离线埋点日志处理

日志是有固定格式的，使用正则匹配将日志转换为我们处理过的spark sql dataframe

前面已经把点击流日志存储hdfs中，这里再实现对曝光日志的采集，但注意曝光日志只需要发送到hdfs即可

/root/bigdata/flume/conf/exposure_log_hdfs.properties：

```
# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = exec
a1.sources.r1.command = tail -F /root/meiduoSourceCode/logs/exposure.log
a1.sources.r1.channels = c1

a1.sources.r1.interceptors = t1
a1.sources.r1.interceptors.t1.type = timestamp

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

a1.sinks.k1.type = hdfs
a1.sinks.k1.channel = c1
a1.sinks.k1.hdfs.path = hdfs://localhost:9000/project2-meiduo-rs/logs/exposure/%y
-%m-%d
as.sinks.k1.hdfs.userLocalTimeStamp = true
a1.sinks.k1.hdfs.filePrefix = exposure-
a1.sinks.k1.hdfs.fileType = DataStream
a1.sinks.k1.hdfs.writeFormat = Text
a1.sinks.k1.hdfs.round = true
a1.sinks.k1.hdfs.roundValue = 10
a1.sinks.k1.hdfs.roundUnit = minute
```

启动flume： `flume-ng agent -f /root/bigdata/flume/conf/exposure_log_hdfs.properties -n a1`

In [1]:
```
import os
# 配置pyspark和spark driver运行时 使用的python解释器
JAVA_HOME = '/root/bigdata/jdk'
PYSPARK_PYTHON = '/miniconda2/envs/py365/bin/python'
# 当存在多个版本时，不指定很可能会导致出错
os.environ['PYSPARK_PYTHON'] = PYSPARK_PYTHON
os.environ['PYSPARK_DRIVER_PYTHON'] = PYSPARK_PYTHON
os.environ['JAVA_HOME'] = JAVA_HOME
# 配置spark信息
from pyspark import SparkConf
from pyspark.sql import SparkSession

SPARK_APP_NAME = "processingSKUMetadata"
SPARK_URL = "spark://192.168.58.100:7077"

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，将随机
    ("spark.executor.memory", "2g"),    # 设置该app启动时占用的内存用量，默认1g，指一
    ("spark.master", SPARK_URL),    # spark master的地址
    ("spark.executor.cores", "2"),    # 设置spark executor使用的CPU核心数，指一台虚拟机
    ("hive.metastore.uris", "thrift://localhost:9083"),    # 配置hive元数据的访问，否则

    # 以下三项配置，可以控制执行器数量
#     ("spark.dynamicAllocation.enabled", True),
#     ("spark.dynamicAllocation.initialExecutors", 1),    # 1个执行器
#     ("spark.shuffle.service.enabled", True)
#   ('spark.sql.pivotMaxValues', '99999'),  # 当需要pivot DF，且值很多时，需要修改，默
)
# 查看更详细配置及说明：https://spark.apache.org/docs/latest/configuration.html

conf.setAll(config)

# 利用config对象，创建spark session
spark = SparkSession.builder.config(conf=conf).enableHiveSupport().getOrCreate()
```

In [2]:
```
!hadoop fs -ls /meiduo_mall/logs/click-trace
```

```
Found 2 items
drwxr-xr-x   - root supergroup          0 2020-12-23 23:04 /meiduo_mall/logs/click-trac
e/20-12-23
drwxr-xr-x   - root supergroup          0 2020-12-24 11:36 /meiduo_mall/logs/click-trac
e/20-12-24
```

In [7]:
```python
date = '20-12-24'
click_trace = spark.read.csv('/meiduo_mall/logs/click-trace/%s'%date)
click_trace.show(truncate=False)
```

```
+--------------------------------------------------------------------------------
-------------------------------------------------------------------+
|_c0
|
+--------------------------------------------------------------------------------
-------------------------------------------------------------------+
|2018/12/01 02:41:57: exposure_timesteamp<1543603102> exposure_loc<detail> timesteamp
<1543603317> behavior<pv> uid<1> sku_id<1> cate_id<1> stay_time<60>|
|2018/12/01 02:43:18: exposure_timesteamp<1543603102> exposure_loc<detail> timesteamp
<1543603398> behavior<pv> uid<1> sku_id<1> cate_id<1> stay_time<60>|
|2018/12/01 02:43:19: exposure_timesteamp<1543603102> exposure_loc<detail> timesteamp
<1543603399> behavior<pv> uid<1> sku_id<1> cate_id<1> stay_time<60>|
|2018/12/01 02:43:20: exposure_timesteamp<1543603102> exposure_loc<detail> timesteamp
<1543603400> behavior<pv> uid<1> sku_id<1> cate_id<1> stay_time<60>|
|2018/12/01 02:43:21: exposure_timesteamp<1543603102> exposure_loc<detail> timesteamp
<1543603401> behavior<pv> uid<1> sku_id<1> cate_id<1> stay_time<60>|
|2018/12/01 13:13:28: exposure_timesteamp<1543641203> exposure_loc<detail> timesteamp
<1543641208> behavior<pv> uid<1> sku_id<1> cate_id<1> stay_time<60>|
|2018/12/01 13:16:01: exposure_timesteamp<1543641203> exposure_loc<detail> timesteamp
```

In [8]:
```
1 date = '20-12-24'
2 exposure = spark.read.csv('/meiduo_mall/logs/exposure/%s'%date)
3 exposure.show(truncate=False)
```

```
+-----------------------------------------------------------------------------------------------
---------------+
|_c0
|
+-----------------------------------------------------------------------------------------------
---------------+
|2018/11/30 03:19:41: exposure_timesteamp<1543519181> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/11/30 13:18:13: exposure_timesteamp<1543555093> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/11/30 13:18:13: exposure_timesteamp<1543555093> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/12/01 01:47:59: exposure_timesteamp<1543600079> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/12/01 01:49:20: exposure_timesteamp<1543600079> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/12/01 01:52:53: exposure_timesteamp<1543600373> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/12/01 01:57:47: exposure_timesteamp<1543600666> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2020/12/23 21:21:21: exposure_timesteamp<1608729679> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2020/12/24 11:09:20: exposure_timesteamp<1608779359> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2020/12/24 11:09:20: exposure_timesteamp<1608779359> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/11/30 03:19:41: exposure_timesteamp<1543519181> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/11/30 13:18:13: exposure_timesteamp<1543555093> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/11/30 13:18:13: exposure_timesteamp<1543555093> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/12/01 01:47:59: exposure_timesteamp<1543600079> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/12/01 01:49:20: exposure_timesteamp<1543600079> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/12/01 01:52:53: exposure_timesteamp<1543600373> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2018/12/01 01:57:47: exposure_timesteamp<1543600666> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2020/12/23 21:21:21: exposure_timesteamp<1608729679> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2020/12/24 11:09:20: exposure_timesteamp<1608779359> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
|2020/12/24 11:09:20: exposure_timesteamp<1608779359> exposure_loc<detail> uid<1> sku_i
d<1> cate_id<1>|
+-----------------------------------------------------------------------------------------------
---------------+
```

In [39]:

```
# 延伸学习：1.正则表达式匹配 2.日志如何转化spark sql dataframe
# import re
# s = '2018/12/01 02:35:13: exposure_timesteamp<1543601846> exposure_loc<detail> times

# match = re.search("\
# exposure_timesteamp<(?P<exposure_timesteamp>.*?)> \
# exposure_loc<(?P<exposure_loc>.*?)> \
# timesteamp<(?P<timesteamp>.*?)> \
# behavior<(?P<behavior>.*?)> \
# uid<(?P<uid>.*?)> \
# sku_id<(?P<sku_id>.*?)> \
# cate_id<(?P<cate_id>.*?)> \
# stay_time<(?P<stay_time>.*?)>", s)
# match.groupdict()

# {'exposure_timesteamp': '1543601846',
#  'exposure_loc': 'detail',
#  'timesteamp': '1543602913',
#  'behavior': 'pv',
#  'uid': '1',
#  'sku_id': '1',
#  'cate_id': '1',
#  'stay_time': '60'}
```

In [12]:

```
# 点击日志的有用信息转换成df
import re
from pyspark.sql import Row

def map(row):
    match = re.search("\
exposure_timesteamp<(?P<exposure_timesteamp>.*?)> \
exposure_loc<(?P<exposure_loc>.*?)> \
timesteamp<(?P<timesteamp>.*?)> \
behavior<(?P<behavior>.*?)> \
uid<(?P<uid>.*?)> \
sku_id<(?P<sku_id>.*?)> \
cate_id<(?P<cate_id>.*?)> \
stay_time<(?P<stay_time>.*?)>", row._c0)

    row = Row(exposure_timesteamp=match.group("exposure_timesteamp"),
              exposure_loc=match.group("exposure_loc"),
              timesteamp=match.group("timesteamp"),
              behavior=match.group("behavior"),
              uid=match.group("uid"),
              sku_id=match.group("sku_id"),
              cate_id=match.group("cate_id"),
              stay_time=match.group("stay_time"))

    return row
click_trace.rdd.map(map).toDF().show()
```

```
+--------+-------+------------+-------------------+------+---------+----------+---+
|behavior|cate_id|exposure_loc|exposure_timesteamp|sku_id|stay_time|timesteamp|uid|
+--------+-------+------------+-------------------+------+---------+----------+---+
|      pv|      1|      detail|         1543603102|     1|       60|1543603317|  1|
|      pv|      1|      detail|         1543603102|     1|       60|1543603398|  1|
|      pv|      1|      detail|         1543603102|     1|       60|1543603399|  1|
|      pv|      1|      detail|         1543603102|     1|       60|1543603400|  1|
|      pv|      1|      detail|         1543603102|     1|       60|1543603401|  1|
|      pv|      1|      detail|         1543641203|     1|       60|1543641208|  1|
|      pv|      1|      detail|         1543641203|     1|       60|1543641361|  1|
|      pv|      1|      detail|         1608780903|     1|       60|1608780903|  1|
|      pv|      1|      detail|         1608780903|     1|       60|1608780903|  1|
|      pv|      1|      detail|         1608780903|     1|       60|1608780903|  1|
|      pv|      1|      detail|         1608780903|     1|       60|1608780903|  1|
|      pv|      1|      detail|         1543641203|     1|       60|1543641363|  1|
|      pv|      1|      detail|         1608729679|     1|       60|1608730353|  1|
|      pv|      1|      detail|         1608779359|     1|       60|1608779362|  1|
|      pv|      1|      detail|         1608779359|     1|       60|1608779362|  1|
|      pv|      1|      detail|         1608780822|     1|       60|1608780822|  1|
|      pv|      1|      detail|         1608780838|     1|       60|1608780838|  1|
|      pv|      1|      detail|         1543603102|     1|       60|1543603316|  1|
|      pv|      1|      detail|         1543603102|     1|       60|1543603316|  1|
|      pv|      1|      detail|         1543603102|     1|       60|1543603316|  1|
+--------+-------+------------+-------------------+------+---------+----------+---+
only showing top 20 rows
```

In [11]:
```
# 曝光日志的有用信息转换成df
import re
from pyspark.sql import Row

def map(row):
    match = re.search("\
exposure_timesteamp<(?P<exposure_timesteamp>.*?)> \
exposure_loc<(?P<exposure_loc>.*?)> \
uid<(?P<uid>.*?)> \
sku_id<(?P<sku_id>.*?)> \
cate_id<(?P<cate_id>.*?)>", row._c0)
    match.group

    row = Row(exposure_timesteamp=match.group("exposure_timesteamp"),
              exposure_loc=match.group("exposure_loc"),
              uid=match.group("uid"),
              sku_id=match.group("sku_id"),
              cate_id=match.group("cate_id"))

    return row
exposure.rdd.map(map).toDF().show()
```

```
+-------+------------+------------------+------+---+
|cate_id|exposure_loc|exposure_timesteamp|sku_id|uid|
+-------+------------+------------------+------+---+
|      1|      detail|        1543519181|     1|  1|
|      1|      detail|        1543555093|     1|  1|
|      1|      detail|        1543555093|     1|  1|
|      1|      detail|        1543600079|     1|  1|
|      1|      detail|        1543600079|     1|  1|
|      1|      detail|        1543600373|     1|  1|
|      1|      detail|        1543600666|     1|  1|
|      1|      detail|        1608729679|     1|  1|
|      1|      detail|        1608779359|     1|  1|
|      1|      detail|        1608779359|     1|  1|
|      1|      detail|        1543519181|     1|  1|
|      1|      detail|        1543555093|     1|  1|
|      1|      detail|        1543555093|     1|  1|
|      1|      detail|        1543600079|     1|  1|
|      1|      detail|        1543600079|     1|  1|
|      1|      detail|        1543600373|     1|  1|
```

曝光日志和点击流日志对齐

利用点击流日志中行为是"pv"的数据同时 cate_id|exposure_loc|exposure_timesteamp|sku_id|uid 一一对应的数据，最终就能得出：所有曝光的商品中，哪些商品被用户浏览了，哪些没有被浏览