一、DataManagingApp，Main方法里初始化了:

1、AppPropertyCenter:继承了SystemPropertyCenter，读取了conf下每一个模块的文件夹里的app.properties配置

2、AtomicReference:多线程环境下，无锁的进行原子操作类，使打包是原子级操作，原本是用于多线程打包

3、同时启动两个任务：TrackletPackingThread负责打包，DataManagingApp负责保存tracklet、属性识别结果、reid结果以及视频切割（目前没有使用）

4、在new DataManagingApp构造方法后，initialize方法初始化整个application

（1）DataManagingApp：继承了SparkStreamingApp，它的构造方法：

[1]super(propCenter, APP\_NAME)：初始化了监控线程MonitorThread；

registerStreams注册了新的流，目前是有保存tracklet、属性识别结果、reid结果以及视频切割（目前没有使用）

（2）TrackletSavingStream：继承了Stream，它的构造方法里

[1]super(APP\_NAME, propCenter)：得到了loggerSingleton、dbConnSingleton、producerSingleton（继承自kafka生产者）

同时初始化了原始视频路径

[2]addToGlobalStream(Map<DataType, JavaPairDStream<UUID, TaskData>> globalStreamMap)方法：filter根据Port过滤流，globalStreamMap是一个流的map，key是topic的名字，value是过滤后的流，遍历rdd(JavaPairRDD)，每次遍历都加锁，通过kv （[Tuple2](eclipse-javadoc:%E2%98%82=las-vpe-platform/src\\/main\\/java<org.cripac.isee.vpe.data{DataManagingApp.java%E2%98%83DataManagingApp%E2%98%83TrackletSavingStream~addToGlobalStream~QMap\\<QDataType;QJavaPairDStream\\<QUUID;QTaskData;>;>;=)="Lorg.apache.spark.api.java.function.VoidFunction\\<Lorg.apache.spark.api.java.JavaPairRDD\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;!23934!28861!23939=&call!1="Lorg.apache.spark.api.java.JavaPairRDD\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;="rdd="V="Lorg\\/apache\\/spark\\/api\\/java\\/function\\/VoidFunction\\<Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;.call\\(Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;)V\\|Ljava\\/lang\\/Exception;@rdd!23934!23936!23934!23936!Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD;!0!true=&=)="Lorg.apache.spark.api.java.function.VoidFunction\\<Ljava.util.Iterator\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;>;!23962!28860!23970=&call!1="Ljava.util.Iterator\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;="kvIter="V="Lorg\\/apache\\/spark\\/api\\/java\\/function\\/VoidFunction\\<Ljava\\/util\\/Iterator\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;>;.call\\(Ljava\\/util\\/Iterator\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;)V\\|Ljava\\/lang\\/Exception;@kvIter!23962!23967!23962!23967!Ljava\\/util\\/Iterator;!0!true=&=)="Ljava.util.function.Consumer\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;!24276!28810!24280=&accept!1="Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;="kv="V="Ljava\\/util\\/function\\/Consumer\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;.accept\\(Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;)V@kv!24276!24277!24276!24277!Lscala\\/Tuple2;!0!true=&@kv!24276!24277!24276!24277!Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;!0!true%E2%98%82scala.Tuple2" \o "in scala)<[UUID](eclipse-javadoc:%E2%98%82=las-vpe-platform/src\\/main\\/java<org.cripac.isee.vpe.data{DataManagingApp.java%E2%98%83DataManagingApp%E2%98%83TrackletSavingStream~addToGlobalStream~QMap\\<QDataType;QJavaPairDStream\\<QUUID;QTaskData;>;>;=)="Lorg.apache.spark.api.java.function.VoidFunction\\<Lorg.apache.spark.api.java.JavaPairRDD\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;!23934!28861!23939=&call!1="Lorg.apache.spark.api.java.JavaPairRDD\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;="rdd="V="Lorg\\/apache\\/spark\\/api\\/java\\/function\\/VoidFunction\\<Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;.call\\(Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;)V\\|Ljava\\/lang\\/Exception;@rdd!23934!23936!23934!23936!Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD;!0!true=&=)="Lorg.apache.spark.api.java.function.VoidFunction\\<Ljava.util.Iterator\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;>;!23962!28860!23970=&call!1="Ljava.util.Iterator\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;="kvIter="V="Lorg\\/apache\\/spark\\/api\\/java\\/function\\/VoidFunction\\<Ljava\\/util\\/Iterator\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;>;.call\\(Ljava\\/util\\/Iterator\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;)V\\|Ljava\\/lang\\/Exception;@kvIter!23962!23967!23962!23967!Ljava\\/util\\/Iterator;!0!true=&=)="Ljava.util.function.Consumer\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;!24276!28810!24280=&accept!1="Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;="kv="V="Ljava\\/util\\/function\\/Consumer\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;.accept\\(Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;)V@kv!24276!24277!24276!24277!Lscala\\/Tuple2;!0!true=&@kv!24276!24277!24276!24277!Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;!0!true%E2%98%82java.util.UUID" \o "in java.util), [TaskData](eclipse-javadoc:%E2%98%82=las-vpe-platform/src\\/main\\/java<org.cripac.isee.vpe.data{DataManagingApp.java%E2%98%83DataManagingApp%E2%98%83TrackletSavingStream~addToGlobalStream~QMap\\<QDataType;QJavaPairDStream\\<QUUID;QTaskData;>;>;=)="Lorg.apache.spark.api.java.function.VoidFunction\\<Lorg.apache.spark.api.java.JavaPairRDD\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;!23934!28861!23939=&call!1="Lorg.apache.spark.api.java.JavaPairRDD\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;="rdd="V="Lorg\\/apache\\/spark\\/api\\/java\\/function\\/VoidFunction\\<Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;.call\\(Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;)V\\|Ljava\\/lang\\/Exception;@rdd!23934!23936!23934!23936!Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD;!0!true=&=)="Lorg.apache.spark.api.java.function.VoidFunction\\<Ljava.util.Iterator\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;>;!23962!28860!23970=&call!1="Ljava.util.Iterator\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;="kvIter="V="Lorg\\/apache\\/spark\\/api\\/java\\/function\\/VoidFunction\\<Ljava\\/util\\/Iterator\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;>;.call\\(Ljava\\/util\\/Iterator\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;)V\\|Ljava\\/lang\\/Exception;@kvIter!23962!23967!23962!23967!Ljava\\/util\\/Iterator;!0!true=&=)="Ljava.util.function.Consumer\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;!24276!28810!24280=&accept!1="Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;="kv="V="Ljava\\/util\\/function\\/Consumer\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;.accept\\(Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;)V@kv!24276!24277!24276!24277!Lscala\\/Tuple2;!0!true=&@kv!24276!24277!24276!24277!Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;!0!true%E2%98%82org.cripac.isee.vpe.ctrl.TaskData" \o "in org.cripac.isee.vpe.ctrl)> ）得到taskID（UUID）和taskData，进一步得到TrackletOrURL，通过trackletOrURL.getTracklet方法调用HadoopHelper.retrieveTracklet从hdfs上重新得到tracklet，

拼接路径，调用HadoopHelper.storeTracklet方法保存tracklet信息，同时将一些信息保存到neo4j中

（3）AttrSavingStream继承了stream，addToGlobalStream方法中，将Attributes对象保存到neo4j中

（4）IDRankSavingStream继承了stream，目前addToGlobalStream方法中只是打印了日志

二、SparkStreamingApp：实现了Serializable接口，

[1]initialize方法：

1. 调用KafkaHelper.checkTopics来创建topic
2. 根据port，过滤addToGlobalStream中的map
3. 加载spark的配置信息，初始化JavaStreamingContext和JavaSparkContext
4. buildDirectStream：接受kafka的消息，转换为JavaPairDStream
5. 将DataType和JavaPairDStream放在map中，作为参数，调用addToGlobalStream方法
6. 这里同时启用了checkpoint

三、Stream：实现了Serializable接口

[1]Stream构造方法：初始化loggerSingleton、dbConnSingleton、producerSingleton（kafka）

[2]output方法：发送kafka消息

[3]filter方法：过滤map

[4]内部类：Port，通过该类，将每个流不同的类别

四、PedestrianTrackingApp，继承SparkStreamingApp

1、main方法：初始化AppPropertyCenter和PedestrianTrackingApp

（1）AppPropertyCenter，内部类，继承SystemPropertyCenter，初始化vpe.num.sample.per.tracklet参数，该参数是用来定在tracking中采样的时候要采多少帧

（2）PedestrianTrackingApp构造方法：初始化AppPropertyCenter和application，同时将HDFSVideoTrackingStream加到要处理的流中

（3）HDFSVideoTrackingStream，内部类，继承Stream

[1]构造方法中，初始化了confCacheSingleton：读取了tracking的配置，即isee-basic/CAM01\_0.conf

[2]addToGlobalStream方法：

1. 从[Tuple2](eclipse-javadoc:%E2%98%82=las-vpe-platform/src\\/main\\/java<org.cripac.isee.vpe.alg.pedestrian.tracking{PedestrianTrackingApp.java%E2%98%83PedestrianTrackingApp%E2%98%83HDFSVideoTrackingStream~addToGlobalStream~QMap\\<QDataType;QJavaPairDStream\\<QUUID;QTaskData;>;>;=)="Lorg.apache.spark.api.java.function.VoidFunction\\<Lorg.apache.spark.api.java.JavaPairRDD\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;!6682!14493!6687=&call!1="Lorg.apache.spark.api.java.JavaPairRDD\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;="rdd="V="Lorg\\/apache\\/spark\\/api\\/java\\/function\\/VoidFunction\\<Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;.call\\(Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;)V\\|Ljava\\/lang\\/Exception;@rdd!6682!6684!6682!6684!Lorg\\/apache\\/spark\\/api\\/java\\/JavaPairRDD;!0!true=&=)="Lorg.apache.spark.api.java.function.VoidFunction\\<Ljava.util.List\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;>;!6708!14492!6716=&call!1="Ljava.util.List\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;="kvList="V="Lorg\\/apache\\/spark\\/api\\/java\\/function\\/VoidFunction\\<Ljava\\/util\\/List\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;>;.call\\(Ljava\\/util\\/List\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;)V\\|Ljava\\/lang\\/Exception;@kvList!6708!6713!6708!6713!Ljava\\/util\\/List;!0!true=&=)="Ljava.util.function.Consumer\\<Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;>;!7281!14183!7285=&accept!1="Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;="kv="V="Ljava\\/util\\/function\\/Consumer\\<Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;>;.accept\\(Lscala\\/Tuple2\\<Ljava\\/util\\/UUID;Lorg\\/cripac\\/isee\\/vpe\\/ctrl\\/TaskData;>;)V@kv!7281!7282!7281!7282!Lscala\\/Tuple2;!0!true=&@kv!7281!7282!7281!7282!Lscala.Tuple2\\<Ljava.util.UUID;Lorg.cripac.isee.vpe.ctrl.TaskData;>;!0!true%E2%98%82scala.Tuple2" \o "in scala)中得到taskID和taskData，
2. 从taskData.predecessorRes中得到videoURL，这是原视频的hdfs路径
3. 从taskData.getDestNode得到curNode，这是正在执行的node
4. 调用curNode.getExecData()得到confFile
5. 由confFile初始化BasicTracker
6. 初始化FileSystem，得到实例
7. 调用tracker.track将视频流解析成Tracklet数组
8. 遍历Tracklet数组，将tracklet发送到kafka，如果太大，导致异常，在捕获异常时，调用HadoopHelper.storeTracklet保存tracklet，将hdfs的地址通过kafka发送
9. PedestrianAttrRecogApp：继承了SparkStreamingApp、

1、main方法

（1）初始化AppPropertyCenter，配置了一些参数，以及属性识别使用的算法

（2）构造方法将RecogStream初始化

（3）RecogStream，内部类

[1]构造方法，使用DeepMARCaffe2Native初始化了recognizerSingleton，同时指定了gpu，DeepMARCaffe2Native有一个识别的系列，下面介绍

[2]addToGlobalStream方法：

1. 从TrackletOrURL中得到Tracklet
2. 调用recognizerSingleton.getInst().recognize方法将得到的Tracklet解析成Attributes
3. 调用output方法将Attributes发送到kafka
4. PedestrianReIDUsingAttrApp：继承SparkStreamingApp
5. main方法

（1）初始化SystemPropertyCenter

（2）初始化PedestrianReIDUsingAttrApp，构造方法将ReIDStream初始化

（3）ReIDStream，内部类，继承stream

[1]构造方法中初始化了reidSingleton，目前还是fake

[2]addToGlobalStream方法，目前只是匹配topic

1. BasicTracker实现了Tracker接口，类图如下

（1）track方法将视频流解析成Tracklet数组，该方法调用该类中的本地方法，initialize用来初始化视频每一帧

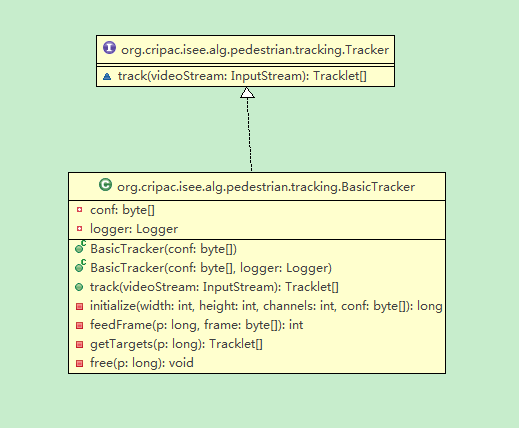


图1 tracking

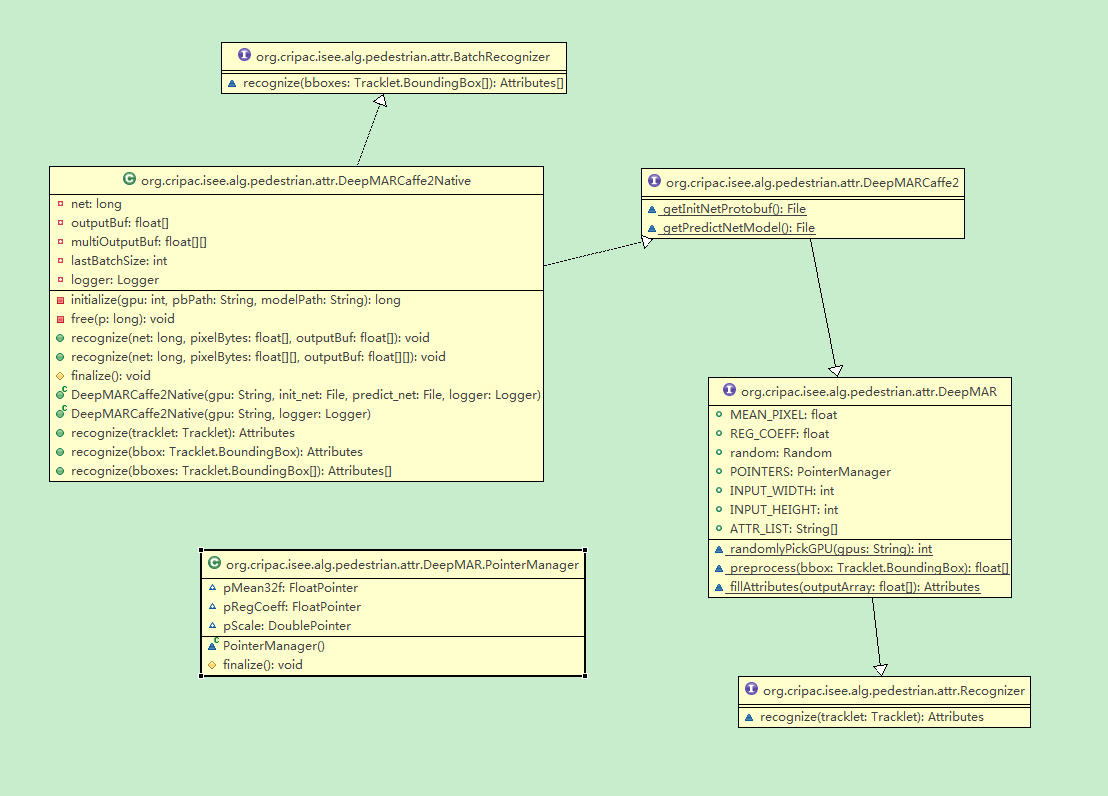


图2 属性识别

1. 控制

1、MessageHandlingApp：继承了SparkStreamingApp

（1）main方法：

[1]初始化SystemPropertyCenter

[2]MessageHandlingApp构造方法

1. 得到kafka生产者ByteArrayProducer
2. 得到HDFSReader

（2）addToContext方法:解析执行命令，循环中调用handle方法

（3）Handle方法：针对不同的命令，初始化不同的app以及执行plan

2、CommandType：内部类，命令的枚举，和app的name相同

1. 属性识别

1、DeepMARCaffe2Native类

1. 构造方法：初始化了gpu，深度学习的.pb文件路径
2. Recognize方法：有几个重载方法，其中一个的入参是Tracklet，调用Attributes.div方法将Tracklet转换成的Attributes结果取了平均，该方法使用了反射，其中该方法将BoundingBox转换成stream，在流的map方法里调用了native的recognize方法，将BoundingBox转换成float数组，该数组是FC8 layer向量
3. net为initialize方法的返回值，是initialize构建出来的网络的指针
4. 调用DeepMAR.preprocess方法将BoundingBox转换为float数组，该数组保存的是像素
5. DeepMAR.fillAttributes方法将存储向量的float数组拼成json，再转换为Attributes对象
6. DeepMARCaffe2接口

两个方法分别得到了init\_net.pb和predict\_net.pb的路径

1. DeepMAR接口
2. ATTR\_LIST：Attributes属性
3. PointerManager类：加载了opencv，负责初始化变量以及释放变量
4. Preprocess方法：处理boundingbox，将其转换成像素float数组

（4）fillAttributes方法：将存储向量的float数组拼成json，再转换为Attributes对象

1. Recognizer和BatchRecognizer接口

分别是不同入参的抽象recognize方法

十、KafkaHelper.scala

操作kafka，

1. send[K, V]方法：发送一条消息，有key-value
2. send[K]方法：发送TaskData
3. createZkUtils：初始化zk
4. checkTopics：创建topic，针对所有类型数据
5. createTopic：创建topic，有很多重载方法

十一、实体类

1、Attributes：属性，要保存到数据库，其中trackletID是Tracklet.Identifier

（1）add方法

（2）div方法

2、FeatureMSCAN：继承Feature，reid向量特征

3、Tracklet：轨迹，包括BoundingBox和Identifier两个内部类  
4、TrackletOrURL：封装了Tracklet和hdfs上保存的URL

5、DataType：枚举类，定义了程序可以接收的类型，初始化Port使用

十二、工具类

1、Factory：针对T，提供produce方法

2、ResourceManager，getResource方法，得到Jar包中指定路径的文件，在得到深度学习文件时使用

3、SerializationHelper，序列化以及反序列化对象

4、Singleton：对任意对象提供单例对象

5、BasicTrackerFactory：实现了Factory接口，Singleton在创建单例的时候，调用Factory，获得一个序列化后的BasicTracker实例对象

6、ParallelExecutor：execute，重载了很多个，将入参转换为stream，进行并行化7、RobustExecutor，如果执行过程出现异常，再次执行

8、AppManager：将app的名字放入map中，启动时使用

9、HDFSReader：listSubfiles方法，将路径下所有的文件路径保存到list中

10、MonitorThread：继承了Thread，在每一个worker上额外开设一个线程，用于监控集群，并将数据发送到kafka

11、SystemPropertyCenter：配置中心，加载了conf/system.properties

十三、数据库

GraphDatabaseConnector：抽象类

Neo4jConnector：具体的实现类

十四、操作hdfs

1. HadoopHelper：
2. retrieveTracklet方法：将hdfs路径下保存的信息转换为Tracklet，用于第二次保存Tracklet时找到第一次保存的Tracklet
3. storeTracklet方法：将Tracklet的信息保存到hdfs
4. HDFSFactory：实现了Factory接口，得到FileSystem实例

十五、kafka

1、ByteArrayProducer：继承了KafkaProducer，得到key=String，value=byte数组型的kafka生产者

2、ByteArrayProducerFactory：实现了Factory接口，得到ByteArrayProducer实例

3、EvenlyDistributingPartitioner：实现了kafka的Partitioner接口，复写了partition方法，用来把消息平均的发送给topic的每个partitions

十六、日志

1、SynthesizedLoggerFactory：实现了Factory接口，产生SynthesizedLogger

2、SynthesizedLogger继承了Logger，加载了log4j.properties

3、Logger：日志

十七、控制级实体类

1. TaskData：保存了执行plan和上一个模块的执行结果，以及当前模块的执行参数
2. ExecutionPlan：执行plan
3. Node：代表了一个app的DStreams流
4. Port：topic
5. 程序入口

SystemEntry:根据app的名字加载对应的app

1. 流程图



二十、附注：

变量有两种风格，一种是public，另一种是private

Pom.xml通过配置一些集群的配置以及app的配置，下图的配置用来生成包含所有依赖项的jar包

