Hadoop RPC方法添加总结

在Hadoop中添加方法getPathByNodeId，用于根据nodeId:String获取该文件的路径名。该方法涉及到的服务主要是DFSClient及Namenode。DFSClient通过RPC调用从Namenode获取数据。

Client与Namenode的protocol是ClientProtocol，DFSClient通过namenode:Client

Protocol RPC调用Namenode中的方法，具体的过程不再分析：

Hadoop RPC通过Protobuf进行序列化，因此要在ClientNamenodeProtocol.proto文件中增加两个通信方法，如下：

message GetPathByNodeIdRequestProto{

required uint64 nodeId = 1;

}

message GetPathByNodeIdResponseProto{

required string fullPath = 1;

}

并且，在service ClientNamenodeProtocol中增加下面代码：

rpc getPathByNodeId(GetPathByNodeIdRequestProto)

returns(GetPathByNodeIdResponseProto);

通过Protobuf编译后会生成方法getPathByNodeId的相关通信类。

# 1、ClientProtocol.java

接口中，增加方法：

public String getPathByNodeId(long nodeId) throws IOException

# 2、DFSClient

客户端中增加，方法getPathByNodeId：

public String getPathByNodeId(long nodeId) throws IOException {

return namenode.getPathByNodeId(nodeId);

}

# 3、NamenodeRpcServer

public String getPathByNodeId(long nodeId) throws IOException {

return namesystem.getFSDirectory().getInode(nodeId).getFullPathName();

}

# 4、ClientNamenodeProtocolServerSideTranslatorPB

public GetPathByNodeIdResponseProto getPathByNodeId( RpcController controller,

GetPathByNodeIdRequestProto req) throws ServiceException{

try {

String path = server.getPathByNodeId(req.getNodeId());

GetPathByNodeIdResponseProto.Builder builder = GetPathByNodeId

ResponseProto.newBuilder();

builder.setFullPath(path);

return builder.build();

} catch (IOException e) {

throw new ServiceException(e);

}

}

用于RPC通信的服务端

# 5.ClientNamenodeProtocolTranslatorPB

public String getPathByNodeId(long nodeId) throws IOException {

GetPathByNodeIdRequestProto req = GetPathByNodeIdRequestProto.

newBuilder() .setNodeId(nodeId).build();

try {

return rpcProxy.getPathByNodeId(null, req).getFullPath();

} catch (ServiceException e) {

throw ProtobufHelper.getRemoteException(e);

}

}

用于RPC通信的客户端

# 总结：

涉及到Client与Namenode（HDFS），增加方法，开发流程是相同的。都要修改以上5个类。