Opendfs是一个用C/C++编写的分布式文件存储系统，它具有高度容错、高并发、高吞吐量、易扩展，具有类似Linux文件系统的文件、目录结构等特点；

类似于HDFS，一个Opendfs集群也由DFSClient、Namenode、Datanode三种角色组成，一个文件将由DFSClient切分成多个数据块存储到集群上，而Datanode则负责保存这些数据块，Namenode负责维护文件由多少个块组成，这些块保存在哪个Datanode上的映射关系，集群整体架构如图1示：



图1

各角色功能介绍如下：

1. Namenode，元数据存储节点，负责元数据的管理，包括文件、目录树在集群中的存储结构，每个文件对应的数据块列表，各数据块存储在哪些数据节点上；管理数据节点的存活状态以及数据流的读写调度等；每次元数据的操作（写、删除）除了更新内存，还会产生一条操作日志写到本地磁盘的日志文件（editlog），在写入磁盘前这条操作日志会通过paxos协议同步到其他从元数据节点，在某个设定时间点（checkpoint），每个元数据节点会把内存中的元数据dump到本地磁盘生成镜像文件（fsimage），以确保editlog文件不至于过大，元数据节点的每次启动，都会通过fsimage和editlog文件动态的重建元数据的映射关系。
2. Datanode，数据存储节点，负责文件数据块的管理，每个数据块将以文件的形式存储到本地文件系统，接收来自客户端的数据块读写请求；启动时会注册到所有元数据节点，通过心跳、块上报随时向元数据节点汇报自己的健康状态，如：剩余容量大小、当前处理的连接数、当前存储的块数以及数据块是否被损坏等，同时接收来自元数据节点的指令，如创建、移动、删除数据块等。
3. DFSClient，文件读写终端，是用户与集群交互的重要手段，提供类似Linux命令（ls、rm、rmr…）及Posix接口等，负责将文件切分成多个块往数据节点写；通过读取来自各个数据节点的数据块，合成一个完整的文件给应用层返回。

功能列表：

1. 存储于Opendfs中的文件，其元信息及文件数据会分别存储，与HDFS不同的是，存储元信息的Namenode，可以是多台主、从热备的架构，其中主、从间通过Paxos协议同步保证元数据的最终一致性，在网络模型处理中，HDFS是一个连接在一个线程里完成，而Opendfs是采用epoll事件驱动模型，从而提高网络的并发访问效率。
2. 文件在集群中将被按一定大小切分成多个数据块（每块默认256M，可配置），分散存储在各个Datanode上，每个数据块通过副本冗余存储于多台Datanode上，通过自动修复损坏、丢失的副本，从而保证数据块的可靠性，在HDFS中，Datanode接收到一个数据块的读或写请求都在一个线程里完成，而在Opendfs里，Datanode将会把一个读或写请求所涉及的网络IO和磁盘IO，分配到不同的线程池里，协同完成，通过零拷贝技术（sendfile）数据块将从磁盘直接拷贝到网络返回给DFSClient，减少用户态和内核态上下文的切换，从而提高数据的传输效率。
3. 在HDFS中，DFSClient往Datanode写数据块的时候，是先把从Namenode要回的Datanode列表建立起一个管道，如：Datanode1 -> Datanode2 -> Datanode3，再往管道里串行的写数据块，而在Opendfs中，数据块将并行的从DFSClient往所有Datanode写。

一个简单的文件写流程，如图2示：



图2

1. DFSClient向主Namenode发起写文件请求，并提供数据块大小、所需副本数（默认为3）等信息；
2. Namenode根据请求的副本数返回可写的Datanode列表（IPs），根据块分布策略使得返回的IPs分布在不同的机架，且每个IP只存一份副本；
3. DFSClient根据返回的IPs并行的向各个Datanode写数据块；
4. Datanode1收到数据块写请求，写入本地磁盘，成功写完一个块后再向所有Namenode上报；
5. Datanode2收到数据块写请求，写入本地磁盘，成功写完一个块后再向所有Namenode上报；
6. Datanode3收到数据块写请求，写入本地磁盘，成功写完一个块后再向所有Namenode上报；因为DFSClient是并行的写数据块，所以步骤4、5、6不是严格意义上的串行关系，而是并行的；
7. Datanode向Namenode上报收到块后，再告知DFSClient数据块成功写完；
8. 至此一个数据块完整写完，如果需要写下一个数据块，则重复步骤1，直至所有数据块写完，再向Namenode发起关闭文件请求。

一个简单的文件读流程，如图3示：



图3

1. DFSClient向任意一Namenode发起读文件请求，问当前要读取的文件数据块分布在那些数据节点上；
2. Namenode返回可读的Datanode列表（IPs），每个数据块都返回所有3副本的IP，且一次请求尽可能返回多个数据块的IP列表（默认为10个块）；
3. DFSClient根据返回的IPs，先向IP1发起读数据块请求，如果失败，则从IPs列表中剔除，从而向IP2发起请求，直至数据块成功读取，如果向列表中所有IP发起请求均失败，则向Namenode上报Datanode不可用，然后告知应用层由于块缺失而导致文件读取失败；
4. 同步骤3依次读取文件的下一个块；
5. 同步骤3依次读取文件的下一个块，直至读取文件的所有块。

一个快速的入门例子：

1. 安装
2. 环境依赖：

Linux、GCC4.8+

1. 软件依赖：

yum -y install pcre-devel

1. 源码编译：

./configure --prefix=/home/opendfs

make

make install

1. 配置
2. 单机配置，即DFSClient、Namenode、Datanode都跑在同一台机器上，且Namenode、Datanode均为单点，各角色配置如下：

|  |
| --- |
| # dfscli.conf  Server server;  server.daemon = ALLOW;  server.namenode\_addr = "127.0.0.1:8000";  server.error\_log = "";  server.log\_level = LOG\_INFO;  server.recv\_buff\_len = 64KB;  server.send\_buff\_len = 64KB;  server.blk\_sz = 256MB;  server.blk\_rep = 3; |

|  |
| --- |
| # namenode.conf  Server server;  server.daemon = ALLOW;  server.workers = 8;  server.connections = 65536;  server.bind\_for\_cli = "0.0.0.0:8000";  server.bind\_for\_dn = "0.0.0.0:8001";  server.my\_paxos = "0.0.0.0:8002";  server.ot\_paxos = "0.0.0.0:8002";  server.paxos\_group\_num = 1;  server.checkpoint\_num = 10000;  server.index\_num = 1000000;  server.editlog\_dir = "/data00/namenode/editlog";  server.fsimage\_dir = "/data00/namenode/fsimage";  server.error\_log = "/data00/namenode/logs/error.log";  server.pid\_file = "/data00/namenode/pid/namenode.pid";  server.coredump\_dir = "/data00/namenode/coredump";  server.log\_level = LOG\_INFO;  server.recv\_buff\_len = 64KB;  server.send\_buff\_len = 64KB;  server.max\_tqueue\_len = 1000;  server.dn\_timeout = 600; |

|  |
| --- |
| # datanode.conf  Server server;  server.daemon = ALLOW;  server.workers = 8;  server.connections = 65536;  server.bind\_for\_cli = "0.0.0.0:8100";  server.ns\_srv = "127.0.0.1:8001";  server.data\_dir = "/data01/block,/data02/block,/data03/block";  server.error\_log = "/data00/datanode/logs/error.log";  server.pid\_file = "/data00/datanode/pid/datanode.pid";  server.coredump\_dir = "/data00/datanode/coredump/";  server.log\_level = LOG\_INFO;  server.recv\_buff\_len = 64KB;  server.send\_buff\_len = 64KB;  server.max\_tqueue\_len = 1000;  server.heartbeat\_interval = 3;  server.block\_report\_interval = 3600; |

1. 集群配置，其中DFSClient一台，Namenode、Datanode均为3台，各角色配置如下：

|  |
| --- |
| # dfscli.conf  Server server;  server.daemon = ALLOW;  server.namenode\_addr = "192.168.1.1:8000";  server.error\_log = "";  server.log\_level = LOG\_INFO;  server.recv\_buff\_len = 64KB;  server.send\_buff\_len = 64KB;  server.blk\_sz = 256MB;  server.blk\_rep = 3; |

|  |
| --- |
| # namenode.conf  # namenode1  Server server;  server.daemon = ALLOW;  server.workers = 8;  server.connections = 65536;  server.bind\_for\_cli = "192.168.1.1:8000";  server.bind\_for\_dn = "192.168.1.1:8001";  server.my\_paxos = "192.168.1.1:8002";  server.ot\_paxos = "192.168.1.1:8002, 192.168.1.2:8002, 192.168.1.3:8002";  server.paxos\_group\_num = 1;  server.checkpoint\_num = 10000;  server.index\_num = 1000000;  server.editlog\_dir = "/data00/namenode/editlog";  server.fsimage\_dir = "/data00/namenode/fsimage";  server.error\_log = "/data00/namenode/logs/error.log";  server.pid\_file = "/data00/namenode/pid/namenode.pid";  server.coredump\_dir = "/data00/namenode/coredump";  server.log\_level = LOG\_INFO;  server.recv\_buff\_len = 64KB;  server.send\_buff\_len = 64KB;  server.max\_tqueue\_len = 1000;  server.dn\_timeout = 600;  # namenode2  Server server;  server.daemon = ALLOW;  server.workers = 8;  server.connections = 65536;  server.bind\_for\_cli = "192.168.1.2:8000";  server.bind\_for\_dn = "192.168.1.2:8001";  server.my\_paxos = "192.168.1.2:8002";  server.ot\_paxos = "192.168.1.1:8002, 192.168.1.2:8002, 192.168.1.3:8002";  server.paxos\_group\_num = 1;  server.checkpoint\_num = 10000;  server.index\_num = 1000000;  server.editlog\_dir = "/data00/namenode/editlog";  server.fsimage\_dir = "/data00/namenode/fsimage";  server.error\_log = "/data00/namenode/logs/error.log";  server.pid\_file = "/data00/namenode/pid/namenode.pid";  server.coredump\_dir = "/data00/namenode/coredump";  server.log\_level = LOG\_INFO;  server.recv\_buff\_len = 64KB;  server.send\_buff\_len = 64KB;  server.max\_tqueue\_len = 1000;  server.dn\_timeout = 600;  # namenode3  Server server;  server.daemon = ALLOW;  server.workers = 8;  server.connections = 65536;  server.bind\_for\_cli = "192.168.1.3:8000";  server.bind\_for\_dn = "192.168.1.3:8001";  server.my\_paxos = "192.168.1.3:8002";  server.ot\_paxos = "192.168.1.1:8002, 192.168.1.2:8002, 192.168.1.3:8002";  server.paxos\_group\_num = 1;  server.checkpoint\_num = 10000;  server.index\_num = 1000000;  server.editlog\_dir = "/data00/namenode/editlog";  server.fsimage\_dir = "/data00/namenode/fsimage";  server.error\_log = "/data00/namenode/logs/error.log";  server.pid\_file = "/data00/namenode/pid/namenode.pid";  server.coredump\_dir = "/data00/namenode/coredump";  server.log\_level = LOG\_INFO;  server.recv\_buff\_len = 64KB;  server.send\_buff\_len = 64KB;  server.max\_tqueue\_len = 1000;  server.dn\_timeout = 600; |

|  |
| --- |
| # datanode.conf  # datanode1  Server server;  server.daemon = ALLOW;  server.workers = 8;  server.connections = 65536;  server.bind\_for\_cli = "192.168.1.4:8100";  server.ns\_srv = "192.168.1.1:8001, 192.168.1.2:8001, 192.168.1.3:8001";  server.data\_dir = "/data01/block,/data02/block,/data03/block";  server.error\_log = "/data00/datanode/logs/error.log";  server.pid\_file = "/data00/datanode/pid/datanode.pid";  server.coredump\_dir = "/data00/datanode/coredump/";  server.log\_level = LOG\_INFO;  server.recv\_buff\_len = 64KB;  server.send\_buff\_len = 64KB;  server.max\_tqueue\_len = 1000;  server.heartbeat\_interval = 3;  server.block\_report\_interval = 3600;  # datanode2  Server server;  server.daemon = ALLOW;  server.workers = 8;  server.connections = 65536;  server.bind\_for\_cli = "192.168.1.5:8100";  server.ns\_srv = "192.168.1.1:8001, 192.168.1.2:8001, 192.168.1.3:8001";  server.data\_dir = "/data01/block,/data02/block,/data03/block";  server.error\_log = "/data00/datanode/logs/error.log";  server.pid\_file = "/data00/datanode/pid/datanode.pid";  server.coredump\_dir = "/data00/datanode/coredump/";  server.log\_level = LOG\_INFO;  server.recv\_buff\_len = 64KB;  server.send\_buff\_len = 64KB;  server.max\_tqueue\_len = 1000;  server.heartbeat\_interval = 3;  server.block\_report\_interval = 3600;  # datanode3  Server server;  server.daemon = ALLOW;  server.workers = 8;  server.connections = 65536;  server.bind\_for\_cli = "192.168.1.6:8100";  server.ns\_srv = "192.168.1.1:8001, 192.168.1.2:8001, 192.168.1.3:8001";  server.data\_dir = "/data01/block,/data02/block,/data03/block";  server.error\_log = "/data00/datanode/logs/error.log";  server.pid\_file = "/data00/datanode/pid/datanode.pid";  server.coredump\_dir = "/data00/datanode/coredump/";  server.log\_level = LOG\_INFO;  server.recv\_buff\_len = 64KB;  server.send\_buff\_len = 64KB;  server.max\_tqueue\_len = 1000;  server.heartbeat\_interval = 3;  server.block\_report\_interval = 3600; |

1. 启动

* Namenode：sbin/namenode
* Datanode：sbin/datanode
* DFSCLient：sbin/dfscli

1. 使用

|  |
| --- |
| $ sbin/dfscli  Usage: sbin/dfscli cmd...  -mkdir <path>  -rmr <path>  -ls <path>  -put <local path> <remote path>  -get <remote path> <local path>  -rm <path> |