分布式计算 第5次作业

姓名：张泽群 学号：19049100002 课程号：CS205105

1. 任务 一

说明三类分布式存储系统的区别：

(1) **块存储系统；** (2) **对象存储系统；** (3) **文件存储系统。**

1.1 应用角度上的区别

1.1.1 应用设备区别

【块存储】

典型设备：磁盘阵列，硬盘，虚拟硬盘

【文件存储】
典型设备：FTP、NFS服务器，SamBa

【对象存储】
典型设备：内置大容量硬盘的分布式服务器

1.1.2 性能容量区别

容量： 【块存储】<【文件存储】<【对象存储】

速度性能： 【块存储】>【文件存储】>【对象存储】

利用一张图片来直观显示：

1.1.3 访问方式区别

【块存储】： 通常都是通过光纤网络连接，服务器/小机上配置FC光纤HBA卡，通过光纤交换机连接存储（IP SAN可以通过千兆以太网，以iSCSI客户端连接存储），主机端以逻辑卷（Volume）的方式访问。连接成功后，应用访问存储是按起始地址，偏移量Offset的方法来访问的。

它的IO特点与传统的硬盘是一致的，一个硬盘应该是能面向通用需求的，即能应付大文件读写，也能处理好小文件读写。但是硬盘的特点是容量大，热点明显。因此块存储主要可以应付热点问题。另外，块存储要求的延迟是最低的。

【文件存储】： NAS文件存储通常只要是局域网内，千兆/百兆的以太网环境皆可。网线连上，服务器端通过操作系统内置的NAS客户端，如NFS/CIFS/FTP客户端挂载存储成为一个本地的文件夹后访问，只要符合POXIS标准，应用就可以用标准的open，seek, write/read,close这些方法对其访问操作。

【对象存储】： 对象存储不在乎网络，而且它的访问比较有特色，只能存取删（put/get/delete），不能打开修改存盘。只能取下来改好后上传，去覆盖原对象。

1.1.4 存储接口区别：

【块存储】：这种接口通常以QEMU Driver或者Kernel Module的方式存在，这种接口需要实现Linux的Block Device的接口或者QEMU提供的Block Driver接口，如Sheepdog，AWS的EBS，青云的云硬盘和阿里云的盘古系统，还有Ceph的RBD(RBD是Ceph面向块存储的接口)

【文件存储】：通常意义是支持POSIX接口，它跟传统的文件系统如Ext4是一个类型的，但区别在于分布式存储提供了并行化的能力，如Ceph的CephFS(CephFS是Ceph面向文件存储的接口)，但是有时候又会把GFS，HDFS这种非POSIX接口的类文件存储接口归入此类。

【对象存储】：也就是通常意义的键值存储，其接口就是简单的GET,PUT,DEL和其他扩展，如七牛、又拍，Swift，S3

1.1.5 适配场景区别

【块存储】

块存储（DAS/SAN）通常应用在某些**专有的系统**中，这类应用要求很高的随机读写性能和高可靠性，上面搭载的通常是Oracle/DB2这种传统数据库，连接通常是以FC光纤为主，走光纤协议。通常使用块存储的都是系统而非用户，并发访问不会很多，经常出现一套存储只服务一个应用系统，例如如交易系统，计费系统。典型行业如金融，制造，能源，电信等。

【文件存储】

文件存储（NAS）相对来说就更能**兼顾多个应用和更多用户访问**，同时提供方便的数据共享手段。毕竟大部分的用户数据都是以文件的形式存放，在PC时代，数据共享也大多是用文件的形式，比如常见的的FTP服务，NFS服务，Samba共享这些都是属于典型的文件存储。几十个用户甚至上百用户的文件存储共享访问都可以用NAS存储加以解决。在中小企业市场，一两台NAS存储设备就能支撑整个IT部门了。CRM系统，SCM系统，OA系统，邮件系统都可以使用NAS存储统统搞定。文件存储的广泛兼容性和易用性，是这类存储的突出特点。但是从性能上来看，相对SAN就要低一些。NAS存储基本上是以太网访问模式，普通千兆网，走NFS/CIFS协议。

【对象存储】

对象存储与云计算和大数据的概念相关，前面说到的块存储和文件存储，基本上都还是在专有的局域网络内部使用，而对象存储的**优势场景却是互联网或者公网**，主要解决海量数据，海量并发访问的需求。基于互联网的应用才是对象存储的主要适配，基本所有成熟的公有云都提供了对象存储产品。对象存储的访问通常是在互联网，走HTTP协议，性能方面，单独看一个连接的是不高的（还要解决掉线断点续传之类的可靠性问题），主要强大的地方是支持的并发数量，聚合起来的性能带宽就非常可观了。

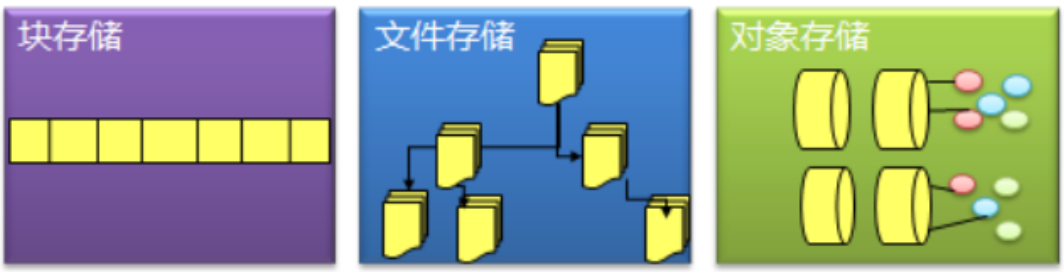
1.2 理论角度上的区别

1.2.1 组成结构区别

与块存储和文件存储管理数据的方式不同，对象存储是以**对象的形式**管理数据的。对象和文件最大的不同，就是在文件基础之上增加了元数据。一般情况下，对象分为三个部分：数据、元数据以及对象id。

对象的数据通常是无结构的数据，比如：图片、视频或文档等；对象的元数据则指的是对象的相关描述，比如：图片的大小、文档的拥有者等；对象id则是一个全局的唯一标识符，用来区分对象的。

从数据结构来看，这三种存储有着根本不同。块存储的数据结构是**数组**，而文件存储是**二叉树**（B,B-,B+,B*各种树），对象存储基本上都是**哈希表**。



1.2.2 实现方法区别

【块存储】： 块存储主要是将裸磁盘空间整个映射给主机，按照物理卷或逻辑卷的方式存储数据。特点：速度快、保存的数据尺寸大、数据变化频率高，缺点：可扩展程度小、容量小。（其实我们日常生活中见到的硬盘，就是块存储）

【文件存储】： 典型代表NAS。NAS是类似一个共享文件夹，用户可以通过相应的传输协议，把数据放在上面。容量相对较大，但是速度较慢。

【对象存储】： **对象存储同兼具SAN高速直接访问磁盘特点及NAS的分布式共享特点**，核心是将数据通路(数据读或写)和控制通路(元数据)分离在不同的节点上，多个节点应对多并发的访问，并且基于对象存储设备(OSD)，构建存储系统，每个对象存储设备具备一定的职能，能够自动管理其上的数据分布。

1.3 小结

	块存储	文件存储	对象存储
概念	用高速（光纤）网络联接专业主机服务器的一种储存方式	使用文件系统，具有目录树结构	将数据和元数据当做一个对象，
速度	低延迟(10ms)，热点突出	不同技术各有不同	100ms - 1s，冷数据
可分步性	异地不现实	可分布式，但有瓶颈	分步并发能力高
文件大小	大小都可以，热点突出	适合大文件	适合各种大小
接口	Driver , kernel module	POSIX	Restful API
典型技术	SAN	HDFS、GFS	Swift、Amazon S3
适合场景	银行	数据中心	网络媒体文件存储

2. 任务 二

阅读《The Hadoop Distributed File System》回答问题。

The Hadoop Distributed File System

Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler
Yahoo!
Sunnyvale, California USA
{Shv, Hairong, SRadia, Chansler}@Yahoo-Inc.com

***Abstract*—The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size. We describe the architecture of HDFS and report on experience using HDFS to manage 25 petabytes of enterprise data at Yahoo!.**

Keywords: Hadoop, HDFS, distributed file system

I. Introduction and Related Work

Hadoop [1][16][19] provides a distributed file system and a framework for the analysis and transformation of very large data sets using the MapReduce [3] paradigm. An important characteristic of Hadoop is the partitioning of data and computation across many (thousands) of hosts, and executing application computations in parallel close to their data. A Hadoop cluster scales computation capacity, storage capacity and IO bandwidth by simply adding commodity servers. Hadoop clusters at Yahoo! span 25 000 servers, and store 25 petabytes of application data, with the largest cluster being 3500 servers. One hundred other organizations worldwide report using Hadoop.

HDFS	Distributed file system Subject of this paper!
MapReduce	Distributed computation framework
HBase	Column-oriented table service
Pig	Dataflow language and parallel execution framework
Hive	Data warehouse infrastructure
ZooKeeper	Distributed coordination service
Chukwa	System for collecting management data
Avro	Data serialization system

Table 1. Hadoop project components

Hadoop is an Apache project; all components are available via the Apache open source license. Yahoo! has developed and contributed to 80% of the core of Hadoop (HDFS and MapReduce). HBase was originally developed at Powerset, now a department at Microsoft. Hive [15] was originated and developed at Facebook. Pig [4], ZooKeeper [6], and Chukwa were originated and developed at Yahoo! Avro was originated at Yahoo! and is being co-developed with Cloudera.

HDFS is the file system component of Hadoop. While the interface to HDFS is patterned after the UNIX file system, faithfulness to standards was sacrificed in favor of improved performance for the applications at hand.

HDFS stores file system metadata and application data separately. As in other distributed file systems, like PVFS [2][14], Lustre [7] and GFS [5][8], HDFS stores metadata on a dedicated server, called the NameNode. Application data are stored on other servers called DataNodes. All servers are fully connected and communicate with each other using TCP-based protocols.

Unlike Lustre and PVFS, the DataNodes in HDFS do not use data protection mechanisms such as RAID to make the data durable. Instead, like GFS, the file content is replicated on multiple DataNodes for reliability. While ensuring data durability, this strategy has the added advantage that data transfer bandwidth is multiplied, and there are more opportunities for locating computation near the needed data.

Several distributed file systems have or are exploring truly distributed implementations of the namespace. Ceph [17] has a cluster of namespace servers (MDS) and uses a dynamic subtree partitioning algorithm in order to map the namespace tree to MDSs evenly. GFS is also evolving into a distributed namespace implementation [8]. The new GFS will have hundreds of namespace servers (masters) with 100 million files per master. Lustre [7] has an implementation of clustered namespace on its roadmap for Lustre 2.2 release. The intent is to stripe a directory over multiple metadata servers (MDS), each of which contains a disjoint portion of the namespace. A file is assigned to a particular MDS using a hash function on the file name.

II. Architecture

A. NameNode

The HDFS namespace is a hierarchy of files and directories. Files and directories are represented on the NameNode by *inodes*, which record attributes like permissions, modification and access times, namespace and disk space quotas. The file content is split into large blocks (typically 128 megabytes, but user selectable file-by-file) and each block of the file is independently replicated at multiple DataNodes (typically three, but user selectable file-by-file). The NameNode maintains the namespace tree and the mapping of file blocks to DataNodes

978-1-4244-7153-9/10/$26.00 ©2010 IEEE

① 客户端读取HDFS系统中指定文件指定偏移量处的数据时，工作流程是什么?

答： 原文： When a client opens a file to read, it fetches the list of blocks and the locations of each block replica from the NameNode. The locations of each block are ordered by their distance from the reader. When reading the content of a block, the client tries the closest replica first. If the read attempt fails, the client tries the next replica in sequence.

当客户端打开文件并读取，它会从NameNode获取块列表和每个块副本的位置。位置信息会根据离reader的距离的顺序排序。当读取一个块的内容，客户端会尝试从最近的副本读。如果失败了，就读取序列中的下一个副本。

工作流程：

(1). 客户端向NameNode发出数据访问请求，询问NameNode它应该从哪里读取目标文件。 (2). Namenode会根据目标文件路径收集所有数据块（block）的位置信息，并根据数据块在文件中的先后顺序，按次序组成数据块定位集合（located blocks），回应给客户端 (3). 客户端检查数据块定位信息，创建HDFS输入流，按照就近距离的顺序，根据读取偏移量定位，联系相关的DataNode，请求数据块。 (4). DataNode返回文件内容给客户端，然后关闭连接，完成读操作。

② 客户端向HDFS系统中指定文件追加写入数据的工作流程是什么?

答： 原文中似乎并没有具体描述追加数据写入，我通过查阅资料完成该题。

原文： After the file is closed, the bytes writ ten cannot be altered or removed except that new data can be added to the file by reopening the file for append. HDFS im plements a single-writer, multiple-reader model.

关闭文件后，无法更改或删除写入的字节，除非可以通过重新打开文件进行追加来向文件中添加新数据。HDFS实现了一个单写多读模型。

An HDFS file consists of blocks. When there is a need for a new block, the NameNode allocates a block with a unique block ID and determines a list of DataNodes to host replicas of the block. The DataNodes form a pipeline, the order of which minimizes the total network distance from the client to the last DataNode. Bytes are pushed to the pipeline as a sequence of packets. The bytes that an application writes first buffer at the client side. After a packet buffer is filled (typically 64 KB), the data are pushed to the pipeline. The next packet can be pushed to the pipeline before receiving the acknowledgement for the previous packets. The number of outstanding packets is limited by the outstanding packets window size of the client.

HDFS文件由块组成。当需要一个新块时，NameNode会分配一个具有唯一块ID的块，并确定一个DataNode列表，以承载该块的副本。数据节点形成一个管道，其顺序使从客户端到最后一个数据节点的总网络距离最小化。字节作为数据包序列推送到管道。应用程序在客户端写入第一个缓冲

区的字节。数据包缓冲区填满后（通常为64 KB），数据被推送到管道。在接收到之前数据包的确认之前，可以将下一个数据包推送到管道。未完成数据包的数量受到客户端未完成数据包窗口大小的限制。

(1). 打开已有的HDFS文件，判断数据块是否写满。

客户端首先打开目标的HDFS文件，并获取文件最后一个数据块的位置信息，判断数据块是否写满。

(2). 如果文件最后一个数据块没有写满，则该数据块的位置信息建立到该数据块的数据流管道；如果文件的最后一个数据块已经写满，申请一个新的空的数据块。

(3). 创建到这个数据块的输出流对象和数据流管道，获取文件租约，开始新一轮文件写入的过程。

(4). 文件写入过程：1. 在NameNode修改文件元数据，2. 通过数据流管道写入数据，3. 按照原文下划线部分的方式向DataNode写入数据包，4. 客户端接受确认包

(5). 关闭输入流并提交文件，完成写操作

③ 新增加一个数据块时，HDFS如何选择存储该数据块的物理节点?

答： 选择块放置策略应该最小化写入成本和最大化数据可靠性、可用性和聚合读取带宽之间进行权衡。并且需要将单个文件的块副本尽可能分布到整个集群。策略可以总结如下

- 1.没有数据节点包含任何块的多个副本。
- 2.如果集群上有足够的机架，则机架中包含的同一块的副本不会超过两个。

具体情况应该具体分析，如原文中所示：

原文： The default HDFS block placement policy provides a tradeoff between minimizing the write cost, and maximizing data reliability, availability and aggregate read bandwidth.

When a new block is created, HDFS places the first replica on the node where the writer is located, the second and the third replicas on two different nodes in a different rack, and the rest are placed on random nodes with restrictions that no more than one replica is placed at one node and no more than two replicas are placed in the same rack when the number of replicas is less than twice the number of racks. The choice to place the second and third replicas on a different rack better distributes the block replicas for a single file across the cluster. If the first two replicas were placed on the same rack, for any file, two-thirds of its block replicas would be on the same rack.

当一个新块被创建，HDFS把第一份副本存放到writer所在的节点，第二和第三份存放到不同机架的不同节点，当副本数小于两倍机架数时，每个节点上的副本不多于1份，每个机架上的副本不多于2份。把第二第三份副本放到不同机架能够在集群中更好地分发单个文件的块副本。如果任意文件的头两份副本放在同一个机架，2/3的块副本会在同一个机架上。

④ HDFS采用了哪些措施应对数据块损坏或丢失问题?

答： 1.保证软件版本的一致性来预防数据损坏丢失问题

原文： During startup each DataNode connects to the NameNode and performs a handshake. The purpose of the handshake is to verify the namespace ID and the software version of the DataNode. If either does not match that of the NameNode the DataNode automatically shuts down.

在启动期间，每个DataNode连接到NameNode并执行握手。握手的目的是验证名称空间ID和DataNode的软件版本。如果其中一个与NameNode不匹配，DataNode将自动关闭。

The consistency of software versions is important because incompatible version may cause data corruption or loss, and on large clusters of thousands of machines it is easy to overlook nodes that did not shut down properly prior to the software upgrade or were not available during the upgrade.

软件版本的一致性很重要，因为不兼容的版本可能会导致数据损坏或丢失，在数千台计算机的大型集群上，很容易忽略在软件升级之前未正确关闭或升级期间不可用的节点。

2.利用block scanner扫描校验和来检测坏块，并重新复制坏块的副本

HDFS为HDFS文件的每个数据块生成和存储校验和。HDFS客户端在读取时验证校验和，以帮助检测由客户端、数据节点或网络导致的任何损坏。如果损坏，客户端将通知NameNode损坏的副本，然后从另一个DataNode获取块的不同副本。

原文： Each DataNode runs a block scanner that periodically scans its block replicas and verifies that stored checksums match the block data.

每个DataNode上都允许了一个block scanner，定期地扫描块副本并验证相关的校验和。

原文： Whenever a read client or a block scanner detects a corrupt block, it notifies the NameNode. The NameNode marks the replica as corrupt, but does not schedule deletion of the replica immediately. Instead, it starts to replicate a good copy of the block. Only when the good replica count reaches the replication factor of the block the corrupt replica is scheduled to be removed. This policy aims to preserve data as long as possible.So even if all replicas of a block are corrupt, the policy allows the user to retrieve its data from the corrupt replicas. 当读客户端或

block scanner检测到坏块时，会通知NameNode。NameNode将块标记为损坏，但是不会马上安排删除副本，而是复制出一个好的副本。仅在好副本数达到复制因子，损坏的副本才会被安排删除。这个策略旨在尽可能久地保存数据。所以如果所有的副本都损坏了，该策略允许用户从损坏的副本中读取数据。

3.**冗余备份：** 数据存储在这些HDFS中的节点上，为了防止因为某个节点宕机而导致数据丢失，HDFS对数据进行冗余备份。 4.**块放置策略：** 仅仅对数据进行冗余备份还不够，假设所有的备份都在一个节点上，那么该节点宕机后，数据一样会丢失，因此Hadoop设计快放置策略使得块副本尽可能分布到整个集群，从而防止一个交换机或机架的损坏影响全局。

⑤ HDFS采用了什么措施应对主节点失效问题?

答： 引入BackupNode将核心文件备份

原文： If the NameNode fails, the BackupNode's image in memory and the checkpoint on disk is a record of the latest namespace state. 如果NameNode发生了故障，BackupNode内存中的image和磁盘上的checkpoint就是最新的命名空间状态。 NameNode保存了所有的元数据信息，其中，最核心的两大数据结构是**映像文件和事务日志**，如果这两个文件发生损坏，那么整个HDFS实例将失效。因此，HDFS设置了备份机制，把这些核心文件同步复制到备份服务器**BackupNode**上。当NameNode失效时，就可以根据备份服务器**BackupNode**的检查点对两个文件进行恢复，从而恢复主节点状态。

⑥ NameNode维护的"数据块—物理节点对应表"需不需要在硬盘中备份？为什么?

答： 不需要。NameNode可以由BackupNode等手段进行恢复，其中的"数据块—物理节点对应表"也会恢复到对应检查点的状态。

原文： HDFS keeps the entire namespace in RAM. The inode data and the list of blocks belonging to each file comprise the meta data of the name system called the image. The persistent record of the image stored in the local host's native files system is called a checkpoint. The NameNode also stores the modification log of the image called the journal in the local host's native file system. **For improved durability, redundant copies of the checkpoint and journal can be made at other servers. During restarts the NameNode restores the namespace by reading the namespace and replaying the journal.** 为了提高耐用性，可以在其他服务器上制作检查点和日志的冗余副本。在重新启动期间，NameNode通过读取名称空间并重播日志来恢复名称空间。即命名空间，包括"数据块—物理节点对应表"可以通过其他服务器上的检查点和日志的冗余副本来恢复。