# Taobao File System

TFS为淘宝提供海量小文件存储，通常文件大小不超过1M，满足了淘宝对小文件存储的需求，被广泛地应用在淘宝各项应用中。它采用了HA（High Available，高可用性集群，双机集群，一般有主备节点）架构和平滑扩容，保证了整个文件系统的可用性和扩展性。同时扁平化的数据组织结构，可将文件名映射到文件的物理地址，简化了文件的访问流程，一定程度上为TFS提供了良好的读写性能。

TFS架构设计时需要考虑如下两个问题：

Metadata信息存储。假设每个文件的元数据100B，100亿图片的元数据需要1TB，单台机器无法提供元数据服务；

减少图片的IO次数：普通Linux文件系统中，读取一个文件需要三次磁盘IO，先读取目录元数据到内存，再把文件的inode节点读入内存，最后读取文件数据到内存。**由于小文件数量太多，无法把所有的目录及文件的inode信息缓存到内存中，因此磁盘IO次数很难达到每个图片读取只需要一次IO理想状态。**

TFS设计时采用的思路：多个逻辑图片文件共享一个物理文件。

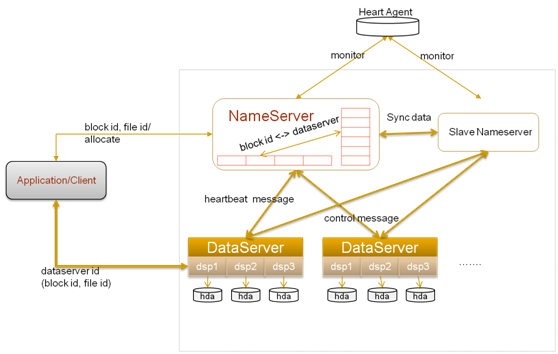
## TFS总体结构

一个TFS集群由两个NameServer节点（一主一备）和多个DataServer节点组成。这些服务程序都是作为一个用户级的程序运行在普通Linux机器上的。在TFS中，将大量的小文件(实际数据文件)合并成为一个大文件，这个大文件称为块(Block), 每个Block拥有在集群内唯一的编号(Block Id), Block Id在NameServer创建Block的时候分配, NameServer维护Block与DataServer的关系。Block中的实际数据都存储在DataServer上。而一台DataServer服务器一般会有多个独立DataServer进程存在，每个进程负责管理一个挂载点，这个挂载点一般是一个独立磁盘上的文件目录，以降低单个磁盘损坏带来的影响。

NameServer主要功能是: 管理维护Block和DataServer相关信息,包括DataServer加入、退出，心跳信息，Block和DataServer的对应关系建立、解除。正常情况下，一个块会在DataServer上存在，主NameServer负责Block的创建、删除、复制、均衡、整理。NameServer不负责实际数据的读写，实际数据的读写由DataServer完成。

DataServer主要功能是: 负责实际数据的存储和读写。

为了考虑容灾，NameServer采用了HA结构，即两台机器互为热备，同时运行，一台为主，一台为备，主机绑定到对外vip（Virtual IP），提供服务；当主机器宕机后，迅速将vip绑定至备份NameServer，将其切换为主机，对外提供服务。通过心跳机制实现故障检测。



TFS内部**不维护文件目录树**，每个**小文件用64位的编号表示**。TFS将大量的小文件合并成一个大文件，大文件称为Block，每个Block拥有在集群内唯一的编号（块ID），通过<块ID，文件编号>可以唯一确定一个文件。Block大小一般为64MB，在DataServer存储三份。DataServer进程会给Block中的每个文件分配一个ID(File ID，该ID在每个Block中唯一)，并将**每个文件在Block中的信息存放在和Block对应的Index文件中**。这个Index文件一般都会全部load在内存，除非出现DataServer服务器内存和集群中所存放文件平均大小不匹配的情况。

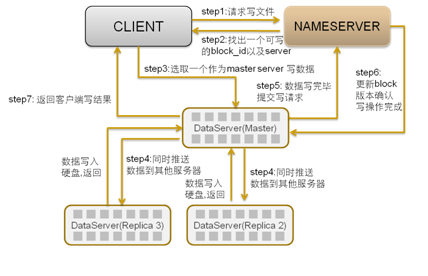
另外，还可以部署一个对等的TFS集群，作为当前集群的辅集群。辅集群不提供来自应用的写入，只接受来自主集群的写入。当前主集群的每个数据变更操作都会重放至辅集群。辅集群也可以提供对外的读，并且在主集群出现故障的时候，可以接管主集群的工作。

客户端不缓存文件数据，只缓存NameServer的元数据。

## 写流程

TFS的追加流程比GFS简单有效（原因是根据所要服务业务的特点，一个分布式系统的设计是根据自己业务需求不断改进得到的），GFS为了减少Master的压力，引入了租约机制，将修改权限下放到主chunkserver，很多的追加操作不需要Master参与。TFS是写少读多的，即使每次写操作要经过NameNode也没关系，这样才大大简化了系统的设计。TFS不支持类似GFS多客户端并发追加操作，同一时刻每个block只能有一个写操作，多个客户端写操作会被串行化。

在TFS系统中，NameServer会保证一个文件有多个副本存储于不同的DataServer上以保证冗余。当由于DataServer服务器宕机或者由于其他原因退出系统导致某些文件副本数量下降时，NameServer将会调度新的DataServer节点存储文件备份。同样为了保证数据一致性，当写入一个文件时，只有所有参与的dataserver均写入成功时，该操作才算成功。TFS的写操作数据流图如下所示：



客户端首先向nameserver发起写请求，nameserver需要根据dataserver上的可写块，容量和负载加权平均来选择一个可写的Block。并且在该Block所在的多个dataserver中选择一个作为写入的master，这个选择过程也需要根据dataserver的负载以及当前作为master的次数来计算，使得每个dataserver作为master的机会均等。**master一旦选定，除非master宕机，不会更换**，一旦master宕机，需要在剩余的dataserver中选择新的master。返回一个dataserver列表。

客户端向master dataserver开始数据写入操作。master server将数据传输为其他的dataserver节点，只有当所有dataserver节点写入均成功时，master server才会向nameserver和客户端返回操作成功的信息。

相比GFS，TFS的写流程不够优化。

1. 每个写请求都需要多次访问NameServer；
2. DataServer之间数据的推送没有采用流水线方式减少延迟。

写操作返回后，会返回客户端两个信息，小文件在TFS中的block编号以及文件编号。系统将这些信息保存在数据库中。读文件时，首先根据block编号从NameServer查找block所在的DataServer，然后根据文件编号读取图片数据。

## 读流程

1. **获得Block ID和File ID**

根据TFS文件名解析出Block ID和Block中的File ID.

1. **获取dataserver地址**

向nameserver发送查询请求得到Block ID所在的dataserver地址。

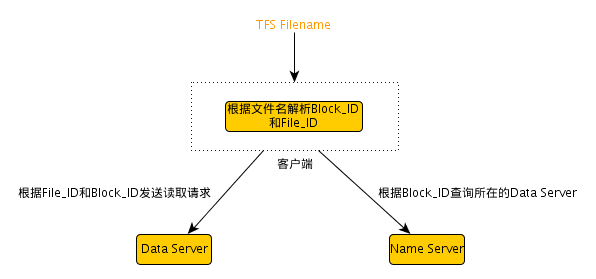
由于nameserver中维护了Block和dataserver的对应关系，所以nameserver能够提供相应的信息。

Note: 由于TFS是把大量小文件放在一个Block里面，所以TFS的文件复制是基于Block的，而且复制出来的Block的block id应该是一致的。

1. **请求文件**

通过发送Block\_ID、File\_ID和offset为参数的读请求到对应的dataserver，得到文件内容。

dataserver会根据本地记录的信息来得到File ID所在block的偏移量，从而读取到正确的文件内容.



## 平滑扩容

原有TFS集群运行一定时间后，集群容量不足，此时需要对TFS集群扩容。由于DataServer与NameServer之间使用心跳机制通信，如果系统扩容，只需要将相应数量的新DataServer服务器部署好应用程序后启动即可。这些DataServer服务器会向NameServer进行心跳汇报。NameServer会根据DataServer容量的比率和DataServer的负载决定新数据写往哪台DataServer的服务器。根据写入策略，容量较小，负载较轻的服务器新数据写入的概率会比较高。同时，在集群负载比较轻的时候，NameServer会对DataServer上的Block进行均衡，使所有DataServer的容量尽早达到均衡。

进行均衡计划时，首先计算每台机器应拥有的blocks平均数量，然后将机器划分为两堆，一堆是超过平均数量的，作为移动源；一类是低于平均数量的，作为移动目的。

移动目的的选择：首先一个block的移动的源和目的，应该保持在同一网段内，也就是要与另外的block不同网段；另外，在作为目的的一定机器内，优先选择同机器的源到目的之间移动，也就是同台DataServer服务器中的不同DataServer进程。

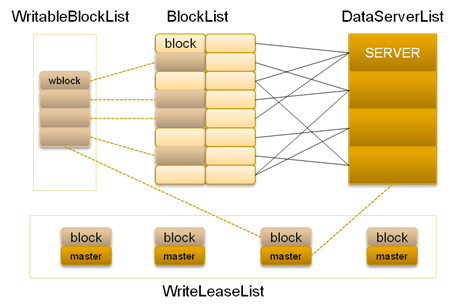
当有服务器故障或者下线退出时（单个集群内的不同网段机器不能同时退出），不影响TFS的服务。此时NameServer会检测到备份数减少的block，对这些block重新进行数据复制。

在创建复制计划时，一次要复制多个block, 每个block的复制源和目的都要尽可能的不同，并且保证每个block在不同的子网段内。因此采用轮换选择(roundrobin)算法，并结合加权平均。

由于DataServer之间的通信是主要发生在数据写入转发的时候和数据复制的时候，集群扩容基本没有影响。假设一个Block为64M，数量级为1PB。那么NameServer上会有 1 \* 1024 \* 1024 \* 1024 / 64 = 16.7M个block。假设每个Block的元数据大小为0.1K，则占用内存不到2G。

## 存储机制

在TFS中，将大量的小文件(实际用户文件)合并成为一个大文件，这个大文件称为块(Block)。TFS以Block的方式组织文件的存储。每一个Block在整个集群内拥有唯一的编号，这个编号是由NameServer进行分配的，而DataServer上实际存储了该Block。在NameServer节点中存储了所有的Block的信息，一个Block存储于多个DataServer中以保证数据的冗余。对于数据读写请求，均先由NameServer选择合适的DataServer节点返回给客户端，再在对应的DataServer节点上进行数据操作。NameServer需要维护Block信息列表，以及Block与DataServer之间的映射关系，其存储的元数据结构如下：



在DataServer节点上，在挂载目录上会有很多物理块，物理块以文件的形式存在磁盘上，并在DataServer部署前预先分配，以保证后续的访问速度和减少碎片产生。为了满足这个特性，DataServer现一般在EXT4文件系统上运行。物理块分为主块和扩展块，一般主块的大小会远大于扩展块，使用扩展块是为了满足文件更新操作时文件大小的变化。每个Block在文件系统上以“主块+扩展块”的方式存储。每一个Block可能对应于多个物理块，其中包括一个主块，多个扩展块。

在DataServer端，每个Block可能会有多个实际的物理文件组成：一个主Physical Block文件，N个扩展Physical Block文件和一个与该Block对应的索引文件。Block中的每个小文件会用一个Block内唯一的fileid来标识。DataServer会在启动的时候把自身所拥有的Block和对应的Index加载进来。

## 容错

**集群容错**

TFS可以配置主辅集群，一般主辅集群会存放在两个不同的机房。主集群提供所有功能，辅集群只提供读。主集群会把所有操作重放到辅集群。这样既提供了负载均衡，又可以在主集群机房出现异常的情况不会中断服务或者丢失数据。

**NameServer容错**

Namserver主要管理了DataServer和Block之间的关系。如每个DataServer拥有哪些Block，每个Block存放在哪些DataServer上等。同时，NameServer采用了HA结构，一主一备，主NameServer上的操作会重放至备NameServer。如果主NameServer出现问题，可以实时切换到备NameServer。另外NameServer和DataServer之间也会有定时的heartbeat，DataServer会把自己拥有的Block发送给NameServer。NameServer会根据这些信息重建DataServer和Block的关系。

**DataServer容错**

TFS采用Block存储多份的方式来实现DataServer的容错。每一个Block会在TFS中存在多份，一般为3份，并且分布在不同网段的不同DataServer上。对于每一个写入请求，必须在所有的Block写入成功才算成功。当出现磁盘损坏DataServer宕机的时候，TFS启动复制流程，把备份数未达到最小备份数的Block尽快复制到其他DataServer上去。TFS对每一个文件会记录校验crc，当客户端发现crc和文件内容不匹配时，会自动切换到一个好的block上读取。此后客户端将会实现自动修复单个文件损坏的情况。

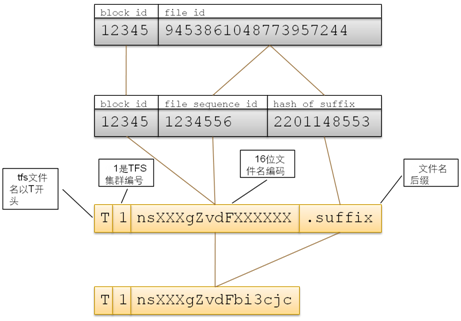
## 并发

对于同一个文件来说，多个用户可以并发读。

现有TFS并不支持并发写一个文件。一个文件只会有一个用户在写。这在TFS的设计里面对应着是一个block同时只能有一个写或者更新操作。

## TFS文件名结构

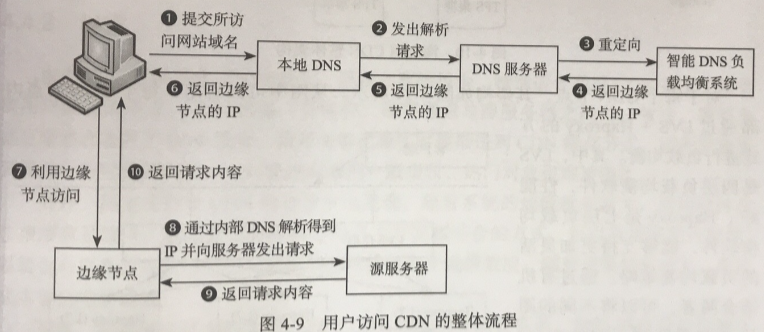
TFS的文件名由块号和文件号通过某种对应关系组成，最大长度为18字节。文件名固定以T开始，第二字节为该集群的编号(可以在配置项中指定，取值范围 1-9)。余下的字节由Block ID和File ID通过一定的编码方式得到。文件名由客户端程序进行编码和解码，它映射方式如下图：



TFS客户程序在读文件的时候通过将文件名转换为BlockID和FileID信息，然后可以在NameServer取得该块所在DataServer信息（如果客户端有该block与DataServere的缓存，则直接从缓存中取），然后与DataServer进行读取操作。

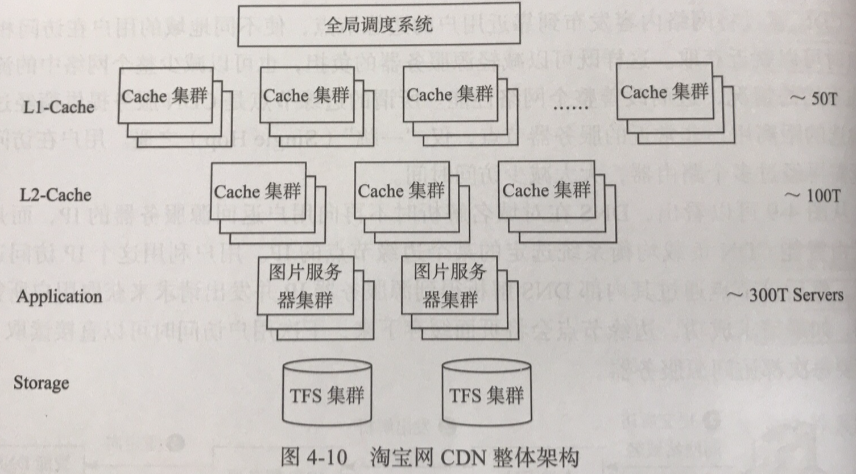
# 内容分发网络

CDN（Content Delivery Network）通过将网络内容发布到靠近用户的边缘节点，使不同区域的用户在访问相同网页时可以就近获取。边缘节点是CDN服务提供商经过挑选出来的距离用户非常近的服务器节点，用户访问时无需经过多个路由，大大减少访问时间。



客户端通过智能DNS负载均衡系统得到边缘节点的地址后访问边缘节点，边缘节点向服务器请求数据，如果请求成功，边缘节点会将页面缓存下来，下次用户访问时可以直接从边缘节点读取页面内容，不需要每次都访问服务器。

## 淘宝CND架构



淘宝CDN采用两级Cache，用户访问淘宝网的图片时，通过全局调度系统调度到某个L1-cache节点，如果L1命中，直接将图片数据返回；否则请求L2-cache节点，并将返回的图片缓存到L1-cache，如果L2-cache命中，直接将图片数据返回给L1-cache，否则请求源服务器的图片服务器集群。每台图片服务器是一个运行Nginx的web服务器，它会本地缓存图片，只有当在图片服务器本地缓存不命中的情况下才会请求TFS集群。图片服务器集群和TFS集群部署在同一个数据中心内。