11.Spark Streaming与Kafka、Flume

In [9]:
```
# spark配置信息
from pyspark import SparkConf
import pyspark

SPARK_APP_NAME = "processing_online_data"
SPARK_URL = "yarn"

conf = SparkConf()    # 创建spark config对象
config = (
    ("spark.app.name", SPARK_APP_NAME),    # 设置启动的spark的app名称，没有提供，将随
    ("spark.executor.memory", "2g"),    # 设置该app启动时占用的内存用量，默认1g
    ("spark.executor.cores", "2"),   # 设置spark executor使用的CPU核心数
    ("spark.executor.instances", 1)    # 设置spark executor数量，yarn时起作用
)
# 查看更详细配置及说明：https://spark.apache.org/docs/latest/configuration.html
#
conf.setAll(config)

sc = pyspark.SparkContext(master=SPARK_URL, conf=conf)
```

StreamingContext同sqlContext一样，这里作为流式计算的上下文

In [10]:
```
# 注意：初次安装并运行时，由于使用了kafka，所以会自动下载一系列的依赖jar包，会耗费一定

from pyspark.streaming.kafka import KafkaUtils
from pyspark.streaming import StreamingContext

# 第2个参数表示 程序运行间隔时间
ssc = StreamingContext(sc, 2)

kafkaParams = {"metadata.broker.list": "192.168.58.100:9092"}
dstream = KafkaUtils.createDirectStream(ssc, ["mytopic"], kafkaParams)
```

DStream (https://spark.apache.org/docs/2.2.2/api/python/pyspark.streaming.html?highlight=dstream#pyspark.streaming.DStream)

Discretized Stream是Spark Streaming的基础抽象，代表持续性的数据流和经过各种Spark算子操作后的结果数据流。

在内部实现上，DStream是一系列连续的RDD来表示。每个RDD含有一段时间间隔内的数据

In [11]:
```
def m(e):
    print("map")    # 不会出现在当前进程
    return e[1].split(",")

i = 100
j = [100]

def f(rdd):
    # 注意！注意！注意！
    # foreachRDD内部的打印信息会在当前进程出现，说明该方法是在当前进程执行的
    # 在内部可以访问全局变量

    # DStream的绝大部分方法都是返回一个新的DStream对象
    # 只有当调用foreachRDD后，才会真正实现“注册”前面一系列的逻辑，且只有在这之后才可
    # 因此可以把foreachRDD当做一系列操作的结束，在这里面做最终的处理
    global i

    print("foreachRDD", i, j, rdd.collect())

    i += 1
    j[0] += 1
```

In [12]:
```
dstream.map(m).foreachRDD(f)
```

In [13]:
```
ssc.start()
# ssc start后，才会真正启动DStream，不断的获取数据，进行前面在DStream已经写好的一系列
```

```
foreachRDD 100 [100] []
foreachRDD 101 [101] []
foreachRDD 102 [102] []
foreachRDD 103 [103] []
foreachRDD 104 [104] []
foreachRDD 105 [105] []
foreachRDD 106 [106] []
foreachRDD 107 [107] []
foreachRDD 108 [108] []
foreachRDD 109 [109] []
foreachRDD 110 [110] []
foreachRDD 111 [111] []
foreachRDD 112 [112] []
foreachRDD 113 [113] []
foreachRDD 114 [114] []
foreachRDD 115 [115] []
foreachRDD 116 [116] []
```

In [14]:
```
ssc.stop()
# stop后，就不能再继续使用，ssc不能复用
```

In [ ]:
```

```