给定的时间必定会有些服务器是不可用的。我们使用两条简单但是有效的策略保证整个系统的高可用性：快速恢复和复制。每个Chunk服务器都使用Checksum来检查保存的数据是否损坏。考虑到一个GFS集群通常都有好几百台机器、几千块硬盘，磁盘损坏导致数据在读写过程中损坏或者丢失是非常常见的（第7节讲了一个原因）。我们可以通过别的Chunk副本来解决数据损坏问题，但是跨越Chunk服务器比较副本来检查数据是否损坏很不实际。

**2.MapReduce**

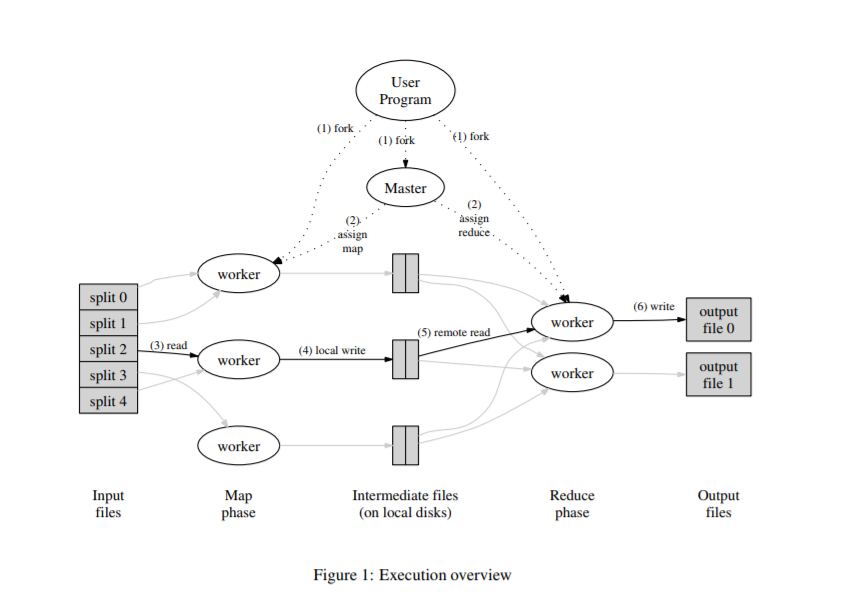
**1.编程模型**

计算过程接受一系列输入键值对，产生一系列输出键值对。MapReduce库的用户用两个函数来表示计算：Map和Reduce。Map函数由用户编写，接收一个输入键值对，产生一组中间键值对。MapReduce库组合所有的具有相同中间键的中间值并且将它们传给Reduce函数。Reduce函数也由用户编写，接收一个中间键和对应的一组值。它将其合并成一组可能规模更小的值。通常每个Reduce函数只会输出零个或一个值。中间值通过一个迭代器提供给用户的reduce函数。这允许我们操作过大以至于不能放在内存中的值列表。

**2.实现**

MapReduce可以有很多不同的实现接口，应当由具体环境来决定正确的实现。比如说，一种实现可能适用于一个小型共享内存机，另一种适用于一个大型NUMA多处理器系统，另一种可能适用于大型网络计算机集群。通过自动划分输入数据为M段来保证Map函数可以多机分布式调用。输入段可以被多机并行处理。通过使用划分函数划分中间键空间为R片，来保证Reduce函数可以分布式调用。片数R和划分函数由用户指定。

如Figure1所示，一个MapReduce操作在我们的实现上的整体运行流程。当用户程序调用MapReduce函数时，紧随其后将发生一系列操作。

(1)与用户程序一起执行的MapReduce库首先将输入文件分成M段，通常每段16MB到64MB。然后在集群电脑上启动程序的很多副本。

(2)有一个副本程序是独特的，称为主机。剩下的是由主机分配任务的工作机。有M个map任务和R个reduce任务需要被分配。主机会选择每个空闲工作机并分别分配一个map或reduce任务。

(3)一个被分配map任务的工作机读取对应输入段的内容。它分析出输入数据的键值对，将其传递给用户定义的Map函数。由后者生成中间键值对，并缓冲在内存中。

(4)每隔一段时间缓冲键值对会被写入本地磁盘，被划分函数划分成R个区域。这些缓冲键值对在磁盘上的位置被传回给主机，主机负责将这些位置发送reduce工作机。

(5)当一个reduce工作机收到主机传来的数据位置时，它使用远程过程调用（RPC）从map工作机本地的磁盘中读取缓冲数据。当一个reduce工作机读取完成所有中间数据后，它会根据中间键对它们进行排序，所有具有相同键的记录会被组合在一起。如果中间数据的总量太大而不能放入内存中，会使用外排序。

(6)reduce工作机在排好序的中间数据上重复工作，遍历每个遇到的独立的中间键。它把键和对应的中间值集合传递给用户的Reduce函数。Reduce函数的输出被添加到最终的输出文件中这个reduce的部分。

(7)当所有的map任务和reduce任务完成后，主机唤醒用户程序。此时，用户的MapReduce调用返回到用户代码中。

**3.BigTable**

**1.介绍**

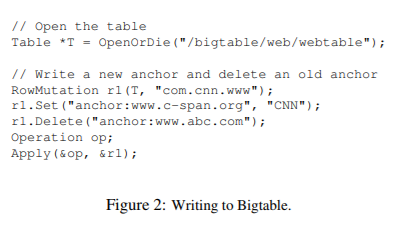
BigTable是Google公司设计、实现并部署了一个用于管理结构化数据的分布式的存储系统。BigTable区别于数据库的地方它提供了一个和这些系统完全不同的接口。数据用行和列的名字进行索引，名字可以是任意的字符串，BigTable为客户提供了简单的数据模型，利用这个模型，客户可以动态控制数据的布局和格式，用户也可以自己推测在底层存储中展示的数据的位置属性，参数动态地控制数据读或写。

**2.数据模型**

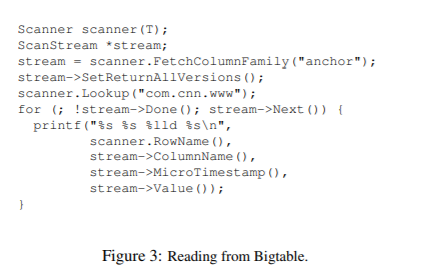
BigTable是一个稀疏的、分布式的、持久化存储的多维度排序Map。Map由行关键字、列关键字以及时间戳索引。Map中的每个value都是一个未经解析的字节数组。Bigtable通过行关键字的字典顺序来维护数据。表中一定范围内的行被动态分区。每个分区叫做一个”Tablet”。 列关键字组成的集合叫做“列族“，列族构成了访问控制的基本单位。列族创建后，其中的任何一个列关键字下都可以存放数据。访问控制、磁盘和内存的计数都是在列族层面进行的。在Bigtable中，每一个数据项都可以包含同一数据的不同版本，这些版本通过时间戳来索引。时间戳可由Bigtable或者该值由库户程序明确指定。需要避免冲突的程序必须自己生成一个唯一的时间戳。数据项中不同版本按照时间戳倒序排列，所以最新的版本可以被先读到。

**3.API**

Bigtable提供了建立和删除表以及列族的API函数。Bigtable还提供了修改集群、表和列族的元数据的API，比如修改访问权限.



如Figure2所示，此C++代码使用RowMutation抽象对象执行一系列的更新操作。



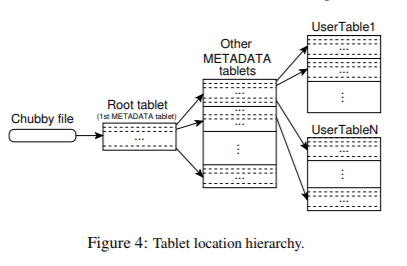
如Figure3所示，此C++代码使用Scanner抽象对象遍历一个特定行内的所有锚点。客户程序可以遍历多个列族，有几种机制可以对扫描输出的行、列和时间戳进行限制。如果限制上面的扫描，让它只输出那些列匹配正则表达式\*.cnn.com的锚点，或者那些时间戳在当前时间前10天的锚点。

**4.BigTable控件**

BigTable集群往往运行在一个共享的机器池中，池中的机器还会运行其它各种各样的分布式应用程序，BigTable的进程经常要和其它应用的进程共享机器。BigTable数据在内部使用SSTable文件格式存储。SSTable提供一个从键到值的持久化的、已排序、不可更改的映射，这里的key和value的都是任意的字节串。BigTable依赖一个称为Chubby高可用的、持久化的分布式锁服务组件。Chubby服务包括了5个活动的副本，其中一个副本被选为Master，并且积极处理请求。当有副本失效时，Chubby使用Paxos算法保证副本的一致性。

**5.实现**

BigTable的实现需要用到连接到每个客户程序的库、一个Master服务器和多个tablet服务器。在一个集群中可以动态地添加（或者删除）一个tablet服务器来适应工作负载的变化。

Master负责为tablet服务器分配tablets，检测新加入的或者过期失效的tablet服务器、平衡tablet服务器的负载、以及对GFS中的文件进行垃圾收集。

如Figure4所示，BigTable使用三层类似B+树的结构存储Tablet的位置信息。第一层是存储在Chubby File中的Root Tablet的位置，Root Tablet中一个特殊的METADATA存储了所有Tablet的位置信息，每个Tablet位置信息都存放在一个行关键字下。客户端使用的库会缓存Tablet的位置信息，当客户端需要访问某一个Tablet时发现地址信息为空或者不正确，需要在存储结构中进行搜索。为了减少搜索次数，每次读取Tablet的数据时，都会多读取几个Tablet。

Master负责探测一个tablet服务器何时不再为它的tablet提供服务，并且尽快重新分配那些tablet。master通过轮询tablet服务器锁的状态来探测tablet服务器何时不再为tablet提供服务。为了保证该tablet服务器不能再提供服，master就删除该tablet服务器在Chubby上的服务器文件。随着写操作的执行，memtable的大小不断增加，当到达一定大小之后会被冻结，然后创建一个新的memtable，这个memtable会转化为一个SSTable。当需要恢复一个Tablet的时候，从之日中读取该Tablet的所有记录重新构建Tablet。