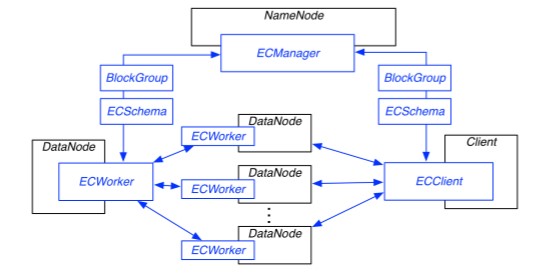
HDFS EC源码分析

HDFS EC的系统结构图如下所示：

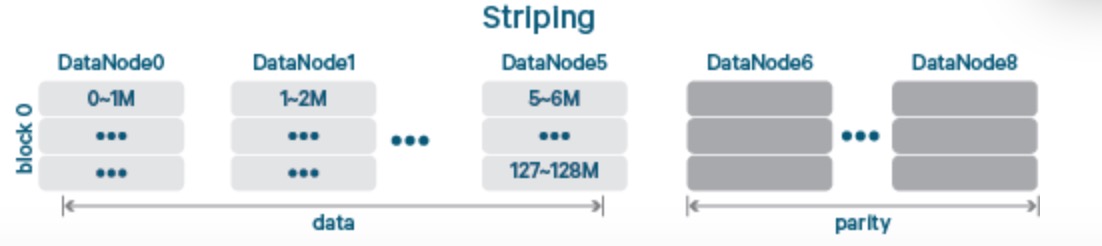


* ECClient，扩展HDFS Client，读取EC文件，与DN交互获取及写入Stripes Data
* ECManager，NN的模块，管理EC Block Groups，包括group分配、数据放置、监控及数据的恢复。当需要进行数据恢复时，将恢复命令及需要的信息（ECSchema,Block Groups等）发送给ECWorker。
* ECWorker，DN不感知EC，但是在DN中添加ECWorker用于接收ECManager的数据恢复命令，根据命令从其他DN中拉取数据、执行codec计算、构建恢复block等。在执行过程中.

下面对EC进行详细的介绍。

# 相关概念

HDFS EC目前支持条形布局，如下图所示：



例如6blocks的文件使用3复本，需要18个Blocks的磁盘空间，但是使用EC(6 data, 3 parity)实现，只需要消耗9 block的磁盘空间。上图使用的是RS(6,3)，前面从DN0-DN5共有6个数据块，DN6-DN8是校验块

1. EC Group

在EC中数据块被分成Group来计算校验码，例如EC(6block,3 parity block)，将6个数据块分成一组，然后计算出3个校验块，从而构成一个EC Group

1. EC Coder，EC Block计算的算法，目前支持XOR和RS两种算法
2. EC Schema

定义了EC执行的参数配置，包括数据块和校验块的数目，codec算法（RS或者XOR），其描述如下：

*private final String codecName; //codec算法，hhxor,xor和rs*

*private final int numDataUnits; //数据块个数*

*private final int numParityUnits; //校验块个数*

1. EC Cell

计算校验块以Cell为单位，默认4KB

1. EC Policy

定义了EC文件的读写及编解码码的执行策略，其示例如下：

*private final String name; //Policy Name，如XOR-2-1-1024k，RS-6-3-1024k等  
private final ECSchema schema; //EC schema  
private final int cellSize; //cell大小  
private final byte id; //id，用于ErasureCodingPolicyManager*

# EC执行流程

EC的使用首先是通过命令设置目录的EC Policy：

*hdfs ec -setPolicy -path /ec -policy XOR-2-1-1024k*

将文件添加到该目录后自动以EC格式存储，

## 2.1 文件写流程

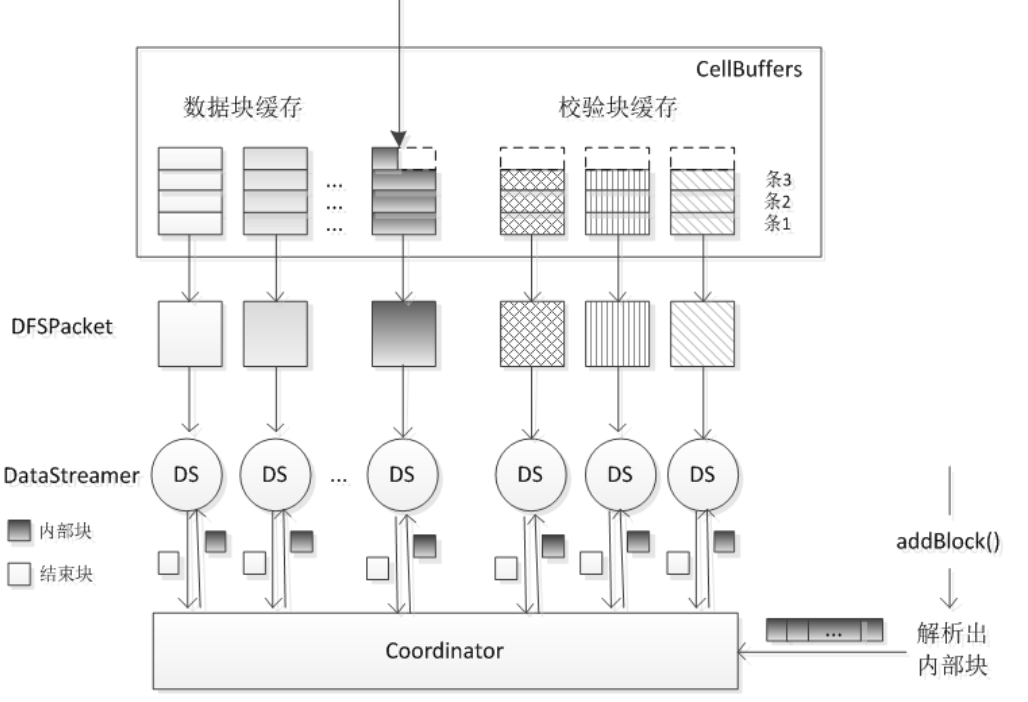
数据写入过程如下所示：



为了支持文件的EC存储，首先要设置其所在目录为EC Zone，其执行过程比较简单，通过客户端调用setEcPolicy设置文件元数据，在InodeXattrFeature中添加EC Policy的属性。设置后向该目录下写入文件，自动写入EC格式数据，其执行过程如下：

1. 客户端发起写文件请求，NN检查目标目录及父目录，将文件状态（HdfsFileStatus）返回给客户端，该状态信息中会有ErasureCodingPolicy
2. 客户端根据ErasureCodingPolicy，创建对应的DFSStripedOutputStream，开始文件写入
3. 客户端会对文件进行切分，将文件块划分BlockGroup，然后向NameNode申请一个BlockGroup的多Block块（例如6 DataBlock+3ParityBlock），如果EC DataBlock为多副本，要建立Pipeline，NN将分配好的BlockGroup分配给客户端
4. 客户端开始写入数据，由于目前EC仅支持Striped格式存储，以CellSize为单位将原文件数据逐次的写入到BlockGroup中的DataPacket，同时将这些数据存储到CellBuffers中
5. CellSize比DataPacket数据缓存数据量小，因此CellBuffers先放满，根据EC Schema进行文件数据的parity计算，生成校验数据，将其写入Parity DataPacket
6. DataPacket数据放满后，客户端以Packet为单位将数据写入到DataNode中
7. 数据流中StriptedDataStreamer和Coordinator起到数据packet的管理作用，以实现文件的写入以BlockGroup为单位完成
8. BlockGroup写入成功后，重复步骤3-7，完成文件写流程

下图是DFSStripedOutputStream的数据流转原理图：



## **2.2 文件读流程**

读的过程相对简单，如下图所示：

****

1. 客户端与NN通信，获取元数据，包括block、block所在的服务器
2. 客户端根据ESStoryPolic，将Block根据offset，构成Stripe BlocksGroup
3. 客户端通过StripeReader与LocatedBlocks对应的所有DN连接，与Data Cell为单位同时读取数据（包括校验block数据）
4. 检查数据是否有丢失，如果有的话，进行decode操作恢复丢失的数据
5. StripeReader将数据进行组装，返回给客户端

# HDFS支持EC作为的扩展

HDFS为EC做了多处扩展，客户端已经在第二部分做了详细介绍

## **Block元数据扩展**

在读写流程中，客户端从NameNode获取元数据，然后根据元数据中保存的EC StoragePolicy初始化EC 数据流，如下所示：

1. 写数据流

*static DFSOutputStream newStreamForCreate( String src,..., String ecPolicyName) {*

*......*

*HdfsFileStatus stat = dfsClient.namenode.create(src, masked, dfsClient.clientName,*

*new EnumSetWritable<>(flag), createParent, replication,*

*blockSize, SUPPORTED\_CRYPTO\_VERSIONS, ecPolicyName);*

*....*

*if(stat.getErasureCodingPolicy() != null) {*

*out = new DFSStripedOutputStream(dfsClient, src, stat,*

*flag, progress, checksum, favoredNodes);} .....*

*}*

1. 读数据流

*private DFSInputStream openInternal(LocatedBlocks locatedBlocks, String src) {*

*.....*

*ErasureCodingPolicy ecPolicy = locatedBlocks.getErasureCodingPolicy();*

*if (ecPolicy != null) {*

*return new DFSStripedInputStream(this, src, verifyChecksum, ecPolicy,*

*locatedBlocks);*

*}...*

*}*

可以看到客户端根据ErasureCodingPolicy来创建EC数据流，其中文件的ErasureCodingPolicy是从NN中获取，其调用如下：

*private static HdfsFileStatus createFileStatus( FSDirectory fsd, INodesInPath iip, INode child){*

*...*

*//从FSNameSystem中获取Policy*

*final ErasureCodingPolicy ecPolicy = FSDirErasureCodingOp*

*.unprotectedGetErasureCodingPolicy(fsd.getFSNamesystem(), iip);*

*//返回FileStatus中，设置ecPolicy*

*return createFileStates(... ecPolicy,...)*

*}*

进一步的分析，可以获知EcPolicy是从FSDirectory#INode中获取，如下:

*private static ErasureCodingPolicy getErasureCodingPolicyForPath(*

*FSDirectory fsd, INodesInPath iip) {*

*......*

*if (inode.isFile()) {*

*byte id = inode.asFile().getErasureCodingPolicyID();*

*return id < 0 ? null :*

*fsd.getFSNamesystem().getErasureCodingPolicyManager().getByID(id);*

*}*

*...*

*final XAttrFeature xaf = inode.getXAttrFeature(iip.getPathSnapshotId());*

*XAttr xattr = xaf.getXAttr(XATTR\_ERASURECODING\_POLICY);*

*.....*

*String ecPolicyName = WritableUtils.readString(dIn);*

*return fsd.getFSNamesystem().getErasureCodingPolicyManager()*

*.getByName(ecPolicyName);*

*}*

以上可知，为了支持EC，在INode的XAttrFeature中，增加了配置

*String XATTR\_ERASURECODING\_POLICY =  
 "system.hdfs.erasurecoding.policy";*

1. 元数据中ECPolicy的配置

客户端通过set ECPolicy来配置文件的policy，执行如下：

*hdfs ec -setPolicy -path /ec -policy XOR-2-1-1024k*

最终调用setINodeXAttrs来对

*static FileStatus setErasureCodingPolicy(*

*final FSNamesystem fsn,final String srcArg, final String ecPolicyName){*

*....*

*String src = srcArg;FSDirectory fsd = fsn.getFSDirectory();final INodesInPath iip;*

*List<XAttr> xAttrs;*

*...*

*WritableUtils.writeString(dOut, ecPolicy.getName());*

*ecXAttr = XAttrHelper.buildXAttr(XATTR\_ERASURECODING\_POLICY,*

*bOut.toByteArray())*

*ErasureCodingPolicy ecPolicy = getErasureCodingPolicyByName(fsn,*

*ecPolicyName);*

*...*

*xAttrs = setErasureCodingPolicyXAttr(fsn, iip, ecPolicy);}*

## **NameNode端的扩展**

1）ErasureCodingPolicyManager

在第一部分，客户端进行EC Policy的获取与设置是都是通过EC PolicyName来配置，但是在NameNode端首先通过ErasureCodingPolicyManager获取其对应的Policy，其调用如下：

*return fsd.getFSNamesystem().getErasureCodingPolicyManager()*

*.getByName(ecPolicyName);*

其相关类如下：



NN通过ErasureCodingPolicyManager来管理可用的ErasureCodingPolicy，如下:

*$ hdfs ec -listPolicies*

*ErasureCodingPolicy=[Name=RS-10-4-1024k, Schema=[..., Id=5], State=DISABLED*

*ErasureCodingPolicy=[Name=RS-3-2-1024k, Schema=[..., Id=2], State=DISABLED*

*ErasureCodingPolicy=[Name=RS-6-3-1024k, Schema=[..., Id=1], State=ENABLED //默认*

*ErasureCodingPolicy=[Name=RS-LEGACY-6-3-1024k, Schema=[..., Id=3], State=DISABLED*

*ErasureCodingPolicy=[Name=XOR-2-1-1024k, Schema=[...., Id=4], State=DISABLE*

在系统中默认支持以上五种Policy，其中Schema的定义如下所示：

*[ECSchema=[Codec=rs, numDataUnits=10, numParityUnits=4]], CellSize=1048576, Id=5]*

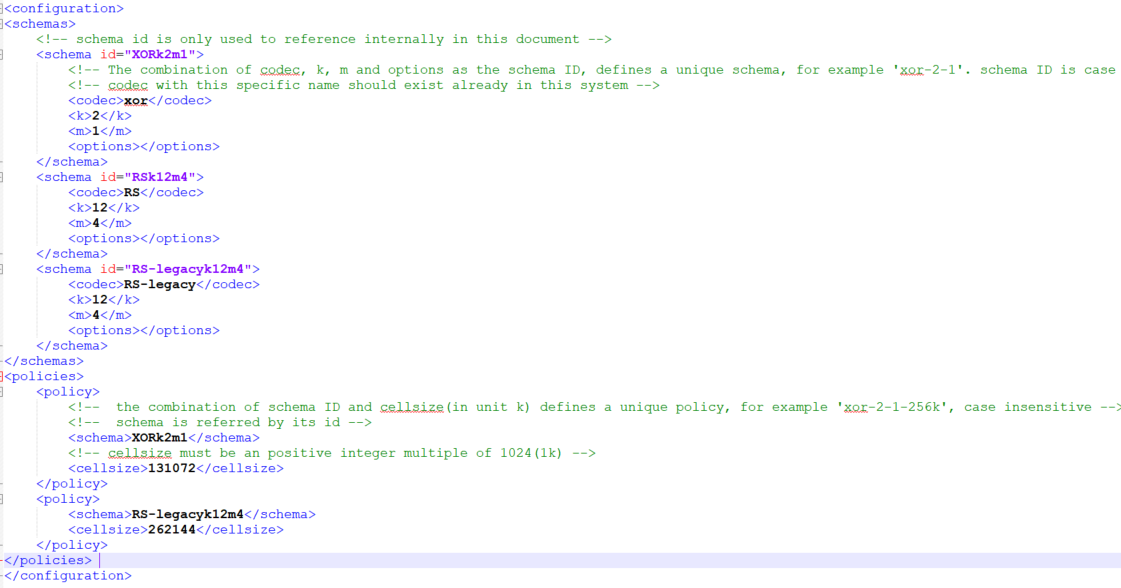
这几种Policy，定义在SystemErasureCodingPolicies中，XOR\_2\_1\_POLICY定义如下：

*public static final byte XOR\_2\_1\_POLICY\_ID = 4;  
private static final ErasureCodingPolicy SYS\_POLICY4 =  
 new ErasureCodingPolicy(ErasureCodeConstants.XOR\_2\_1\_SCHEMA,  
 DEFAULT\_CELLSIZE, XOR\_2\_1\_POLICY\_ID);*

用户可以通过添加自定义的policies，命令如下：

*$hdfs ec -addPolicies -policyFile <file>*

policyFile的定义格式如下所示：



https://issues.apache.org/jira/browse/HDFS-11604

1. EC Block的自动恢复

在NN中有一个守护线程RedundancyMonitor，周期性的执行数据块的备份和删除任务，其维护数据块的三副本的有效性，包括丢失块的恢复等，当检测到数据块丢失时，调度数据块恢复流程

*BlockManager#scheduleReconstruction(BlockInfo block,priority){*

*BlockCollection bc = getBlockCollection(block);*

*//判断block是否为ec block，构建响应的恢复Work*

*if (block.isStriped()) {*

*....*

*return new ErasureCodingWork(getBlockPoolId(), block, bc, srcNodes,*

*containingNodes, liveReplicaNodes, additionalReplRequired,*

*priority, indices);*

*}*

*}*

将其添加到DatanodeDescriptor#erasurecodeBlocks中

*void addBlockToBeErasureCoded(ExtendedBlock block,*

*DatanodeDescriptor[] sources, DatanodeStorageInfo[] targets,*

*byte[] liveBlockIndices, ErasureCodingPolicy ecPolicy) {*

*BlockECReconstructionInfo task = new BlockECReconstructionInfo(block,*

*sources, targets, liveBlockIndices, ecPolicy);*

*erasurecodeBlocks.offer(task);}*

## **DataNode ErasureCodingWorker**

DN通过BPOfferService从NN接收BlockECReconstrunctionCommand, NN端通过心跳发送，DatanodeManager#heartbeat执行如下：

*BlockRecoveryCommand brCommand = new BlockRecoveryCommand(blocks.length);*

*rBlock = new RecoveringBlock(primaryBlock, recoveryInfos,  
 uc.getBlockRecoveryId());  
 if (b.isStriped()) {  
 rBlock = new RecoveringStripedBlock(rBlock, uc.getBlockIndices(),  
 ((BlockInfoStriped) b).getErasureCodingPolicy());  
 }  
}  
brCommand.add(rBlock);*

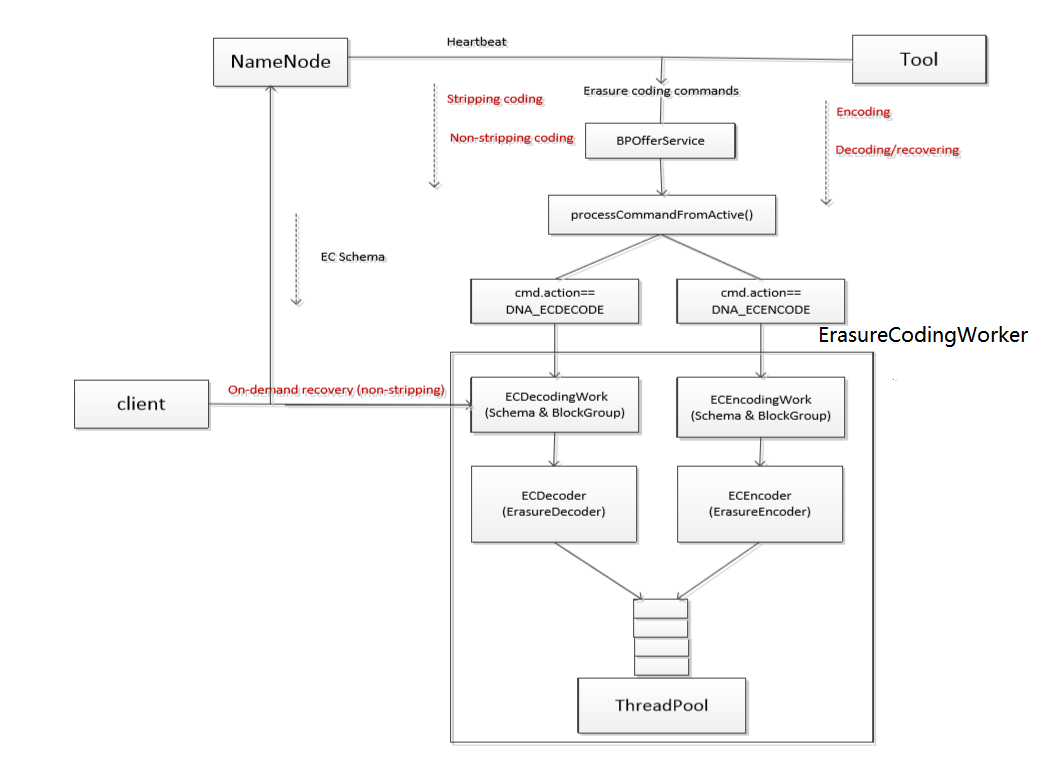
DN接收到BlockECReconstructionCommand后，通过ErasureCodingWorker来执行恢复，其相关类图如下：



StripedBlockReconstructor执行如下：

*private void reconstructTargets(int toReconstructLen) throws IOException {  
 ByteBuffer[] inputs = getStripedReader().getInputBuffers(toReconstructLen);  
 int[] erasedIndices = stripedWriter.getRealTargetIndices();  
 ByteBuffer[] outputs = stripedWriter.getRealTargetBuffers(toReconstructLen);  
 getDecoder().decode(inputs, erasedIndices, outputs); //数据的解码，实现数据的恢复  
 ...  
}*

执行流程如下所示：



相关JIRA:

https://issues.apache.org/jira/browse/HADOOP-11264

https://issues.apache.org/jira/browse/HDFS-7285

https://issues.apache.org/jira/browse/HDFS-8031

EC简介：https://blog.csdn.net/icycode/article/details/51542235