**VEP Platform 利用Kafka传输文件测试报告**

一、Kafka性能调研情况

经过近期调研得知，Kafka设计的初衷为迅速处理短小消息，一般在10KB左右大小的消息的吞吐量性能最好。若我们想传输视频中的某一帧下的图片，其大小基本在1MB左右。这时候，我们可以进行如下处理：

1. 通过HDFS进行共享存储，并通过Kafka传输文件的位置信息等。
2. 可将大的数据切片成多个10KB大小的信息片段，使用分区主键确保一个大消息的所有部分会被发送到同一个kafka分区，而消费端使用时再将消息片段重新进行还原。

二、测试过程及结果

1、 测试过程

我主要对第二种方法进行了测试，使用的测试素材是一张1.6MB大小的jpg图片。采用的传输方式是单producer、单consumer通过kafka进行传输。（多producer或多consumer的情况后文中会做讨论）

这种消息切片的传输过程主要有：

1. producer端文件转译成byte数组（以下称为消息），切片成每个10KB大小的byte数组片段（以下成为片段），其中包括前20位的信息标志位。

共四位，代表消息的编号。

共四位，代表当前片段的编号。

共四位，代表原消息切割成几段。

共四位，用于判断是否为有效片段。

共四位，代表原消息byte数组长度。

共10240位，10KB。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 消息编号 | 片段编号 | 片段总数 | 片段有效位 | 消息总长度 | 片段有效数据段 | …… |

1. consumer端接收消息的片段，并重新还原消息。主要通过创建二维可变大小数组的形式进行重组。通过标志位，判断片段的归属，并判断消息是否还原完成，一旦完成，立即进行处理。

2、 测试结果

传输图片为一张1.6MB大小的jpg图片（项目中所涉及的图片大小为1.2MB左右），测试结果如下：

1. 传输10次：

|  |  |
| --- | --- |
| 单纯produce发送（单位：毫微秒） | 单纯consumer接收、处理（单位：毫微秒） |
| 727505787 | 138064829 |

1. 传输100次：

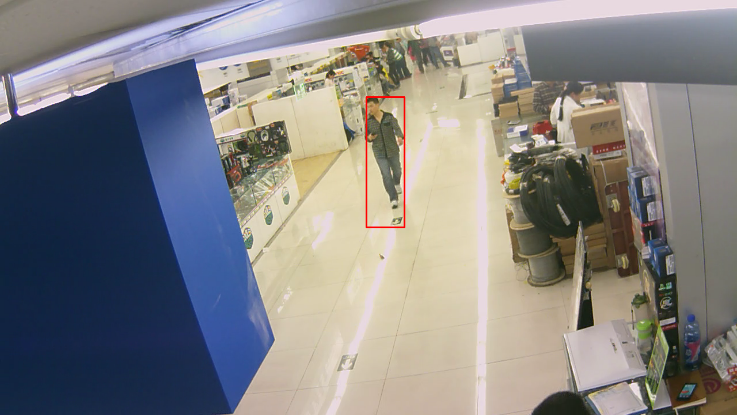
|  |  |
| --- | --- |
| 单纯produce发送（单位：毫微秒） | 单纯consumer接收、处理（单位：毫微秒） |
| 3645802118 | 1804442590 |

1. 传输1000次：

|  |  |
| --- | --- |
| 单纯produce发送（单位：毫微秒） | 单纯consumer接收、处理（单位：毫微秒） |
| 48186098615 | 18812036871 |

这里列出的数据，是producer、consumer单独运行时得到的数据均值。由数据可见，consumer端的速度快于producer的速度，当两段并行运行时，consumer段所用的时间基本等于producer段单独运行所需时间。可见当两段并行运行时，发送接受并处理10张图片的时间约为0.7s，100张约为3.6秒，1000张约为48秒左右。

当我们只需要传输原图中bounding box的截取图片时，得到的图片大小约为几十KB:





截取图片26KB

原图1.02MB

若是这种情况，可以不用对截图其进行切割，其大小完全在kafka的消息默认值1MB以内。可以直接通过kafka传送，避免不必要的误差。

三、遗留问题

1、 修改消息片段的标志位，使其可以满足多producer并行发送的要求。现有的方案有两种：引入发送消息第一条片段的时间到标志位中，引入消息第一条片段的发送信息到标识为中。

2、所传文件的大小上限可通过程序内的参数调整。

3、HDFS传输测试仍在继续中。

4、HDFS的分块大小（默认为64MB）可以修改，不过效率还需根据平台中数据的特性进行测试。