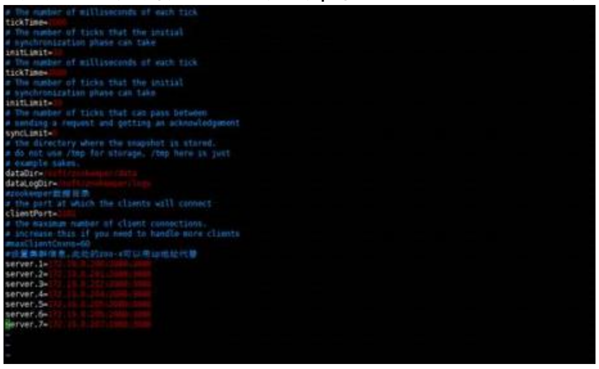
zookeeper和kafka的配置

2020年7月30日　　13:49

每一台机器都要执行：
zookeeper-3.4.6
kafka 2.11
安装和配置zookeeper:
1前期的java准备： https://www.cnblogs.com/ysocean/p/9860529.html
2 cd zookeeper/conf/   & cp zoo_sample.cfg   zoo.cfg
3 更改server参数们，把集群的所有ip都写上，注意server.*后面的数字不要重复

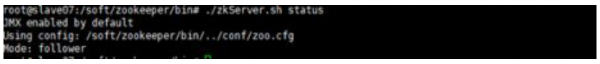


```
# The number of milliseconds of each tick
tickTime=[illegible]
# The number of ticks that the initial
# synchronization phase can take
initLimit=[illegible]
# The number of milliseconds of each tick
tickTime=[illegible]
# The number of ticks that the initial
# synchronization phase can take
initLimit=[illegible]
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=[illegible]
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=[illegible]
dataLogDir=[illegible]
#zookeeper[illegible]
# the port at which the clients will connect
clientPort=[illegible]
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#[illegible]
server.1=[illegible]
server.2=[illegible]
server.3=[illegible]
server.4=[illegible]
server.5=[illegible]
server.6=[illegible]
server.7=[illegible]
```

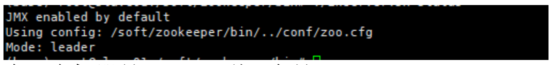
4 cd zookeeper/data & vim myid
5 把本机ip对应的server.*那个数字写入myid，只写这一个数字。
6 配置环境变量如下：

```
[illegible]
```

7 source /etc/profile
8 启动zookeeper:./zkServer.sh start
9 查看zookeeper状态：./zkServer.sh status



```
root@slave07:/soft/zookeeper/bin# ./zkServer.sh status
JMX enabled by default
Using config: /soft/zookeeper/bin/../conf/zoo.cfg
Mode: follower
```



```
JMX enabled by default
Using config: /soft/zookeeper/bin/../conf/zoo.cfg
Mode: leader
```

有一台机器是leader,剩下全是follwer

10 关闭防火墙**systemctl stop firewalld**

service iptables stop

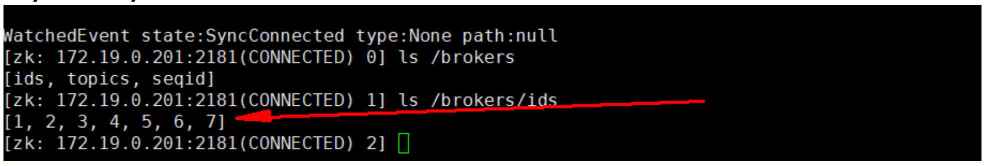
11 任意连接一台zookeeper查看状态:

./zkCli.sh -server 172.19.0.201:2181

Ls /

Ls /

Ls /brokers/ids

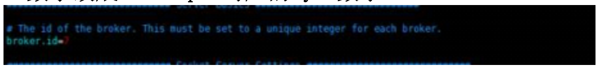


```
WatchedEvent state:SyncConnected type:None path:null
[zk: 172.19.0.201:2181(CONNECTED) 0] ls /brokers
[ids, topics, seqid]
[zk: 172.19.0.201:2181(CONNECTED) 1] ls /brokers/ids
[1, 2, 3, 4, 5, 6, 7]
[zk: 172.19.0.201:2181(CONNECTED) 2]
```

箭头2这样是正常的，集群现在有1-7 共7台服务器，一定要先启动kafka后验证这个。

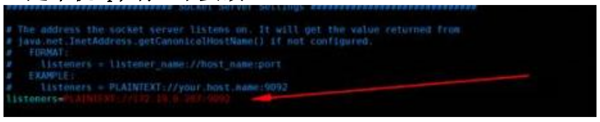
安装和配置kafka:
1 vim config/ server.properties

2 数字改成zookeeper对应的myid数字



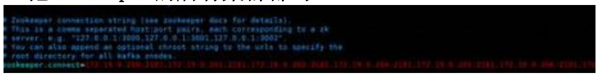
```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=7
```

3 是本机ip,端口不要改



```
######################### Socket Server Settings #########################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://172.19.0.207:9092
```

4 把zookeeper的所有集群都写上



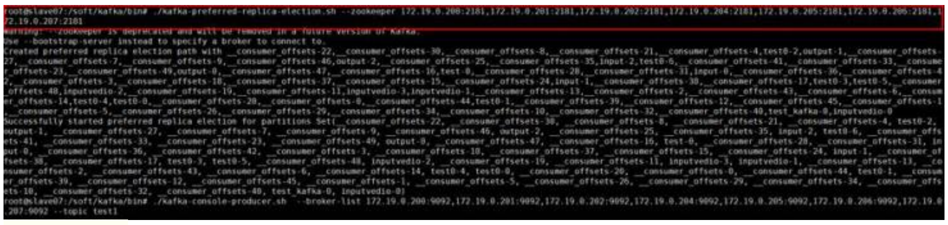
```
# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=172.19.0.200:2181,172.19.0.201:2181,172.19.0.202:2181,172.19.0.204:2181,172.19.0.205:2181,172.19.0.206:2181,1[illegible]
```

5启动kafka ./bin/kafka-server-start.sh ./config/server.properties

6 之前启动过kafka的机器查看config/ server.properties log_dirs对应目录下的meta. Properties文件，确定id是不是和broker.id一致

验证kafka:

1先负载均衡，每一台机器都要加上,这个最好不要一次执行一批，否则有失败的可能。失败了的话。从新执行就行了：

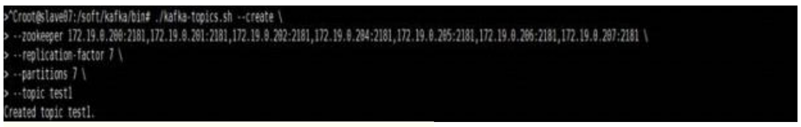


```
root@slave07:/soft/kafka/bin# ./kafka-preferred-replica-election.sh --zookeeper 172.19.0.200:2181,172.19.0.201:2181,172.19.0.202:2181,172.19.0.204:2181,172.19.0.205:2181,172.19.0.206:2181,1
72.19.0.207:2181
Warning: --zookeeper is deprecated and will be removed in a future version of Kafka.
Use --bootstrap-server instead to specify a broker to connect to.
Created preferred replica election path with __consumer_offsets-22,__consumer_offsets-30,__consumer_offsets-8,__consumer_offsets-21,__consumer_offsets-4,test0-2,output-1,__consumer_offsets-
27,__consumer_offsets-7,__consumer_offsets-9,__consumer_offsets-46,output-2,__consumer_offsets-25,__consumer_offsets-35,input-2,test0-6,__consumer_offsets-41,__consumer_offsets-33,__consume
r_offsets-23,__consumer_offsets-49,output-0,__consumer_offsets-47,__consumer_offsets-16,test-0,__consumer_offsets-28,__consumer_offsets-31,input-0,__consumer_offsets-36,__consumer_offsets-4
2,__consumer_offsets-3,__consumer_offsets-18,__consumer_offsets-37,__consumer_offsets-15,__consumer_offsets-24,input-1,__consumer_offsets-38,__consumer_offsets-17,test0-3,test0-5,__consumer
_offsets-48,inputvedio-2,__consumer_offsets-19,__consumer_offsets-11,inputvedio-3,inputvedio-1,__consumer_offsets-13,__consumer_offsets-2,__consumer_offsets-43,__consumer_offsets-6,__consum
er_offsets-14,test0-4,test0-0,__consumer_offsets-20,__consumer_offsets-0,__consumer_offsets-44,test0-1,__consumer_offsets-39,__consumer_offsets-12,__consumer_offsets-45,__consumer_offsets-1
,__consumer_offsets-5,__consumer_offsets-26,__consumer_offsets-29,__consumer_offsets-34,__consumer_offsets-10,__consumer_offsets-32,__consumer_offsets-40,test_kafka-0,inputvedio-0
Successfully started preferred replica election for partitions Set(__consumer_offsets-22, __consumer_offsets-30, __consumer_offsets-8, __consumer_offsets-21, __consumer_offsets-4, test0-2,
output-1, __consumer_offsets-27, __consumer_offsets-7, __consumer_offsets-9, __consumer_offsets-46, output-2, __consumer_offsets-25, __consumer_offsets-35, input-2, test0-6, __consumer_offs
ets-41, __consumer_offsets-33, __consumer_offsets-23, __consumer_offsets-49, output-0, __consumer_offsets-47, __consumer_offsets-16, test-0, __consumer_offsets-28, __consumer_offsets-31, in
put-0, __consumer_offsets-36, __consumer_offsets-42, __consumer_offsets-3, __consumer_offsets-18, __consumer_offsets-37, __consumer_offsets-15, __consumer_offsets-24, input-1, __consumer_of
fsets-38, __consumer_offsets-17, test0-3, test0-5, __consumer_offsets-48, inputvedio-2, __consumer_offsets-19, __consumer_offsets-11, inputvedio-3, inputvedio-1, __consumer_offsets-13, __co
nsumer_offsets-2, __consumer_offsets-43, __consumer_offsets-6, __consumer_offsets-14, test0-4, test0-0, __consumer_offsets-20, __consumer_offsets-0, __consumer_offsets-44, test0-1, __consum
er_offsets-39, __consumer_offsets-12, __consumer_offsets-45, __consumer_offsets-1, __consumer_offsets-5, __consumer_offsets-26, __consumer_offsets-29, __consumer_offsets-34, __consumer_offs
ets-10, __consumer_offsets-32, __consumer_offsets-40, test_kafka-0, inputvedio-0)
root@slave07:/soft/kafka/bin# ./kafka-console-producer.sh --broker-list 172.19.0.200:9092,172.19.0.201:9092,172.19.0.202:9092,172.19.0.204:9092,172.19.0.205:9092,172.19.0.206:9092,172.19.0
.207:9092 --topic test1
```

2创建topic:

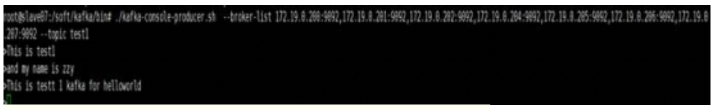
bin/kafka-topics.sh --create --zookeeper **172.19.0.200:2181,172.19.0.201:2181,172.19.0.202:2181,172.19.0.204:2181,172.19.0.205:2181,172.19.0.206:2181,172.19.0.207:2181** --replication-factor 3 --partitions 1 --topic test

创建名字为test的topic,备份为3，分区为1。Zookeeperip 是要把集群所有的ip都算上。



```
>^Croot@slave07:/soft/kafka/bin# ./kafka-topics.sh --create \
> --zookeeper 172.19.0.200:2181,172.19.0.201:2181,172.19.0.202:2181,172.19.0.204:2181,172.19.0.205:2181,172.19.0.206:2181,172.19.0.207:2181 \
> --replication-factor 7 \
> --partitions 7 \
> --topic test1
Created topic test1.
```

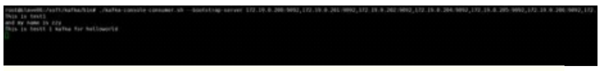
3 执行生产者生产数据，把所有ip都写上：



```
root@slave07:/soft/kafka/bin# ./kafka-console-producer.sh --broker-list 172.19.0.200:9092,172.19.0.201:9092,172.19.0.202:9092,172.19.0.204:9092,172.19.0.205:9092,172.19.0.206:9092,172.19.0
.207:9092 --topic test1
>This is test1
>and my name is zzy
>This is testt 1 kafka for helloworld
>
```

./kafka-console-producer.sh --broker-list 172.19.0.200:9092,172.19.0.201:9092,172.19.0.202:9092,172.19.0.204:9092,172.19.0.205:9092,172.19.0.206:9092,172.19.0.207:9092 --topic test1

4 执行消费者消费数据，把所有ip都写上:



```
root@slave08:/soft/kafka/bin# ./kafka-console-consumer.sh --bootstrap-server 172.19.0.200:9092,172.19.0.201:9092,172.19.0.202:9092,172.19.0.204:9092,172.19.0.205:9092,172.19.0.206:9092,172.
This is test1
and my name is zzy
This is testt 1 kafka for helloworld
```

./kafka-console-consumer.sh --bootstrap-server 172.19.0.200:9092,172.19.0.201:9092,172.19.0.202:9092,172.19.0.204:9092,172.19.0.205:9092,172.19.0.206:9092,172.19.0.207:9092 --topic test1

This is test1

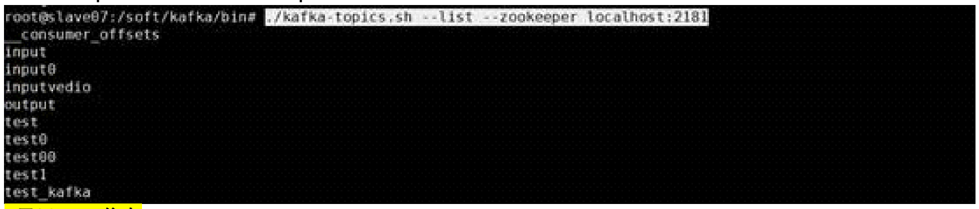
选举机制

kafka的使用和理解

2020年7月30日 13:55

1显示所有topic

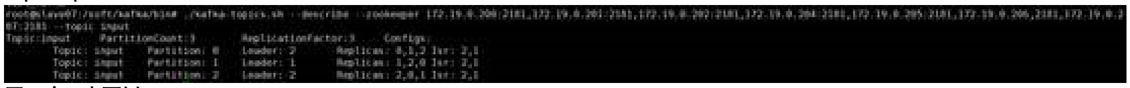
./kafka-topics.sh --list --zookeeper localhost:2181



```
root@slave07:/soft/kafka/bin# ./kafka-topics.sh --list --zookeeper localhost:2181
__consumer_offsets
input
input0
inputvedio
output
test
test0
test00
test1
test_kafka
```

2显示topic信息

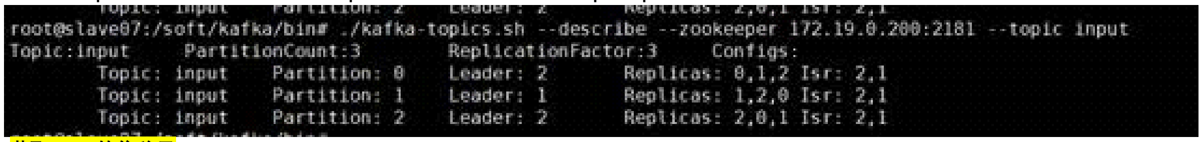
./kafka-topics.sh --describe --zookeeper 172.19.0.200:2181,172.19.0.201:2181,172.19.0.202:2181,172.19.0.204:2181,172.19.0.205:2181,172.19.0.206,2181,172.19.0.207:2181 --topic input



```
root@slave07:/soft/kafka/bin# ./kafka-topics.sh --describe --zookeeper 172.19.0.200:2181,172.19.0.201:2181,172.19.0.202:2181,172.19.0.204:2181,172.19.0.205:2181,172.19.0.206,2181,172.19.0.2
07:2181 --topic input
Topic:input     PartitionCount:3        ReplicationFactor:3     Configs:
        Topic: input    Partition: 0    Leader: 2       Replicas: 0,1,2 Isr: 2,1
        Topic: input    Partition: 1    Leader: 1       Replicas: 1,2,0 Isr: 2,1
        Topic: input    Partition: 2    Leader: 2       Replicas: 2,0,1 Isr: 2,1
```

写一个ip也可以：

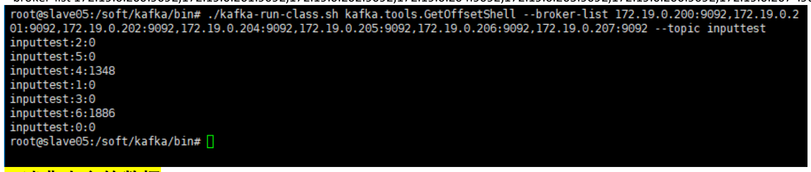
./kafka-topics.sh --describe --zookeeper 172.19.0.200:2181 --topic input



```
root@slave07:/soft/kafka/bin# ./kafka-topics.sh --describe --zookeeper 172.19.0.200:2181 --topic input
Topic:input     PartitionCount:3        ReplicationFactor:3     Configs:
        Topic: input    Partition: 0    Leader: 2       Replicas: 0,1,2 Isr: 2,1
        Topic: input    Partition: 1    Leader: 1       Replicas: 1,2,0 Isr: 2,1
        Topic: input    Partition: 2    Leader: 2       Replicas: 2,0,1 Isr: 2,1
```

获取topic的偏移量

--broker-list 172.19.0.200:9092,172.19.0.201:9092,172.19.0.202:9092,172.19.0.204:9092,172.19.0.205:9092,172.19.0.206:9092,172.19.0.207 :9092 --topic inputtest



```
root@slave05:/soft/kafka/bin# ./kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list 172.19.0.200:9092,172.19.0.2
01:9092,172.19.0.202:9092,172.19.0.204:9092,172.19.0.205:9092,172.19.0.206:9092,172.19.0.207:9092 --topic inputtest
inputtest:2:0
inputtest:5:0
inputtest:4:1348
inputtest:1:0
inputtest:3:0
inputtest:6:1886
inputtest:0:0
root@slave05:/soft/kafka/bin# 
```

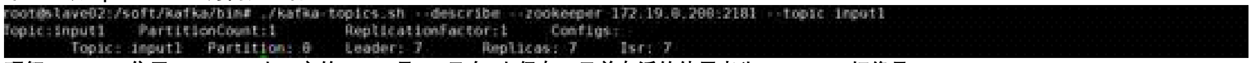
3 消费生产的数据

./kafka-console-consumer.sh --bootstrap-server 172.19.0.200:9092,172.19.0.201:9092,172.19.0.202:9092,172.19.0.204:9092,172.19.0.205:9092,172.19.0.206:9092,172.19.0.207:9092 --topic input0 --from-beginning

4 replication和partitions 理解

Replication 是集群中的同一数据有多少份。Partitions代表集群被分成几个区。

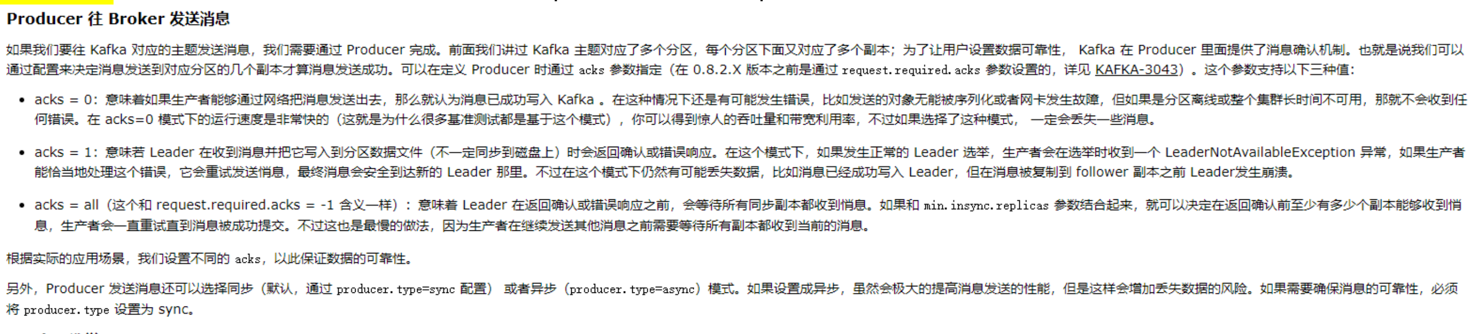
例如replication:3,partitions：7,那么集群7台机器时候，且brokers = 7时候，代表一个机器有1个区.。每个区有三分数据，可能数据的偏移量不同。以replication=7为例，查看：



```
root@slave02:/soft/kafka/bin# ./kafka-topics.sh --describe --zookeeper 172.19.0.200:2181 --topic input1
Topic:input1    PartitionCount:1        ReplicationFactor:1     Configs:
        Topic: input1   Partition: 0    Leader: 7       Replicas: 7     Isr: 7
```

理解：input1 位于partition0上，它的leader是7，只在7上保存，目前存活的使用者为7。Leader好像是partition0

5 基本可确定，ProducerRecord产生的记录，在创建kafka 的topic时候，只会给这个topic分配一个分区。



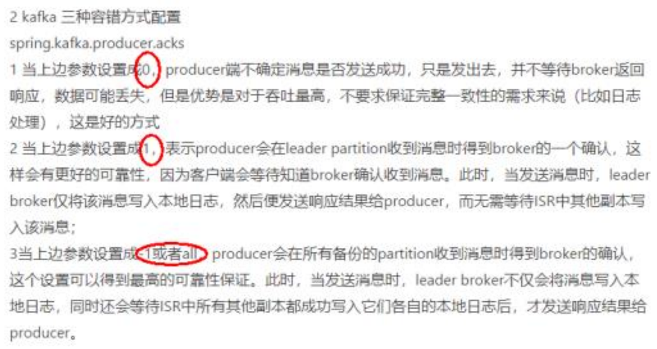
Producer 往 Broker 发送消息

如果我们要往 Kafka 对应的主题发送消息，我们需要通过 Producer 完成。前面我们讲过 Kafka 主题对应了多个分区，每个分区下面又对应了多个副本；为了让用户设置数据可靠性， Kafka 在 Producer 里面提供了消息确认机制。也就是说我们可以通过配置来决定消息发送到对应分区的几个副本才算消息发送成功。可以在定义 Producer 时通过 acks 参数指定（在 0.8.2.X 版本之前是通过 request.required.acks 参数设置的，详见 KAFKA-3043）。这个参数支持以下三种值：

- acks = 0：意味着如果生产者能够通过网络把消息发送出去，那么就认为消息已成功写入 Kafka 。在这种情况下还是有可能发生错误，比如发送的对象无能被序列化或者网卡发生故障，但如果是分区离线或整个集群长时间不可用，那就不会收到任何错误。在 acks=0 模式下的运行速度是非常快的（这就是为什么很多基准测试都是基于这个模式），你可以得到惊人的吞吐量和带宽利用率，不过如果选择了这种模式， 一定会丢失一些消息。
- acks = 1：意味若 Leader 在收到消息并把它写入到分区数据文件（不一定同步到磁盘上）时会返回确认或错误响应。在这个模式下，如果发生正常的 Leader 选举，生产者会在选举时收到一个 LeaderNotAvailableException 异常，如果生产者能恰当地处理这个错误，它会重试发送消息，最终消息会安全到达新的 Leader 那里。不过在这个模式下仍然有可能丢失数据，比如消息已经成功写入 Leader，但在消息被复制到 follower 副本之前 Leader发生崩溃。
- acks = all（这个和 request.required.acks = -1 含义一样）：意味着 Leader 在返回确认或错误响应之前，会等待所有同步副本都收到消息。如果和 min.insync.replicas 参数结合起来，就可以决定在返回确认前至少有多少个副本能够收到消息，生产者会一直重试直到消息被成功提交。不过这也是最慢的做法，因为生产者在继续发送其他消息之前需要等待所有副本都收到当前的消息。

根据实际的应用场景，我们设置不同的 acks，以此保证数据的可靠性。

另外，Producer 发送消息还可以选择同步（默认，通过 producer.type=sync 配置） 或者异步（producer.type=async）模式。如果设置成异步，虽然会极大的提高消息发送的性能，但是这样会增加丢失数据的风险。如果需要确保消息的可靠性，必须将 producer.type 设置为 sync。



2 kafka 三种容错方式配置

spring.kafka.producer.acks

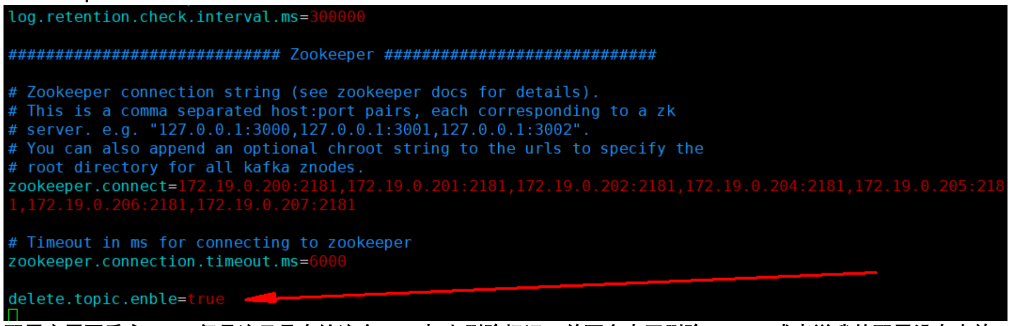
1 当上边参数设置成0，producer端不确定消息是否发送成功，只是发出去，并不等待broker返回响应，数据可能丢失，但是优势是对于吞吐量高，不要求保证完整一致性的需求来说（比如日志处理），这是好的方式

2 当上边参数设置成1，表示producer会在leader partition收到消息时得到broker的一个确认，这样会有更好的可靠性，因为客户端会等待知道broker确认收到消息。此时，当发送消息时，leader broker仅将该消息写入本地日志，然后便发送响应结果给producer，而无需等待ISR中其他副本写入该消息；

3当上边参数设置成-1或者all，producer会在所有备份的partition收到消息时得到broker的确认，这个设置可以得到最高的可靠性保证。此时，当发送消息时，leader broker不仅会将消息写入本地日志，同时还会等待ISR中所有其他副本都成功写入它们各自的本地日志后，才发送响应结果给producer。

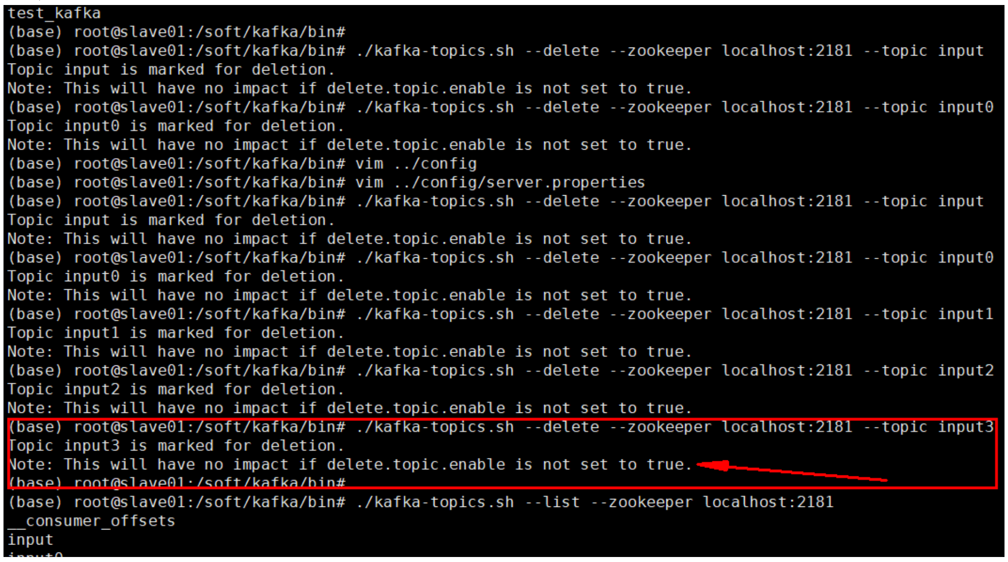
6 删除topic

6.1 给每一个kafka的配置文件配置：
delete.topic.enable=true



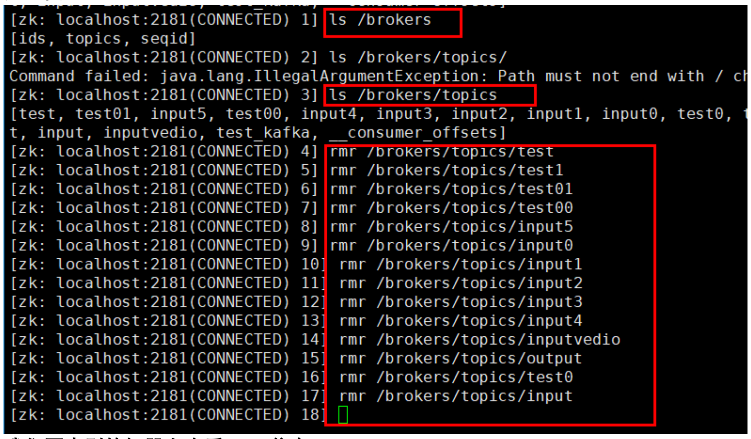
```
log.retention.check.interval.ms=300000

############################# Zookeeper #############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=172.19.0.200:2181,172.19.0.201:2181,172.19.0.202:2181,172.19.0.204:2181,172.19.0.205:218
1,172.19.0.206:2181,172.19.0.207:2181

# Timeout in ms for connecting to zookeeper
zookeeper.connection.timeout.ms=6000

delete.topic.enble=true
```

配置完需要重启kafka,但是这只是在给这个topic打上删除标记，并不会真正删除kafka，或者说我的配置没有奏效。因为我打上这个标记，仍然会出现



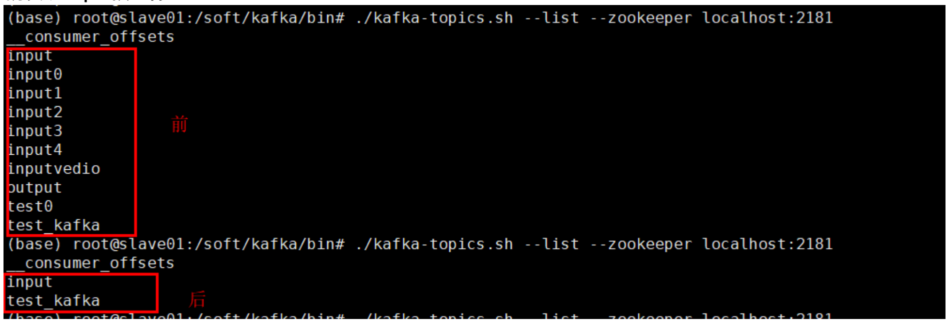
```
test_kafka
(base) root@slave01:/soft/kafka/bin#
(base) root@slave01:/soft/kafka/bin# ./kafka-topics.sh --delete --zookeeper localhost:2181 --topic input
Topic input is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
(base) root@slave01:/soft/kafka/bin# ./kafka-topics.sh --delete --zookeeper localhost:2181 --topic input0
Topic input0 is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
(base) root@slave01:/soft/kafka/bin# vim ../config
(base) root@slave01:/soft/kafka/bin# vim ../config/server.properties
(base) root@slave01:/soft/kafka/bin# ./kafka-topics.sh --delete --zookeeper localhost:2181 --topic input
Topic input is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
(base) root@slave01:/soft/kafka/bin# ./kafka-topics.sh --delete --zookeeper localhost:2181 --topic input0
Topic input0 is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
(base) root@slave01:/soft/kafka/bin# ./kafka-topics.sh --delete --zookeeper localhost:2181 --topic input1
Topic input1 is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
(base) root@slave01:/soft/kafka/bin# ./kafka-topics.sh --delete --zookeeper localhost:2181 --topic input2
Topic input2 is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
(base) root@slave01:/soft/kafka/bin# ./kafka-topics.sh --delete --zookeeper localhost:2181 --topic input3
Topic input3 is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
(base) root@slave01:/soft/kafka/bin#
(base) root@slave01:/soft/kafka/bin# ./kafka-topics.sh --list --zookeeper localhost:2181
__consumer_offsets
input
```

6.2要想彻底删除，则需要进入zookeeper进行删除。执行步骤是
6.2.1 执行任意一台集群机器的/zookeeper/bin/zkCli.sh
6.2.2 ls /brokers/topics
6.23 rmr /brokers/topics/*[topicname]*
如下图：



```
[zk: localhost:2181(CONNECTED) 1] ls /brokers
[ids, topics, seqid]
[zk: localhost:2181(CONNECTED) 2] ls /brokers/topics/
Command failed: java.lang.IllegalArgumentException: Path must not end with / ch
[zk: localhost:2181(CONNECTED) 3] ls /brokers/topics
[test, test01, input5, test00, input4, input3, input2, input1, input0, test0, t
t, input, inputvedio, test_kafka, __consumer_offsets]
[zk: localhost:2181(CONNECTED) 4] rmr /brokers/topics/test
[zk: localhost:2181(CONNECTED) 5] rmr /brokers/topics/test1
[zk: localhost:2181(CONNECTED) 6] rmr /brokers/topics/test01
[zk: localhost:2181(CONNECTED) 7] rmr /brokers/topics/test00
[zk: localhost:2181(CONNECTED) 8] rmr /brokers/topics/input5
[zk: localhost:2181(CONNECTED) 9] rmr /brokers/topics/input0
[zk: localhost:2181(CONNECTED) 10] rmr /brokers/topics/input1
[zk: localhost:2181(CONNECTED) 11] rmr /brokers/topics/input2
[zk: localhost:2181(CONNECTED) 12] rmr /brokers/topics/input3
[zk: localhost:2181(CONNECTED) 13] rmr /brokers/topics/input4
[zk: localhost:2181(CONNECTED) 14] rmr /brokers/topics/inputvedio
[zk: localhost:2181(CONNECTED) 15] rmr /brokers/topics/output
[zk: localhost:2181(CONNECTED) 16] rmr /brokers/topics/test0
[zk: localhost:2181(CONNECTED) 17] rmr /brokers/topics/input
[zk: localhost:2181(CONNECTED) 18]
```

我们再去别的机器上查看topic信息：
./kafka-topics.sh --list --zookeeper localhost:2181
就发现topic被删除了。



```
(base) root@slave01:/soft/kafka/bin# ./kafka-topics.sh --list --zookeeper localhost:2181
__consumer_offsets
input
input0
input1
input2
input3
input4
inputvedio
output
test0
test_kafka
(base) root@slave01:/soft/kafka/bin# ./kafka-topics.sh --list --zookeeper localhost:2181
__consumer_offsets
input
test_kafka
```

7 kafka Java的自动创建topic规定，所以我们在执行生产者的时候，就是创建了leader，默认1分区和1副本：

1. 如果kafka broker中的config/server.properties配置文件中配置了auto.create.topics.enable参数为true（默认值就是true），那么当生产者向一个尚未创建的topic发送消息时，会自动创建一个num.partitions（默认值为1）个分区和default.replication.factor（默认值为1）个副本的对应topic。不过我们一般不建议将auto.create.topics.enable参数设置为true，因为这个参数会影响topic的管理与维护。
2. 通过kafka提供的kafka-topics.sh脚本来创建，并且我们也建议通过这种方式（或者相关的变种方式）来创建topic。

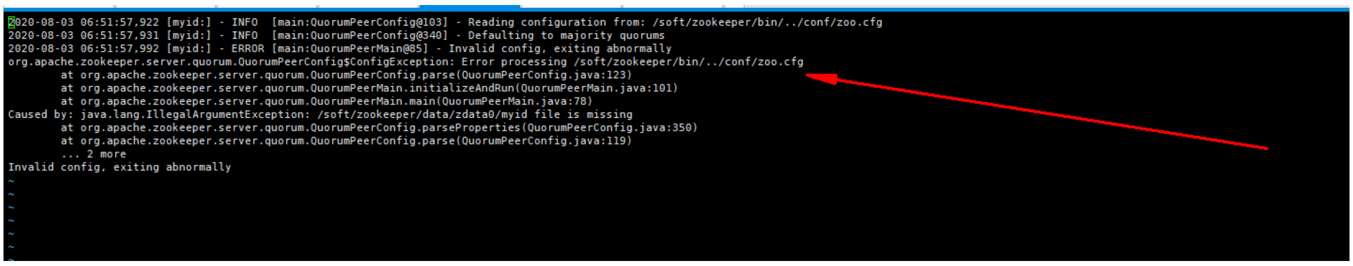
8 创建topic:

./bin/kafka-topics.sh --create --zookeeper zookeeperip:2181 --replication-factor 3 --partitions 1 --topic test

9zookeeper 停止服务的命令

./zkServer.sh stop

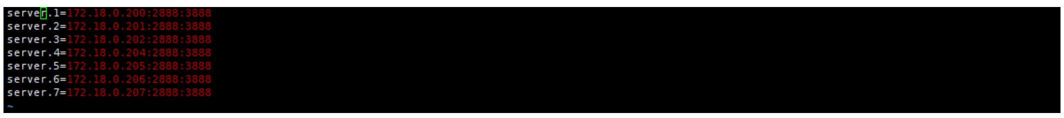
10zookeeper 启动错误



```
2020-08-03 06:51:57,922 [myid:] - INFO  [main:QuorumPeerConfig@103] - Reading configuration from: /soft/zookeeper/bin/../conf/zoo.cfg
2020-08-03 06:51:57,931 [myid:] - INFO  [main:QuorumPeerConfig@340] - Defaulting to majority quorums
2020-08-03 06:51:57,992 [myid:] - ERROR [main:QuorumPeerMain@85] - Invalid config, exiting abnormally
org.apache.zookeeper.server.quorum.QuorumPeerConfig$ConfigException: Error processing /soft/zookeeper/bin/../conf/zoo.cfg
        at org.apache.zookeeper.server.quorum.QuorumPeerConfig.parse(QuorumPeerConfig.java:123)
        at org.apache.zookeeper.server.quorum.QuorumPeerMain.initializeAndRun(QuorumPeerMain.java:101)
        at org.apache.zookeeper.server.quorum.QuorumPeerMain.main(QuorumPeerMain.java:78)
Caused by: java.lang.IllegalArgumentException: /soft/zookeeper/data/zdata0/myid file is missing
        at org.apache.zookeeper.server.quorum.QuorumPeerConfig.parseProperties(QuorumPeerConfig.java:350)
        at org.apache.zookeeper.server.quorum.QuorumPeerConfig.parse(QuorumPeerConfig.java:119)
        ... 2 more
Invalid config, exiting abnormally
~
~
~
~
~
~
```

对比和其他完好zookeeper的文件，感觉是端口占用问题。所以我们执行ls-o -i:2888 和3888找到对应端口杀掉，然后重新启动,发现还是不行。同样，之前好的机器也不行了。然后我们把所有的机器上的端口都杀死了。

将myid文件移动到了上述对应目录下。如果还不行，很大可能是zoo.cfg出错了！继续检查，发现不知怎么被改成了172.18.0.200...
更改好后重启zookeeper时候记得一定要把2181 ,2888 3888全部杀死重来。



```
server.1=172.18.0.200:2888:3888
server.2=172.18.0.201:2888:3888
server.3=172.18.0.202:2888:3888
server.4=172.18.0.204:2888:3888
server.5=172.18.0.205:2888:3888
server.6=172.18.0.206:2888:3888
server.7=172.18.0.207:2888:3888
~
```

5kafka写检查点出错会造成写卡住，这时候你要换个检查点目录

microBatch错误会造成kafka写检查点异常

6kafka参数解析

https://www.cnblogs.com/miracleYu/p/10213807.html

zookeeper无法正确暂停

netstat -nltp | grep 2181

找到端口杀进程

应用1的运行问题

2020年8月4日 9:53

1 卡住，什么也不提示：

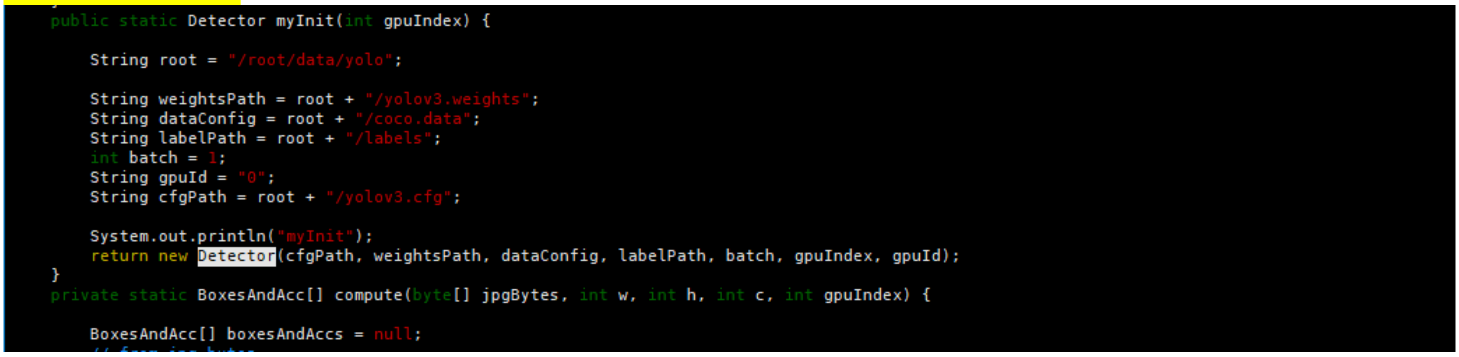
sort或者输出 kafka-size之后之后，更改检查点，更改输入输出为文件名称

检查./kafka-producer.sh生产者是否还产生数据。

如果生产数据，但是spark-sql-kafka无法生产数据，考虑是spark-sql-kafka的问题。那就把运行完好机器上的spark 和maven copy过去。

出现microexecption这种错误，检查点可能就坏了，需要更换。

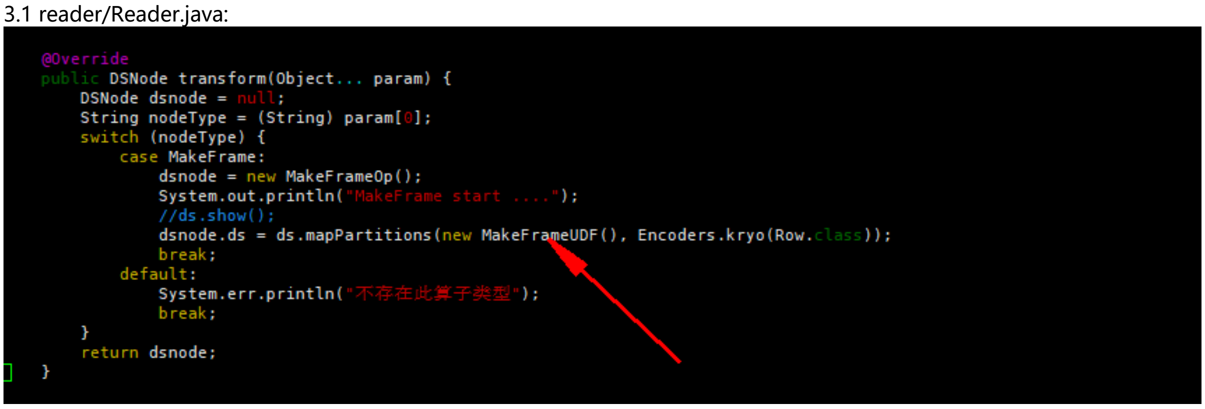
2 yolo数据在这里



```
    public static Detector myInit(int gpuIndex) {

        String root = "/root/data/yolo";

        String weightsPath = root + "/yolov3.weights";
        String dataConfig = root + "/coco.data";
        String labelPath = root + "/labels";
        int batch = 1;
        String gpuId = "0";
        String cfgPath = root + "/yolov3.cfg";

        System.out.println("myInit");
        return new Detector(cfgPath, weightsPath, dataConfig, labelPath, batch, gpuIndex, gpuId);
    }
    private static BoxesAndAcc[] compute(byte[] jpgBytes, int w, int h, int c, int gpuIndex) {

        BoxesAndAcc[] boxesAndAccs = null;
```

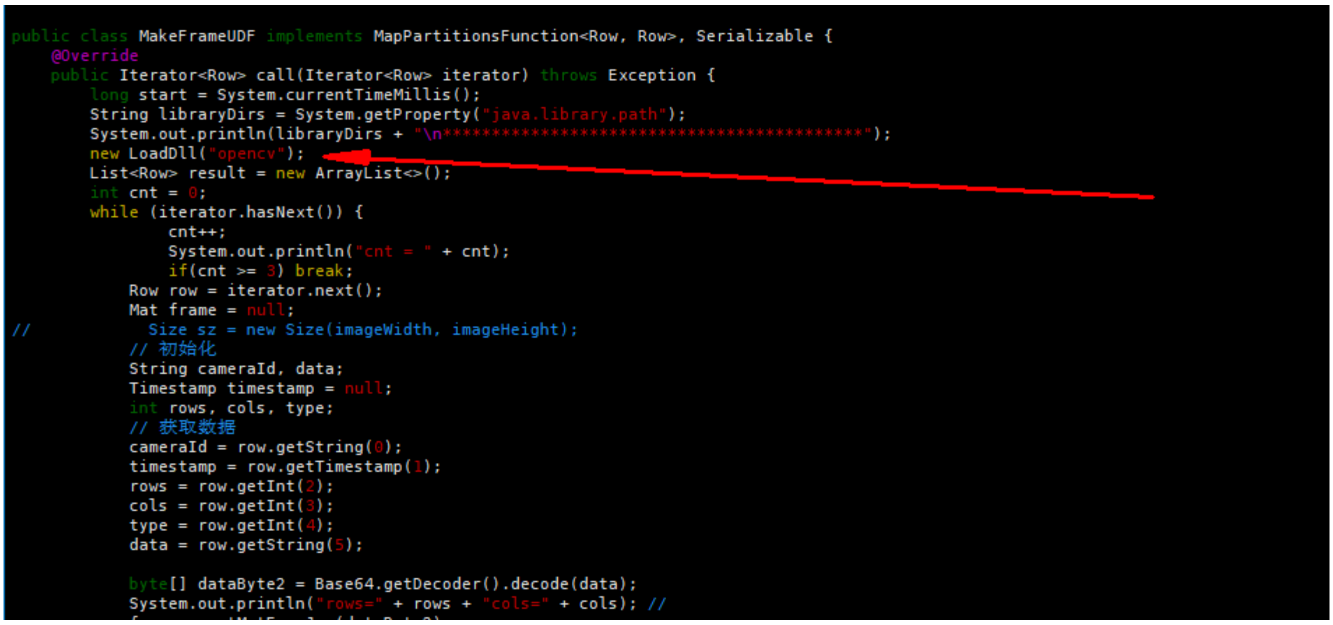
3 yolo执行过程：

3.1 reader/Reader.java:

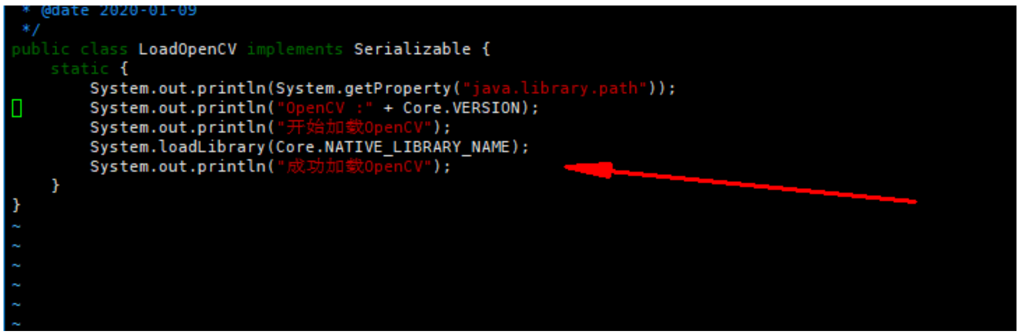


```
    @Override
    public DSNode transform(Object... param) {
        DSNode dsnode = null;
        String nodeType = (String) param[0];
        switch (nodeType) {
            case MakeFrame:
                dsnode = new MakeFrameOp();
                System.out.println("MakeFrame start ....");
                //ds.show();
                dsnode.ds = ds.mapPartitions(new MakeFrameUDF(), Encoders.kryo(Row.class));
                break;
            default:
                System.err.println("不存在此算子类型");
                break;
        }
        return dsnode;
    }
```

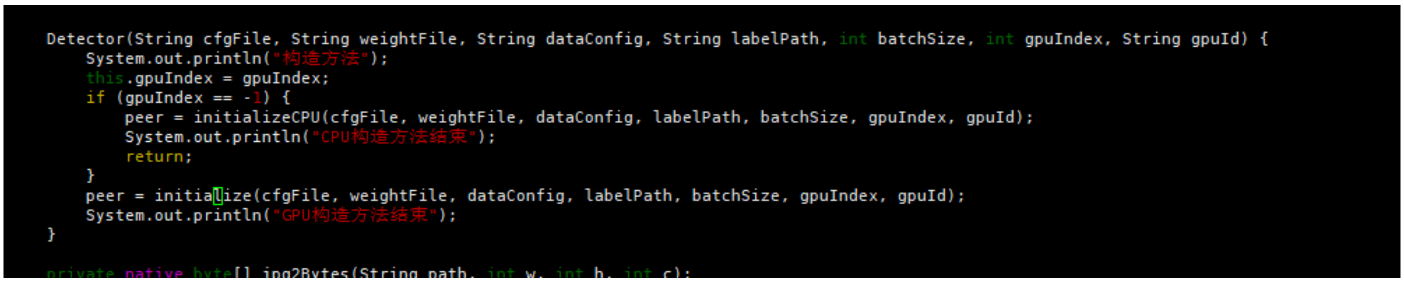
3.2 frame/MakeFrameUDF.java:



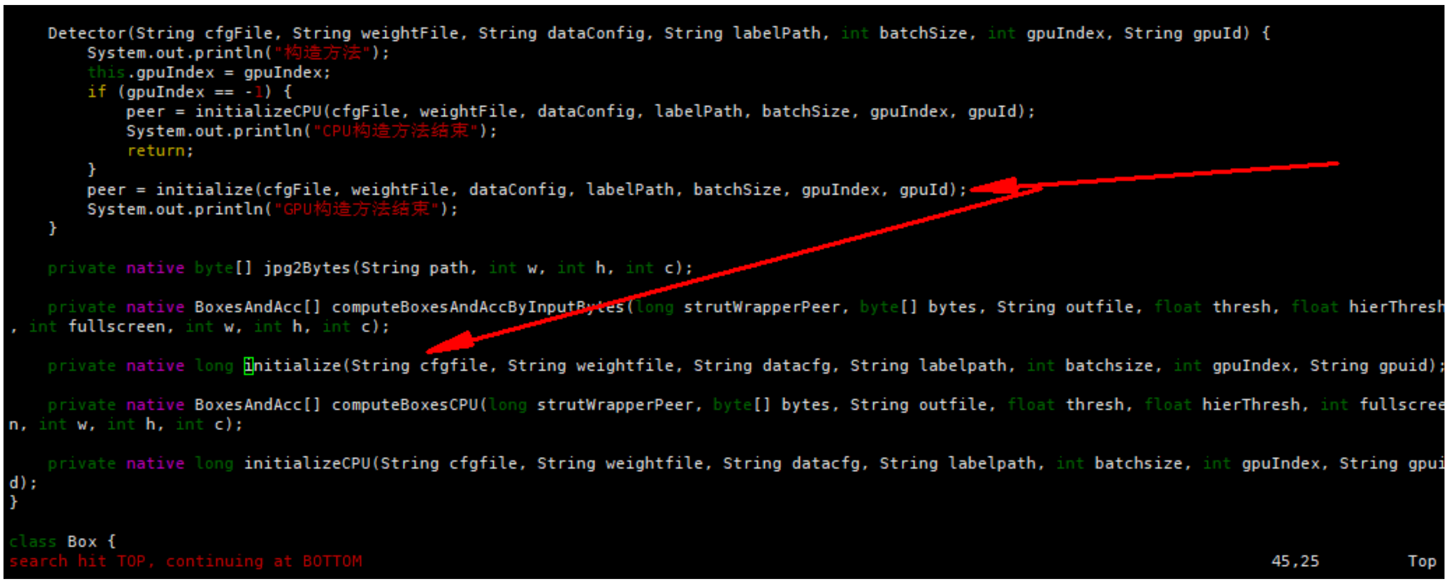
```
public class MakeFrameUDF implements MapPartitionsFunction<Row, Row>, Serializable {
    @Override
    public Iterator<Row> call(Iterator<Row> iterator) throws Exception {
        long start = System.currentTimeMillis();
        String libraryDirs = System.getProperty("java.library.path");
        System.out.println(libraryDirs + "\n*******************************************");
        new LoadDll("opencv");
        List<Row> result = new ArrayList<>();
        int cnt = 0;
        while (iterator.hasNext()) {
                cnt++;
                System.out.println("cnt = " + cnt);
                if(cnt >= 3) break;
            Row row = iterator.next();
            Mat frame = null;
//              Size sz = new Size(imageWidth, imageHeight);
            // 初始化
            String cameraId, data;
            Timestamp timestamp = null;
            int rows, cols, type;
            // 获取数据
            cameraId = row.getString(0);
            timestamp = row.getTimestamp(1);
            rows = row.getInt(2);
            cols = row.getInt(3);
            type = row.getInt(4);
            data = row.getString(5);

            byte[] dataByte2 = Base64.getDecoder().decode(data);
            System.out.println("rows=" + rows + "cols=" + cols); //
```



```
 * @date 2020-01-09
 */
public class LoadOpenCV implements Serializable {
    static {
        System.out.println(System.getProperty("java.library.path"));
        System.out.println("OpenCV :" + Core.VERSION);
        System.out.println("开始加载OpenCV");
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        System.out.println("成功加载OpenCV");
    }
}
~
~
~
~
~
~
```



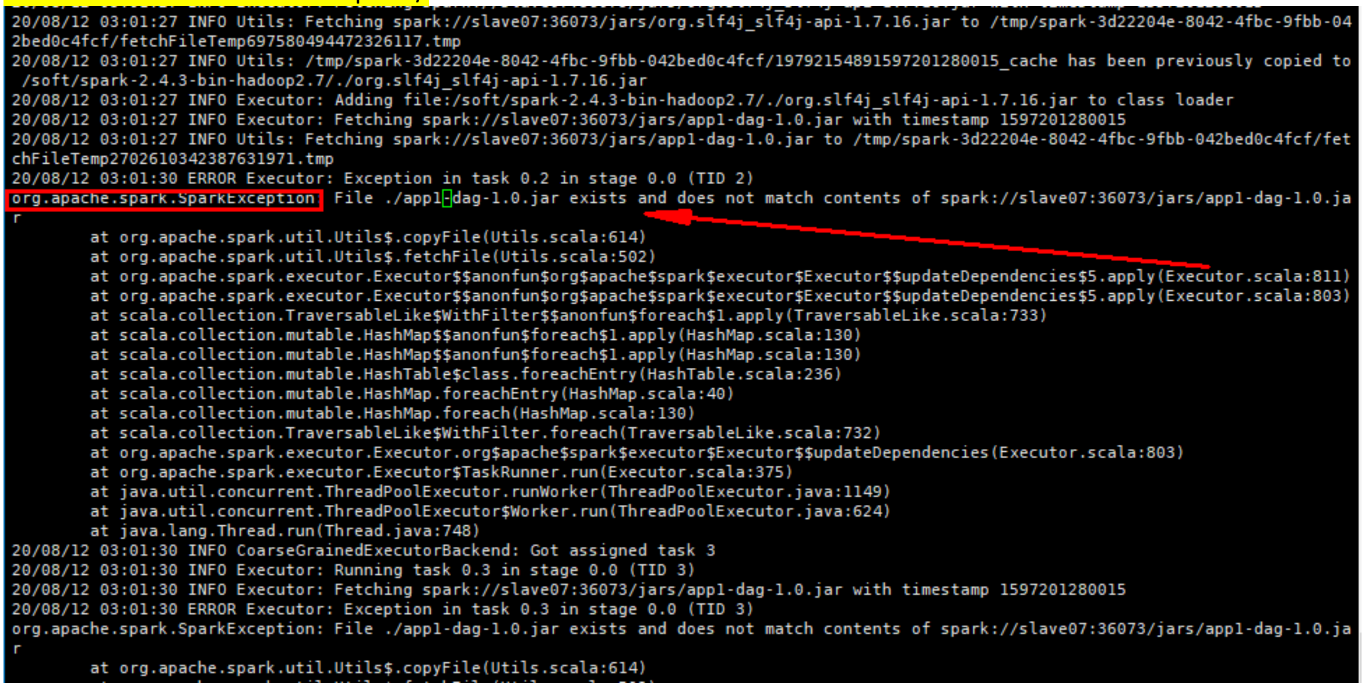
```
    Detector(String cfgFile, String weightFile, String dataConfig, String labelPath, int batchSize, int gpuIndex, String gpuId) {
        System.out.println("构造方法");
        this.gpuIndex = gpuIndex;
        if (gpuIndex == -1) {
            peer = initializeCPU(cfgFile, weightFile, dataConfig, labelPath, batchSize, gpuIndex, gpuId);
            System.out.println("CPU构造方法结束");
            return;
        }
        peer = initialize(cfgFile, weightFile, dataConfig, labelPath, batchSize, gpuIndex, gpuId);
        System.out.println("GPU构造方法结束");
    }

    private native byte[] jpg2Bytes(String path, int w, int h, int c);
```



```
    Detector(String cfgFile, String weightFile, String dataConfig, String labelPath, int batchSize, int gpuIndex, String gpuId) {
        System.out.println("构造方法");
        this.gpuIndex = gpuIndex;
        if (gpuIndex == -1) {
            peer = initializeCPU(cfgFile, weightFile, dataConfig, labelPath, batchSize, gpuIndex, gpuId);
            System.out.println("CPU构造方法结束");
            return;
        }
        peer = initialize(cfgFile, weightFile, dataConfig, labelPath, batchSize, gpuIndex, gpuId);
        System.out.println("GPU构造方法结束");
    }

    private native byte[] jpg2Bytes(String path, int w, int h, int c);

    private native BoxesAndAcc[] computeBoxesAndAccByInputBytes(long strutWrapperPeer, byte[] bytes, String outfile, float thresh, float hierThresh
, int fullscreen, int w, int h, int c);

    private native long initialize(String cfgfile, String weightfile, String datacfg, String labelpath, int batchsize, int gpuIndex, String gpuid);

    private native BoxesAndAcc[] computeBoxesCPU(long strutWrapperPeer, byte[] bytes, String outfile, float thresh, float hierThresh, int fullscree
n, int w, int h, int c);

    private native long initializeCPU(String cfgfile, String weightfile, String datacfg, String labelpath, int batchsize, int gpuIndex, String gpui
d);
}

class Box {
search hit TOP, continuing at BOTTOM                                                                                    45,25          Top
```

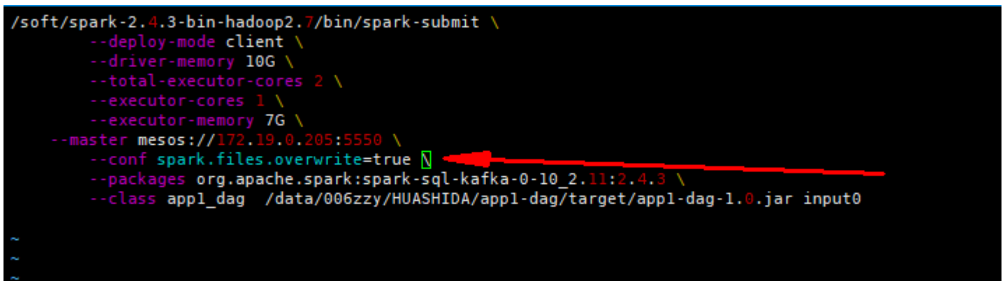
然后就在处理jni了。

3 org.apache.spark.SparkException: File ./app1-dag-1.0.jar exists and does not match contents of spark:/



```
20/08/12 03:01:27 INFO Utils: Fetching spark://slave07:36073/jars/org.slf4j_slf4j-api-1.7.16.jar to /tmp/spark-3d22204e-8042-4fbc-9fbb-04
2bed0c4fcf/fetchFileTemp6975804944723261l7.tmp
20/08/12 03:01:27 INFO Utils: /tmp/spark-3d22204e-8042-4fbc-9fbb-042bed0c4fcf/197921548915972012800l5_cache has been previously copied to
 /soft/spark-2.4.3-bin-hadoop2.7/./org.slf4j_slf4j-api-1.7.16.jar
20/08/12 03:01:27 INFO Executor: Adding file:/soft/spark-2.4.3-bin-hadoop2.7/./org.slf4j_slf4j-api-1.7.16.jar to class loader
20/08/12 03:01:27 INFO Executor: Fetching spark://slave07:36073/jars/appl-dag-1.0.jar with timestamp 1597201280015
20/08/12 03:01:27 INFO Utils: Fetching spark://slave07:36073/jars/appl-dag-1.0.jar to /tmp/spark-3d22204e-8042-4fbc-9fbb-042bed0c4fcf/fet
chFileTemp2702610342387631971.tmp
20/08/12 03:01:30 ERROR Executor: Exception in task 0.2 in stage 0.0 (TID 2)
org.apache.spark.SparkException: File ./appl-dag-1.0.jar exists and does not match contents of spark://slave07:36073/jars/appl-dag-1.0.ja
r
        at org.apache.spark.util.Utils$.copyFile(Utils.scala:614)
        at org.apache.spark.util.Utils$.fetchFile(Utils.scala:502)
        at org.apache.spark.executor.Executor$$anonfun$org$apache$spark$executor$Executor$$updateDependencies$5.apply(Executor.scala:811)
        at org.apache.spark.executor.Executor$$anonfun$org$apache$spark$executor$Executor$$updateDependencies$5.apply(Executor.scala:803)
        at scala.collection.TraversableLike$WithFilter$$anonfun$foreach$1.apply(TraversableLike.scala:733)
        at scala.collection.mutable.HashMap$$anonfun$foreach$1.apply(HashMap.scala:130)
        at scala.collection.mutable.HashMap$$anonfun$foreach$1.apply(HashMap.scala:130)
        at scala.collection.mutable.HashTable$class.foreachEntry(HashTable.scala:236)
        at scala.collection.mutable.HashMap.foreachEntry(HashMap.scala:40)
        at scala.collection.mutable.HashMap.foreach(HashMap.scala:130)
        at scala.collection.TraversableLike$WithFilter.foreach(TraversableLike.scala:732)
        at org.apache.spark.executor.Executor.org$apache$spark$executor$Executor$$updateDependencies(Executor.scala:803)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:375)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
20/08/12 03:01:30 INFO CoarseGrainedExecutorBackend: Got assigned task 3
20/08/12 03:01:30 INFO Executor: Running task 0.3 in stage 0.0 (TID 3)
20/08/12 03:01:30 INFO Executor: Fetching spark://slave07:36073/jars/appl-dag-1.0.jar with timestamp 1597201280015
20/08/12 03:01:30 ERROR Executor: Exception in task 0.3 in stage 0.0 (TID 3)
org.apache.spark.SparkException: File ./appl-dag-1.0.jar exists and does not match contents of spark://slave07:36073/jars/appl-dag-1.0.ja
r
        at org.apache.spark.util.Utils$.copyFile(Utils.scala:614)
```

网上说原因好像是空间不足，对提交命令增加：



```
/soft/spark-2.4.3-bin-hadoop2.7/bin/spark-submit \
        --deploy-mode client \
        --driver-memory 10G \
        --total-executor-cores 2 \
        --executor-cores 1 \
        --executor-memory 7G \
    --master mesos://172.19.0.205:5550 \
        --conf spark.files.overwrite=true \
        --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.3 \
        --class appl_dag  /data/006zzy/HUASHIDA/appl-dag/target/appl-dag-1.0.jar input0


~
~
~
```

应用一集群搭建

2020年8月4日 13:58

1 安装opencv

1.0安装ant 方便生成libopencv_java347.so

apt -get install ant

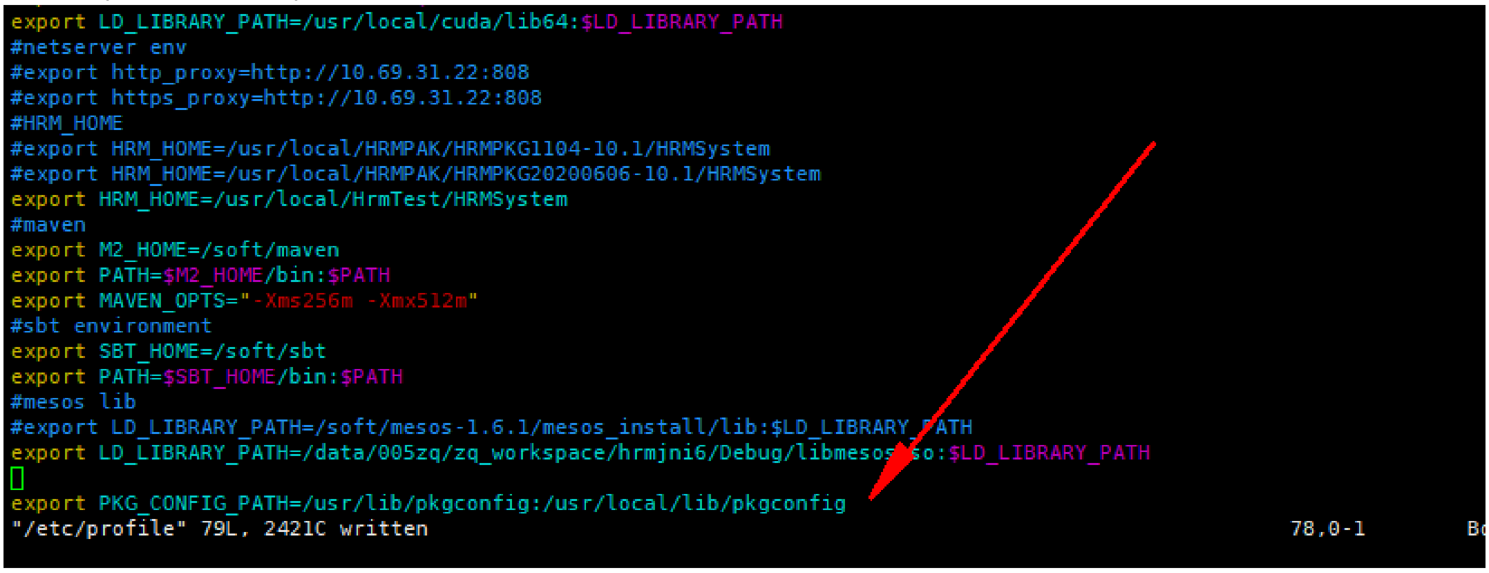
1.0安装ffpmeg

contrib移动到opencv里面，然后执行：

https://www.cnblogs.com/lvdongjie/p/11484575.html

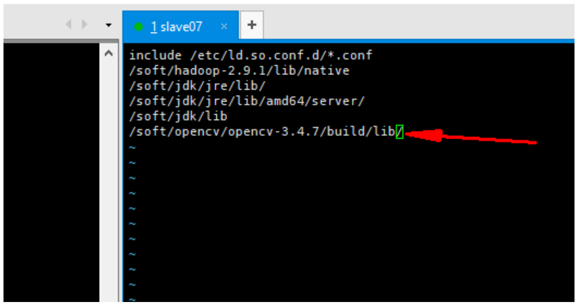
```
sudo apt-get install build-essential
sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev
sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev
libdc1394-22-dev


mkdir build


cmake -D CMAKE_INSTALL_PREFIX=/usr/local -D CMAKE_BUILD_TYPE=Release -D
OPENCV_EXTRA_MODULES_PATH=../opencv_contrib/modules ..

make -j8

sudo make install
```

1.1 检查opencv有关的/etc/profile



```
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
#netserver env
#export http_proxy=http://10.69.31.22:808
#export https_proxy=http://10.69.31.22:808
#HRM_HOME
#export HRM_HOME=/usr/local/HRMPAK/HRMPKG1104-10.1/HRMSystem
#export HRM_HOME=/usr/local/HRMPAK/HRMPKG20200606-10.1/HRMSystem
export HRM_HOME=/usr/local/HrmTest/HRMSystem
#maven
export M2_HOME=/soft/maven
export PATH=$M2_HOME/bin:$PATH
export MAVEN_OPTS="-Xms256m -Xmx512m"
#sbt environment
export SBT_HOME=/soft/sbt
export PATH=$SBT_HOME/bin:$PATH
#mesos lib
#export LD_LIBRARY_PATH=/soft/mesos-1.6.1/mesos_install/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/data/005zq/zq_workspace/hrmjni6/Debug/libmesos.so:$LD_LIBRARY_PATH

export PKG_CONFIG_PATH=/usr/lib/pkgconfig:/usr/local/lib/pkgconfig
"/etc/profile" 79L, 2421C written                                    78,0-1         Bo
```

1.2 vim /etc/ld.so.conf，这是刚才装opencv对应的build路径



```
include /etc/ld.so.conf.d/*.conf
/soft/hadoop-2.9.1/lib/native
/soft/jdk/jre/lib/
/soft/jdk/jre/lib/amd64/server/
/soft/jdk/lib
/soft/opencv/opencv-3.4.7/build/lib
~
~
~
~
~
~
~
~
~
~
~
```

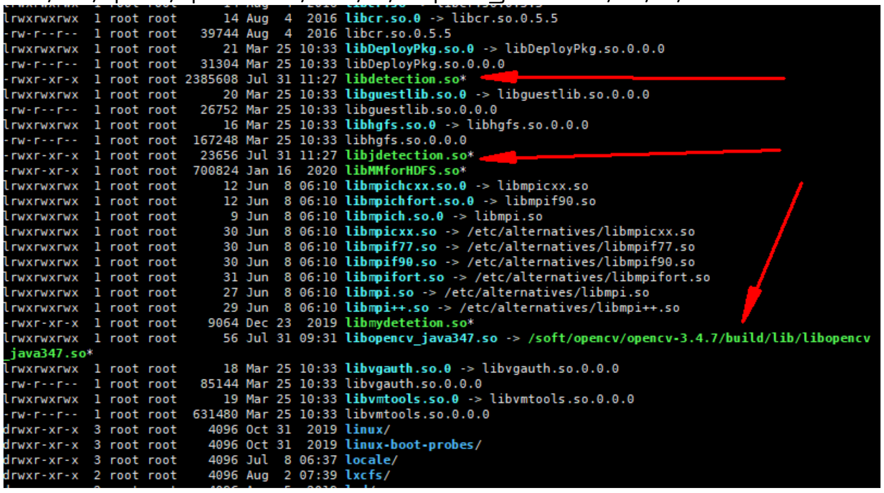
1.3 ldconfig & source /etc/profile

1.4 构造这三个路径,SCP之后文件的权限可能会更改，，记得检查

软连接：

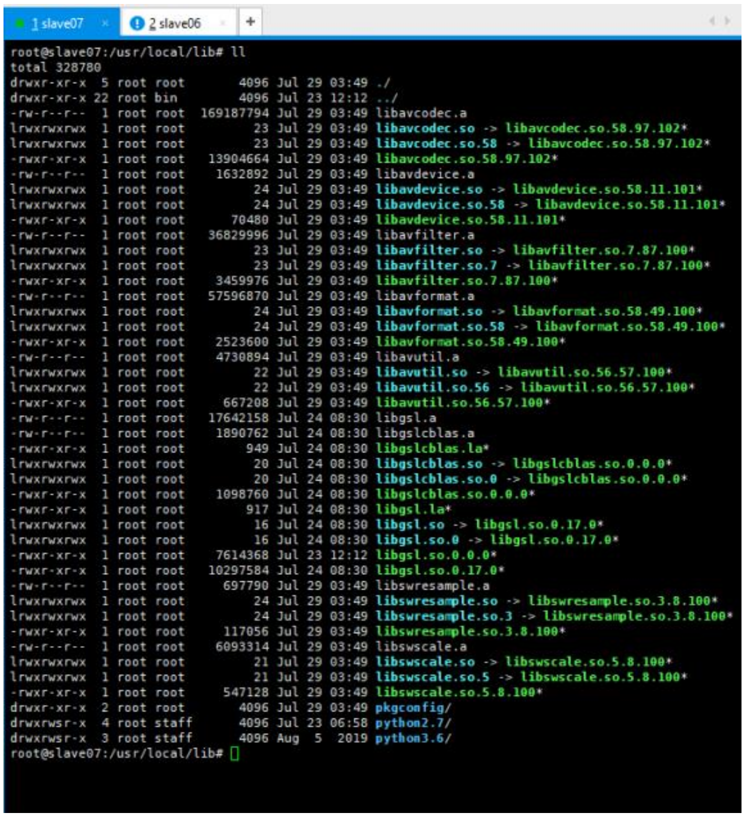
/soft/opencv/opencv-3.4.7/build/lib

ln -s /soft/opencv/opencv-3.4.7/build/lib/libopencv_java347.so /usr/lib/



```
lrwxrwxrwx  1 root root      14 Aug  4  2016 libcr.so.0 -> libcr.so.0.5.5
-rw-r--r--  1 root root   39744 Aug  4  2016 libcr.so.0.5.5
lrwxrwxrwx  1 root root      21 Mar 25 10:33 libDeployPkg.so.0 -> libDeployPkg.so.0.0.0
-rw-r--r--  1 root root   31304 Mar 25 10:33 libDeployPkg.so.0.0.0
-rwxr-xr-x  1 root root 2385608 Jul 31 11:27 libdetection.so*
lrwxrwxrwx  1 root root      20 Mar 25 10:33 libguestlib.so.0 -> libguestlib.so.0.0.0
-rw-r--r--  1 root root   26752 Mar 25 10:33 libguestlib.so.0.0.0
lrwxrwxrwx  1 root root      16 Mar 25 10:33 libhgfs.so.0 -> libhgfs.so.0.0.0
-rw-r--r--  1 root root  167248 Mar 25 10:33 libhgfs.so.0.0.0
-rwxr-xr-x  1 root root   23656 Jul 31 11:27 libjdetection.so*
-rwxr-xr-x  1 root root  700824 Jan 16  2020 libMMforHDFS.so*
lrwxrwxrwx  1 root root      12 Jun  8 06:10 libmpichcxx.so.0 -> libmpicxx.so
lrwxrwxrwx  1 root root      12 Jun  8 06:10 libmpichfort.so.0 -> libmpif90.so
lrwxrwxrwx  1 root root       9 Jun  8 06:10 libmpich.so.0 -> libmpi.so
lrwxrwxrwx  1 root root      30 Jun  8 06:10 libmpicxx.so -> /etc/alternatives/libmpicxx.so
lrwxrwxrwx  1 root root      30 Jun  8 06:10 libmpif77.so -> /etc/alternatives/libmpif77.so
lrwxrwxrwx  1 root root      30 Jun  8 06:10 libmpif90.so -> /etc/alternatives/libmpif90.so
lrwxrwxrwx  1 root root      31 Jun  8 06:10 libmpifort.so -> /etc/alternatives/libmpifort.so
lrwxrwxrwx  1 root root      27 Jun  8 06:10 libmpi.so -> /etc/alternatives/libmpi.so
lrwxrwxrwx  1 root root      29 Jun  8 06:10 libmpi++.so -> /etc/alternatives/libmpi++.so
-rwxr-xr-x  1 root root    9064 Dec 23  2019 libmydetetion.so*
lrwxrwxrwx  1 root root      56 Jul 31 09:31 libopencv_java347.so -> /soft/opencv/opencv-3.4.7/build/lib/libopencv
_java347.so*
lrwxrwxrwx  1 root root      18 Mar 25 10:33 libvgauth.so.0 -> libvgauth.so.0.0.0
-rw-r--r--  1 root root   85144 Mar 25 10:33 libvgauth.so.0.0.0
lrwxrwxrwx  1 root root      19 Mar 25 10:33 libvmtools.so.0 -> libvmtools.so.0.0.0
-rw-r--r--  1 root root  631480 Mar 25 10:33 libvmtools.so.0.0.0
drwxr-xr-x  3 root root    4096 Oct 31  2019 linux/
drwxr-xr-x  3 root root    4096 Oct 31  2019 linux-boot-probes/
drwxr-xr-x  3 root root    4096 Jul  8 06:37 locale/
drwxr-xr-x  2 root root    4096 Aug  2 07:39 lxcfs/
```

1.5 检查/usr/local/lib,至少这样才对。



```
root@slave07:/usr/local/lib# ll
total 328780
drwxr-xr-x  5 root root       4096 Jul 29 03:49 ./
drwxr-xr-x 22 root bin        4096 Jul 23 12:12 ../
-rw-r--r--  1 root root  169187794 Jul 29 03:49 libavcodec.a
lrwxrwxrwx  1 root root         23 Jul 29 03:49 libavcodec.so -> libavcodec.so.58.97.102*
lrwxrwxrwx  1 root root         23 Jul 29 03:49 libavcodec.so.58 -> libavcodec.so.58.97.102*
-rwxr-xr-x  1 root root   13904664 Jul 29 03:49 libavcodec.so.58.97.102*
-rw-r--r--  1 root root    1632892 Jul 29 03:49 libavdevice.a
lrwxrwxrwx  1 root root         24 Jul 29 03:49 libavdevice.so -> libavdevice.so.58.11.101*
lrwxrwxrwx  1 root root         24 Jul 29 03:49 libavdevice.so.58 -> libavdevice.so.58.11.101*
-rwxr-xr-x  1 root root      70480 Jul 29 03:49 libavdevice.so.58.11.101*
-rw-r--r--  1 root root   36829996 Jul 29 03:49 libavfilter.a
lrwxrwxrwx  1 root root         23 Jul 29 03:49 libavfilter.so -> libavfilter.so.7.87.100*
lrwxrwxrwx  1 root root         23 Jul 29 03:49 libavfilter.so.7 -> libavfilter.so.7.87.100*
-rwxr-xr-x  1 root root    3459976 Jul 29 03:49 libavfilter.so.7.87.100*
-rw-r--r--  1 root root   57596870 Jul 29 03:49 libavformat.a
lrwxrwxrwx  1 root root         24 Jul 29 03:49 libavformat.so -> libavformat.so.58.49.100*
lrwxrwxrwx  1 root root         24 Jul 29 03:49 libavformat.so.58 -> libavformat.so.58.49.100*
-rwxr-xr-x  1 root root    2523600 Jul 29 03:49 libavformat.so.58.49.100*
-rw-r--r--  1 root root    4730894 Jul 29 03:49 libavutil.a
lrwxrwxrwx  1 root root         22 Jul 29 03:49 libavutil.so -> libavutil.so.56.57.100*
lrwxrwxrwx  1 root root         22 Jul 29 03:49 libavutil.so.56 -> libavutil.so.56.57.100*
-rwxr-xr-x  1 root root     667208 Jul 29 03:49 libavutil.so.56.57.100*
-rw-r--r--  1 root root   17642158 Jul 24 08:30 libgsl.a
-rw-r--r--  1 root root    1890762 Jul 24 08:30 libgslcblas.a
-rwxr-xr-x  1 root root        949 Jul 24 08:30 libgslcblas.la*
lrwxrwxrwx  1 root root         20 Jul 24 08:30 libgslcblas.so -> libgslcblas.so.0.0.0*
lrwxrwxrwx  1 root root         20 Jul 24 08:30 libgslcblas.so.0 -> libgslcblas.so.0.0.0*
-rwxr-xr-x  1 root root    1098760 Jul 24 08:30 libgslcblas.so.0.0.0*
-rwxr-xr-x  1 root root        917 Jul 24 08:30 libgsl.la*
lrwxrwxrwx  1 root root         16 Jul 24 08:30 libgsl.so -> libgsl.so.0.17.0*
lrwxrwxrwx  1 root root         16 Jul 24 08:30 libgsl.so.0 -> libgsl.so.0.17.0*
-rwxr-xr-x  1 root root    7614368 Jul 23 12:12 libgsl.so.0.0.0*
-rwxr-xr-x  1 root root   10297584 Jul 24 08:30 libgsl.so.0.17.0*
-rw-r--r--  1 root root     697790 Jul 29 03:49 libswresample.a
lrwxrwxrwx  1 root root         24 Jul 29 03:49 libswresample.so -> libswresample.so.3.8.100*
lrwxrwxrwx  1 root root         24 Jul 29 03:49 libswresample.so.3 -> libswresample.so.3.8.100*
-rwxr-xr-x  1 root root     117056 Jul 29 03:49 libswresample.so.3.8.100*
-rw-r--r--  1 root root    6093314 Jul 29 03:49 libswscale.a
lrwxrwxrwx  1 root root         21 Jul 29 03:49 libswscale.so -> libswscale.so.5.8.100*
lrwxrwxrwx  1 root root         21 Jul 29 03:49 libswscale.so.5 -> libswscale.so.5.8.100*
-rwxr-xr-x  1 root root     547128 Jul 29 03:49 libswscale.so.5.8.100*
drwxr-xr-x  2 root root       4096 Jul 29 03:49 pkgconfig/
drwxrwsr-x  4 root staff      4096 Jul 23 06:58 python2.7/
drwxrwsr-x  3 root staff      4096 Aug  5  2019 python3.6/
root@slave07:/usr/local/lib# 
```

2 移动数据集合到别的机器的对应位置
/root/Detector
/root/data
/root/.ivy2
/soft/gsl

3 安装grpc这个不用管，这只是个jar包

4 移动maven,把maven store下的jar包移动
先查看maven版本
mvn -v
一定得是Ubuntu的源。
把7上的maven 仓库传送过去

5 检查spark版本：

得是2.4.3

应用二集群搭建

2021年2月3日 10:45

1 更新pip

sudo pip3 install --upgrade pip

2 执行

pip3 install cycler==0.10.0 future==0.18.2 kiwisolver==1.2.0 matplotlib==3.2.2 numpy==1.19.0 opencv-python==4.2.0.34
Pillow==7.2.0 pyparsing==2.4.7 python-dateutil==2.8.1 six==1.15.0 tqdm==4.47.0 albumentations flask

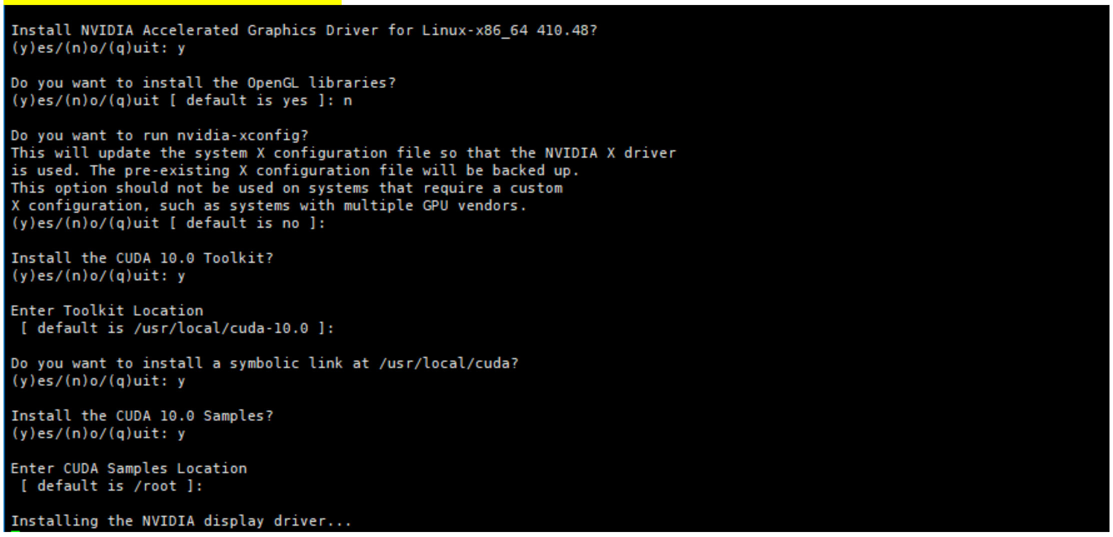
执行

pip3 install torch==1.7.0+cu101 torchvision==0.8.1+cu101 torchaudio==0.7.0 -f
https://download.pytorch.org/whl/torch_stable.html

应用三集群搭建

2020年12月10日　　11:48

0 安装cuda 10.0，选项如下：



```
Install NVIDIA Accelerated Graphics Driver for Linux-x86_64 410.48?
(y)es/(n)o/(q)uit: y

Do you want to install the OpenGL libraries?
(y)es/(n)o/(q)uit [ default is yes ]: n

Do you want to run nvidia-xconfig?
This will update the system X configuration file so that the NVIDIA X driver
is used. The pre-existing X configuration file will be backed up.
This option should not be used on systems that require a custom
X configuration, such as systems with multiple GPU vendors.
(y)es/(n)o/(q)uit [ default is no ]:

Install the CUDA 10.0 Toolkit?
(y)es/(n)o/(q)uit: y

Enter Toolkit Location
 [ default is /usr/local/cuda-10.0 ]:

Do you want to install a symbolic link at /usr/local/cuda?
(y)es/(n)o/(q)uit: y

Install the CUDA 10.0 Samples?
(y)es/(n)o/(q)uit: y

Enter CUDA Samples Location
 [ default is /root ]:

Installing the NVIDIA display driver...
```

安装cudnn ,文件在/usr/local下

tar zxvf cudnn-10.0-linux-x64-v7.6.4.38.tgz
sudo cp cuda/include/cudnn.h /usr/local/cuda/include/
sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64/
sudo chmod a+r /usr/local/cuda/include/cudnn.h
sudo chmod a+r /usr/local/cuda/lib64/libcudnn*

1安装软件bazel 0.26.1

github搜索该软件并找到对应版本
https://github.com/bazelbuild/bazel/releases/tag/0.26.1
安装过程:
 执行脚本 bazel-0.26.1-installer-linux-x86_64.sh

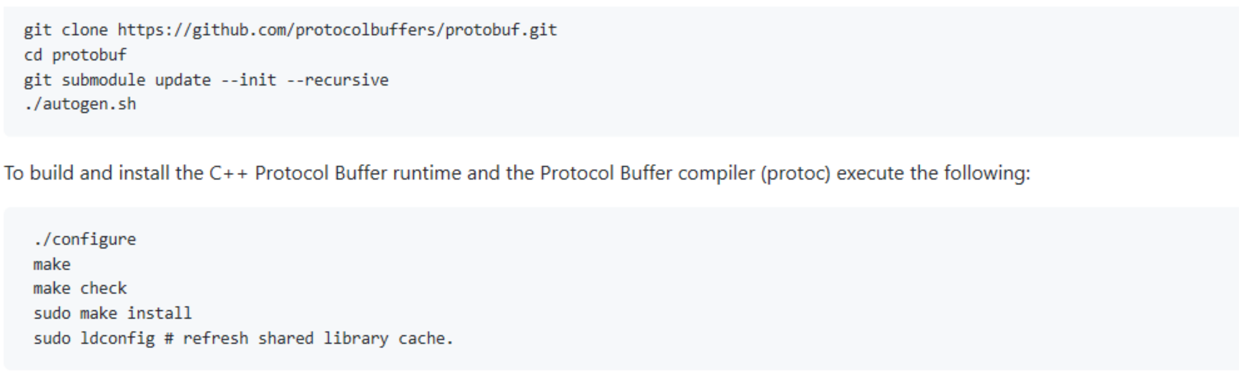
根据提示 source /root/.bazel/bazel-complete.bash

把/usr/local/lib/bazel/bin/放入PATH目录(这一步可以不做，只要找得到bazel就行)

bazel clean --expunge //去除外部依赖，这个单词间好像有乱码，需要重新输入一下。

2 安装protobuf3.8

下载该版本：
 https://github.com/protocolbuffers/protobuf/releases/tag/v3.8.0
安装文档：
https://github.com/protocolbuffers/protobuf/blob/master/src/README.md
执行如下（执行下面那个就行）：



```
git clone https://github.com/protocolbuffers/protobuf.git
cd protobuf
git submodule update --init --recursive
./autogen.sh
```

To build and install the C++ Protocol Buffer runtime and the Protocol Buffer compiler (protoc) execute the following:

```
./configure
make
make check
sudo make install
sudo ldconfig # refresh shared library cache.
```

3 确定python版本为2.7.17
安装anaconda(我装的是2.3.0)//装这个会影响原系统python的使用，所以我们暂时不要装这个了。

执行脚本Anaconda-2.3.0-Linux-x86_64.sh

4开始安装tensorflow V1.15.2(过程很痛苦)

执行：

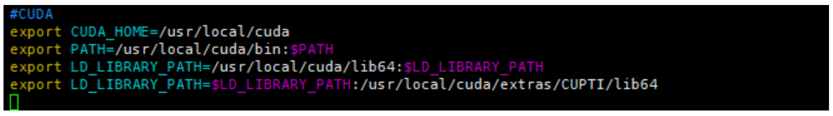
./configure

选：cuda=Y

选：[7.5,7.5]=[10.1,7.5]//不执行这个了

执行前确定CUDA相关环境变量，否则tensorflow配置cuda会有问题：

```
export CUDA_HOME=/usr/local/cuda
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/extras/CUPTI/lib64
```



```
#CUDA
export CUDA_HOME=/usr/local/cuda
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/extras/CUPTI/lib64
```

pip需要适配python2系列：/usr/bin/python2 -m pip install --upgrade pip

如果这条命令没用，pip install --target=路径 numpy这样下载这四个包。

```
pip install numpy
pip install future
pip install mock
pip install keras_preprocessing
```

bazel build --config=opt --config=cuda //tensorflow/tools/pip_package:build_pip_package --local_ram_resources=2048 --cxxopt="-D_GLIBCXX_USE_CXX11_ABI=0"

拷贝别的机器的/soft/tensorflow-1.15.2/bazel-bin/tensorflow/__init__.py或者自己机器的/soft/tensorflow-1.15.2/tensorflow/tools/__init__.py到/soft/tensorflow-1.15.2/bazel-bin/tensorflow/下（这一步后来好像也不用做了）

执行:bazel build --config=opt --config=cuda //tensorflow:libtensorflow_cc.so

这个看网上如果没有更好的办法，就只能安装代理了：export http_proxy=10.70.170.227:1080

执行：**./bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorflow_pkg**

执行：

```
pip install /tmp/tensorflow_pkg/tensorflow-1.15.2-cp27-cp27mu-linux_x86_64.whl
 mkdir /usr/local/include/tf
  cd /soft/tensorflow-1.15.2/
  cp -r tensorflow /usr/local/include/tf/
cp -r third_party /usr/local/include/tf/
 cp -r bazel-genfiles/* /usr/local/include/tf/
 cp bazel-bin/tensorflow/libtensorflow_cc.so /usr/local/lib/
cp bazel-bin/tensorflow/libtensorflow_cc.so.1 /usr/local/lib/
  cp bazel-bin/tensorflow/libtensorflow_framework.so /usr/local/lib/
 cp bazel-bin/tensorflow/libtensorflow_framework.so.1 /usr/local/lib/
 ldconfig
```

遇到io_bazel_rules_docker错误，在WORKSPACE文件中添加：

```
http_archive(
    name = "io_bazel_rules_docker",
    sha256 = "aed1c249d4ec8f703edddf35cbe9dfaca0b5f5ea6e4cd9e83e99f3b0d1136c3d",
    strip_prefix = "rules_docker-0.7.0",
    urls = ["https://github.com/bazelbuild/rules_docker/archive/v0.7.0.tar.gz"],
)
```

遇到llvm问题，执行:

sudo pip install future

遇到pybind11 timeout找不到包,执行:

apt -y install python-pybind11 和：

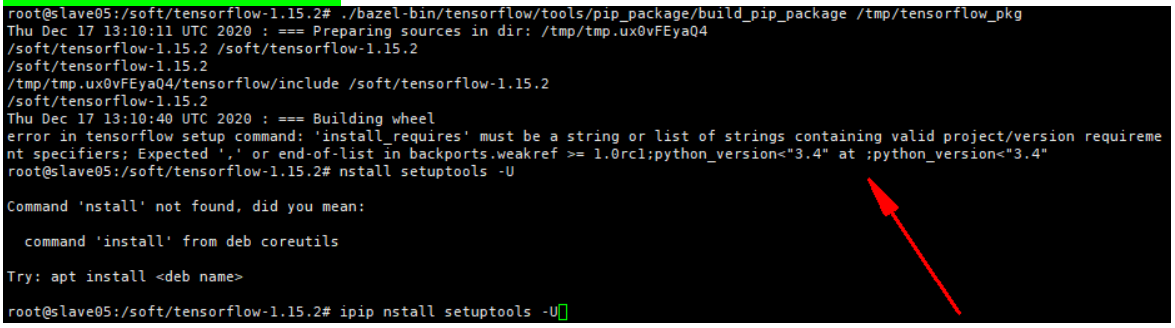
conda install pybind11（这一步可能无效）

执行bazel build --config=opt --config=cuda //tensorflow:libtensorflow_cc.so遇到的问题：

这个看网上如果没有更好的办法，就只能安装代理了：export http_proxy=10.70.170.227:1080

执行./bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorflow_pkg遇到的问题：

遇到问题string containing…：



```
root@slave05:/soft/tensorflow-1.15.2# ./bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorflow_pkg
Thu Dec 17 13:10:11 UTC 2020 : === Preparing sources in dir: /tmp/tmp.ux0vFEyaQ4
/soft/tensorflow-1.15.2 /soft/tensorflow-1.15.2
/soft/tensorflow-1.15.2
/tmp/tmp.ux0vFEyaQ4/tensorflow/include /soft/tensorflow-1.15.2
/soft/tensorflow-1.15.2
Thu Dec 17 13:10:40 UTC 2020 : === Building wheel
error in tensorflow setup command: 'install_requires' must be a string or list of strings containing valid project/version requireme
nt specifiers; Expected ',' or end-of-list in backports.weakref >= 1.0rc1;python_version<"3.4" at ;python_version<"3.4"
root@slave05:/soft/tensorflow-1.15.2# nstall setuptools -U

Command 'nstall' not found, did you mean:

  command 'install' from deb coreutils

Try: apt install <deb name>

root@slave05:/soft/tensorflow-1.15.2# ipip nstall setuptools -U
```

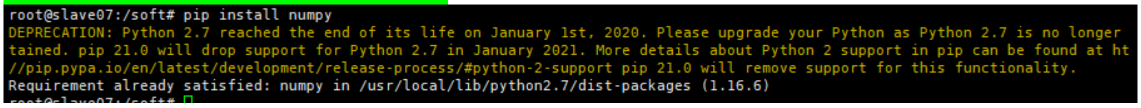
执行： pip install setuptools -U

遇到问题：

/tensorflow/python/keras/api:keras_python_api_gen failed

pip install -U --user keras_preprocessing --no-deps

遇到Requirement already satisfie问题：



```
root@slave07:/soft# pip install numpy
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer
tained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at ht
//pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functionality.
Requirement already satisfied: numpy in /usr/local/lib/python2.7/dist-packages (1.16.6)
```

/usr/bin/python3 -m pip install --upgrade pip

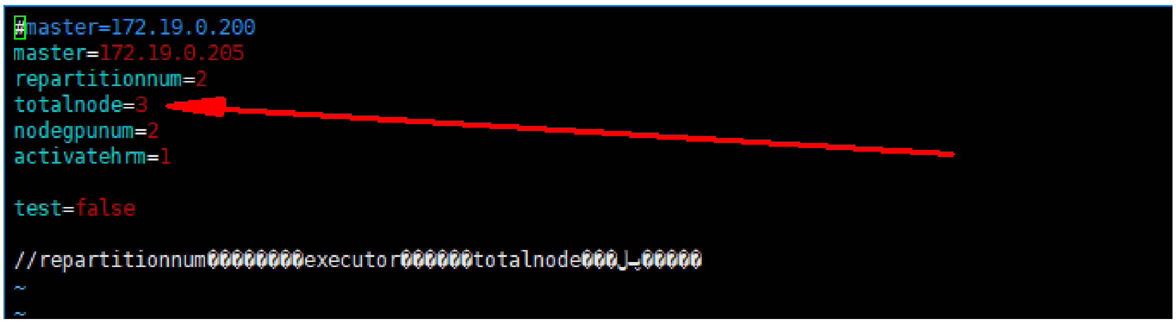
应用3运行问题

2021年1月4日　　10:13

1 启动grpc server

cd /data/006zzy/HUASHIDA/app123/app/src/main/resources

有几个节点配几个totalnode，然后重新编译jar包



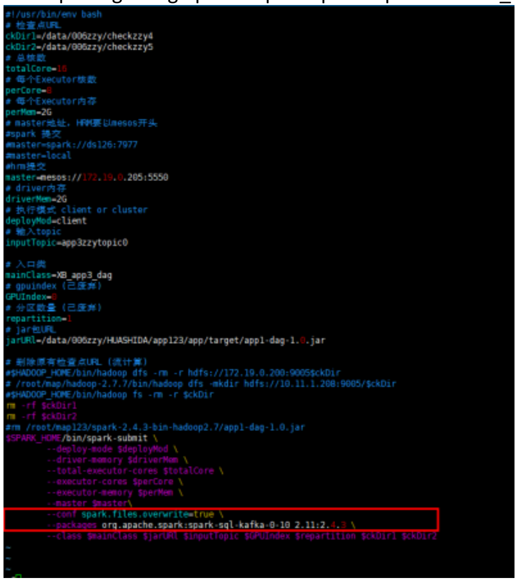
```
#master=172.19.0.200
master=172.19.0.205
repartitionnum=2
totalnode=3
nodegpunum=2
activatehrm=1

test=false

//repartitionnum��������executor������totalnode���J_�����
~
~
```

执行：java -cp ./app1-dag-1.0-jar-with-dependencies.jar runtime.grpc.CoScheGrpcServer

2 遇到找不到kafka data source 问题：

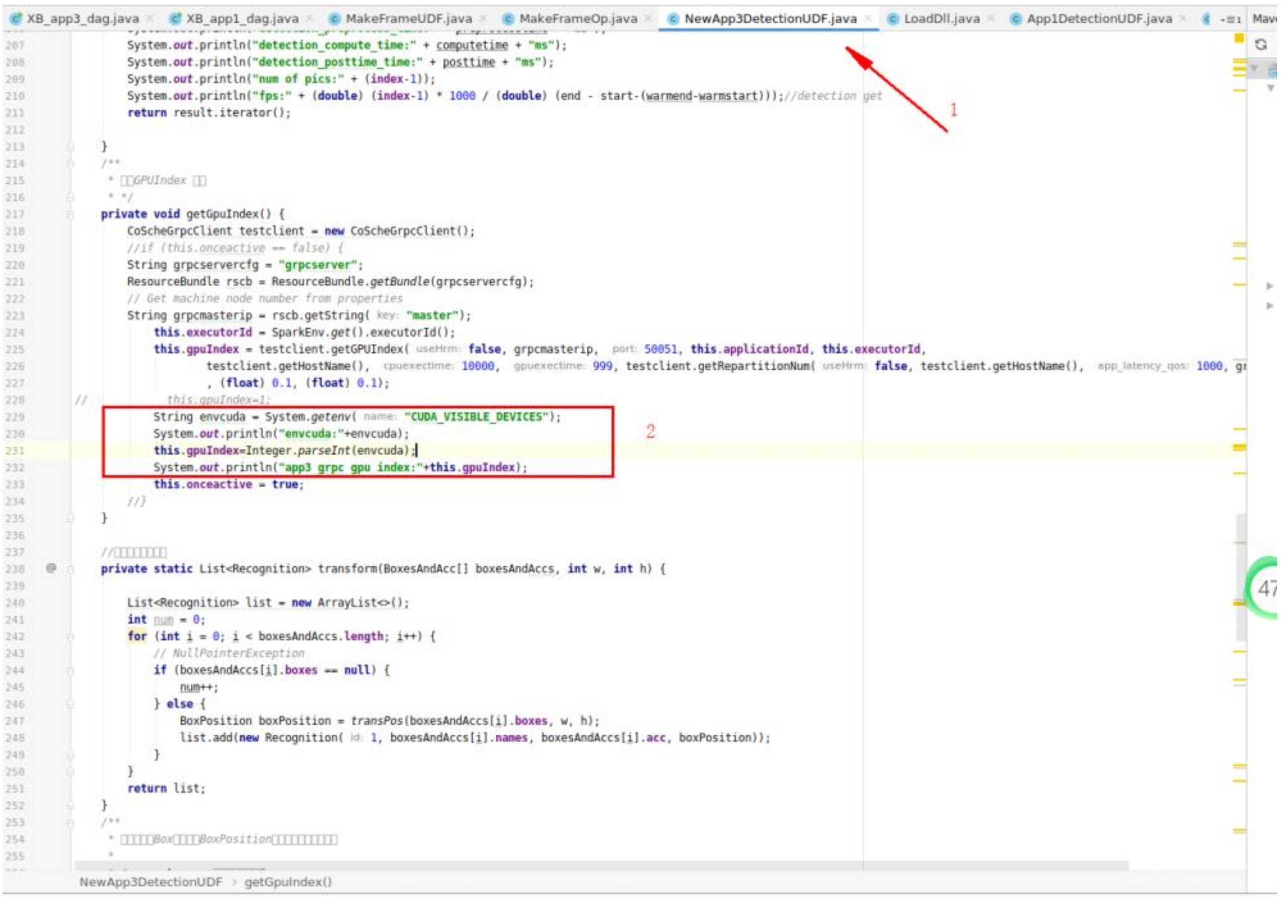
添加这两句话：

```
        --conf spark.files.overwrite=true \
        --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.3 \
```



```
#!/usr/bin/env bash
# 检查点URL
ckDir1=/data/006zzy/checkzzy4
ckDir2=/data/006zzy/checkzzy5
# 总核数
totalCore=16
# 每个Executor核数
perCore=8
# 每个Executor内存
perMem=2G
# master地址，HRM要以mesos开头
#spark 提交
#master=spark://ds126:7077
#master=local
#hrm提交
master=mesos://172.19.0.205:5550
# driver内存
driverMem=2G
# 执行模式 client or cluster
deployMod=client
# 输入topic
inputTopic=app3zzytopic0

# 入口类
mainClass=XB_app3_dag
# gpuindex (已废弃)
GPUIndex=0
# 分区数量 (已废弃)
repartition=1
# jar包URL
jarURl=/data/006zzy/HUASHIDA/app123/app/target/app1-dag-1.0.jar

# 删除原有检查点URL (流计算)
#$HADOOP_HOME/bin/hadoop dfs -rm -r hdfs://172.19.0.200:9005$ckDir
# /root/map/hadoop-2.7.7/bin/hadoop dfs -mkdir hdfs://10.11.1.208:9005/$ckDir
#$HADOOP_HOME/bin/hadoop fs -rm -r $ckDir
rm -rf $ckDir1
rm -rf $ckDir2
#rm /root/map123/spark-2.4.3-bin-hadoop2.7/app1-dag-1.0.jar
$SPARK_HOME/bin/spark-submit \
        --deploy-mode $deployMod \
        --driver-memory $driverMem \
        --total-executor-cores $totalCore \
        --executor-cores $perCore \
        --executor-memory $perMem \
        --master $master\
        --conf spark.files.overwrite=true \
        --packages org.apache.spark:spark-sql-kafka-0-10 2.11:2.4.3 \
        --class $mainClass $jarURl $inputTopic $GPUIndex $repartition $ckDir1 $ckDir2
~
~
~
```

3 tensorflow加载显存使用的GPUID是系统变量，所以我们需要在应用三对应部分添加如下内容并去除/root/.bashrc下CUDA_VISIBLE_DEVICES环境变量：



```java
        System.out.println("detection_compute_time:" + computetime + "ms");
        System.out.println("detection_posttime_time:" + posttime + "ms");
        System.out.println("num of pics:" + (index-1));
        System.out.println("fps:" + (double) (index-1) * 1000 / (double) (end - start-(warmend-warmstart)));//detection get
        return result.iterator();

    }
    /**
     * □□GPUIndex □□
     * */
    private void getGpuIndex() {
        CoScheGrpcClient testclient = new CoScheGrpcClient();
        //if (this.onceactive == false) {
        String grpcservercfg = "grpcserver";
        ResourceBundle rscb = ResourceBundle.getBundle(grpcservercfg);
        // Get machine node number from properties
        String grpcmasterip = rscb.getString( key: "master");
            this.executorId = SparkEnv.get().executorId();
            this.gpuIndex = testclient.getGPUIndex( useHrm: false, grpcmasterip,  port: 50051, this.applicationId, this.executorId,
                    testclient.getHostName(),  cpuexectime: 10000,  gpuexectime: 999, testclient.getRepartitionNum( useHrm: false, testclient.getHostName(),  app_latency_qos: 1000, gr
                    , (float) 0.1, (float) 0.1);
//              this.gpuIndex=1;
            String envcuda = System.getenv( name: "CUDA_VISIBLE_DEVICES");
            System.out.println("envcuda:"+envcuda);
            this.gpuIndex=Integer.parseInt(envcuda);
            System.out.println("app3 grpc gpu index:"+this.gpuIndex);
            this.onceactive = true;
        //}
    }

    //□□□□□□□□
    private static List<Recognition> transform(BoxesAndAcc[] boxesAndAccs, int w, int h) {

        List<Recognition> list = new ArrayList<>();
        int num = 0;
        for (int i = 0; i < boxesAndAccs.length; i++) {
            // NullPointerException
            if (boxesAndAccs[i].boxes == null) {
                num++;
            } else {
                BoxPosition boxPosition = transPos(boxesAndAccs[i].boxes, w, h);
                list.add(new Recognition( id: 1, boxesAndAccs[i].names, boxesAndAccs[i].acc, boxPosition));
            }
        }
        return list;
    }
    /**
     * □□□□Box□□□□BoxPosition□□□□□□□□□
     *
```