Alluxio Policy

在Alluxio写数据时，在FileOutStream通过一些条件检查之后，调用getNextBlock方法，然后调用mCurrentBlockOutStream来真正将数据写入到相应的worker中，代码如下所示：

*private void getNextBlock() throws IOException {*

*if (mCurrentBlockOutStream != null) {*

*Preconditions.checkState(mCurrentBlockOutStream.remaining() <= 0,*

*PreconditionMessage.ERR\_BLOCK\_REMAINING);*

*mCurrentBlockOutStream.flush();*

*mPreviousBlockOutStreams.add(mCurrentBlockOutStream);*

*}*

*if (mAlluxioStorageType.isStore()) {*

*mCurrentBlockOutStream =*

*mBlockStore.getOutStream(getNextBlockId(), mBlockSize, mOptions);*

*mShouldCacheCurrentBlock = true;*

*}*

*}*

mCurrentBlockOutStream时通过mContext先获取mBlockStore，然后getOutStream，代码如下：

*public BlockOutStream getOutStream(long blockId, long blockSize, OutStreamOptions options)*

*throws IOException {*

*WorkerNetAddress address;*

*FileWriteLocationPolicy locationPolicy = Preconditions.checkNotNull(options.getLocationPolicy(),*

*PreconditionMessage.FILE\_WRITE\_LOCATION\_POLICY\_UNSPECIFIED);*

*address = locationPolicy.getWorkerForNextBlock(getEligibleWorkers(), blockSize);*

*if (address == null) {*

*throw new UnavailableException(*

*ExceptionMessage.NO\_SPACE\_FOR\_BLOCK\_ON\_WORKER.getMessage(blockSize));*

*}*

*return getOutStream(blockId, blockSize, address, options);*

*}*

这里可以看到LocationPolicy会根据用户设置的文件写位置选择策略来返回Worker的Ip地址，然后将blockId，blockSize和address再调用getOutStream，从而返回FileOutStream

FileWriteLocationPlicy可以通过*alluxio.user.file.write.avoid.location.policy.class*来指定Worker的选择策略，默认为LocalFirstPolicy，目前支持多种主机选择策略包括：RoundRobinPolicy，SpecificHostPolicy等多种策略，如下所示：



# 1.LocalFirstPolicy&LocalFirstAvoidEvictionPolicy

LocalFirstPolicy时一个优先返回本地主机的定位策略，如果本地Worker没有足够的容量，那么就从活跃有效的workers列表随机选择一个worker用于每个块写入。再其内部，有一个关键的成员变量mTieredIdentity，即本地主机信息，用于选择本地worker的关键变量：

*// 本地主机名，用于选择本地本机worker的关键变量*

*Private TieredIdentity mTieredIdentity = null;*

在构造函数中，初始化这个本机信息变量：

*public LocalFirstPolicy() {*

*this(TieredIdentityFactory.localIdentity());*

*}*

LocalFirstPolicy对外提供定位策略的核心方法：getWorkerForNextBlock，为下一个数据块选择一个Worker，返回的网络地址为WorkerNetAddress，代码如下：

*public WorkerNetAddress getWorkerForNextBlock(Iterable<BlockWorkerInfo> workerInfoList,*

*long blockSizeBytes) {*

*List<BlockWorkerInfo> shuffledWorkers = Lists.newArrayList(workerInfoList);*

*Collections.shuffle(shuffledWorkers);*

*//遍历Worker列表workerInfoList，排除不满足需要的blockSizeBytes的节点*

*List<BlockWorkerInfo> candidateWorkers = shuffledWorkers.stream()*

*.filter(worker -> worker.getCapacityBytes() >= blockSizeBytes)*

*.collect(Collectors.toList());*

*//从列表中选择本地Worker节点，即nearest节点*

*// List<TieredIdentity> identities = candidateWorkers.stream()*

*.map(worker -> worker.getNetAddress().getTieredIdentity())*

*.filter(Objects::nonNull)*

*.collect(Collectors.toList());*

*Optional<TieredIdentity> nearest = mTieredIdentity.nearest(identities);*

*if (!nearest.isPresent()) {*

*return null;*

*}*

*// Map back to the worker with the nearest tiered identity.*

*return candidateWorkers.stream()*

*.filter(worker -> worker.getNetAddress().getTieredIdentity().equals(nearest.get()))*

*.map(worker -> worker.getNetAddress())*

*.findFirst().orElse(null);*

*}*

LocalFirstAvoidEvictionPolicy是LocalFirstPolicy的改进版，首先返回本地host，如果local Worker没有足够大的空间，则从worker list中随机选择一个Worker返回。

# 2.RoundRobinPolicy

通过循环遍历的方式跳过没有足够空间的Workers的下一个Block选择的策略，如果没有找到，则返回Null，其中核心的成员变量：

*private List<BlockWorkerInfo> mWorkerInfoList;*

*private int mIndex;*

*private boolean mInitialized = false;*

mWorkerInfoList为初始化的BlockWorkerInfo列表，每次都从这个列表中选择BlockWorkerInfo当核心方法getWorkerForNextBlock第一次被调用时进行初始化，由传入的BlockWorkerInfo列表进行赋值。

mIndex为mWorkerInfoList列表当前遍历到的索引

mInitialized为是否初始化的标志位，默认为未初始化false

核心方法getWorkerForNextBlock代码如下：

*public WorkerNetAddress getWorkerForNextBlock(Iterable<BlockWorkerInfo> workerInfoList, long blockSizeBytes) {*

*if (!mInitialized) {*

*mWorkerInfoList = Lists.newArrayList(workerInfoList);*

*Collections.shuffle(mWorkerInfoList); //进行shuffle,避免热点*

*mIndex = 0;*

*mInitialized = true;*

*}*

*// 遍历所有的worker*

*for (int i = 0; i < mWorkerInfoList.size(); i++) {*

*WorkerNetAddress candidate = mWorkerInfoList.get(mIndex).getNetAddress();*

*BlockWorkerInfo workerInfo = findBlockWorkerInfo(workerInfoList, candidate);*

*//索引Index的重置，也就是下次取节点的时候取index下一个*

*mIndex = (mIndex + 1) % mWorkerInfoList.size();*

*if (workerInfo != null && workerInfo.getCapacityBytes() >= blockSizeBytes) {*

*return candidate;*

*}*

*}*

*return null;*

*}*

# 3.SpecificHostPolicy

指定主机策略SpecificHostPolicy总是返回一个指定主机名的worker定位策略，如果该主机名对应机器上没有活跃的worker，则返回null。在SpecificHostPolicy内部，封装的核心成员遍历为：

*private final String mHostname;*

核心方法getWorkerForNextBlock，如下：

*public WorkerNetAddress getWorkerForNextBlock(Iterable<BlockWorkerInfo> workerInfoList,*

*long blockSizeBytes) {*

*// 选择匹配mHostName的Worker*

*for (BlockWorkerInfo info : workerInfoList) {*

*if (info.getNetAddress().getHost().equals(mHostname)) {*

*return info.getNetAddress();*

*}*

*}*

*return null;*

*}*

# 4. MostAvaliableFirstPolicy

选择剩余空间最大的Worker的Policy，核心方法getWorkerForNextBlock如下：

*public WorkerNetAddress getWorkerForNextBlock(Iterable<BlockWorkerInfo> workerInfoList,*

*long blockSizeBytes) {*

*long mostAvailableBytes = -1;*

*WorkerNetAddress result = null;*

*for (BlockWorkerInfo workerInfo : workerInfoList) {*

*//遍历worker列表，如果大于目前的最大空间空间，则返回该worker*

*if (workerInfo.getCapacityBytes() - workerInfo.getUsedBytes() > mostAvailableBytes) {*

*mostAvailableBytes = workerInfo.getCapacityBytes() - workerInfo.getUsedBytes();*

*result = workerInfo.getNetAddress();*

*}*

*}*

*return result;*

*}*

参考文献：

本地优先策略：https://blog.csdn.net/lipeng\_bigdata/article/details/50969036

循环遍历策略：https://blog.csdn.net/lipeng\_bigdata/article/details/50970152

指定主机策略：https://blog.csdn.net/lipeng\_bigdata/article/details/50970561

https://github.com/gjhkael/Alluxio-Internal/blob/master/markdown/4-AlluxioBlockWrite.md