链滴

# openvidu-server 搭建 kurento 负载均衡机制

作者：liuliang133

原文链接：https://ld246.com/article/1596024759097

来源网站：链滴

**为什么要扩展流媒体服务器（kurento）**

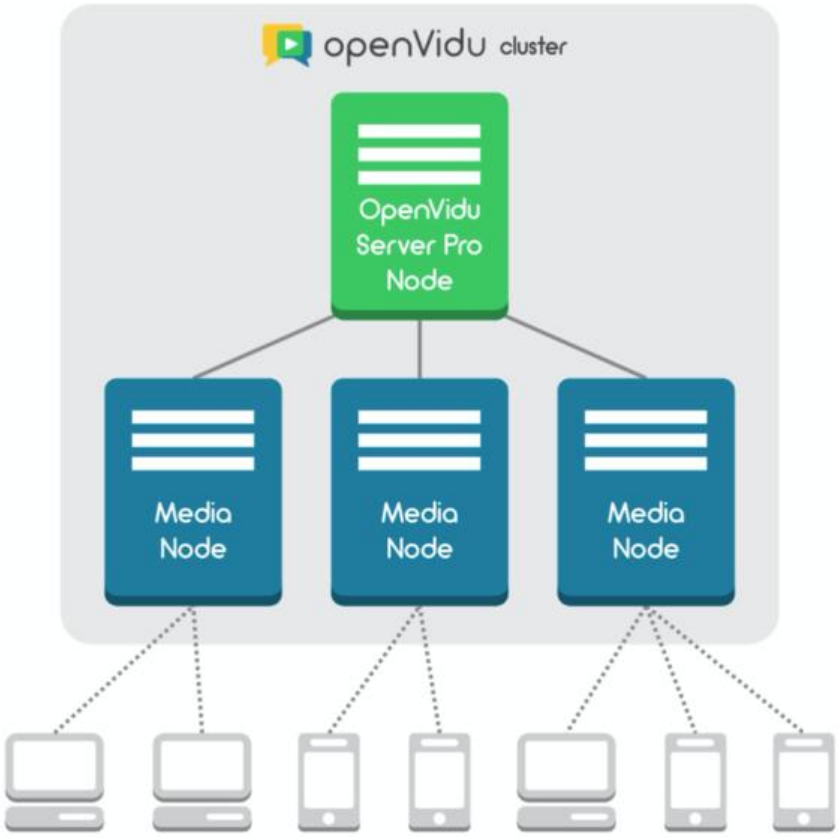
我们从openvidu官网上拉下代码,启动后，默认是一个openvidu服务器对一个kurento服务器。分别处理信令和流媒体。根据官方文档所指的，一个session,参与者有7个的会话条件下，不同配置的服务能够承受的压力如下图：

| Instance type [cores, memory] | c5.large [2 cores, 4 GB] | c5.xlarge [4 cores, 8 GB] | c5.2xlarge [8 cores, 16 GB] | [16 |
|---|---|---|---|---|
| Number of Sessions | 4 | 7 | 15 | 26 |
| Number of Publishers (= browsers) | 28 | 49 | 105 | 182 |
| Number of Subscribers | 168 | 294 | 630 | 1092 |

是的，当我们使用4c8g的服务器的时候，理论上只能处理7个session,也就是只有7个房间同时存在！在生产上是根本不能够承受的。因为信令服务器压力本身不大，所以我们必须扩展媒体服务器（kurento），让其承受能力增加。

**怎么扩展kurento**

openvidu 分为CE版本和Pro版本，我们使用的是CE版本，也就是源代码里面不会将kurento的代码我们，那我们该怎么扩展源代码，让其可以负载多个kurento呢？首先可以看一下openviduPro-kureto的架构




是的，如上图，一个openvidu服务器扩展了3台kurento服务器，由于瓶颈在kurento上，不考虑openviduserver的话，理论上我们的可承受能力会乘以3！接下来我们分析一下openvidu的源码。

```
KMS_URIS=["ws://116.196.10.***:8888/kurento"]
```

```
io.openvidu.server.config.OpenviduConfig:
```

```
...
 public List<String> checkKmsUris() {

       String property = "KMS_URIS";

       return asKmsUris(property, getValue(property));

  }
...
```

从这里可以看到，代码的入口在这里。但是如果我们在KMS_URIS里面添加多个kurento地址，他还只会取地址里面的第一条kurento地址，也就是没有处理负载操作。原因在下面：

```
public class FixedOneKmsManager extends KmsManager {

   @Override
   public List<Kms> initializeKurentoClients(List<KmsProperties> kmsProperties, boolean dis
onnectUponFailure) throws Exception {
       KmsProperties firstProps = kmsProperties.get(0);
       KurentoClient kClient = null;
       Kms kms = new Kms(firstProps, loadManager);
```

```
        try {
            kClient = KurentoClient.create(firstProps.getUri(), this.generateKurentoConnectionList
ner(kms.getId()));
            this.addKms(kms);
            kms.setKurentoClient(kClient);

            // TODO: This should be done in KurentoClient connected event
            kms.setKurentoClientConnected(true);
            kms.setTimeOfKurentoClientConnection(System.currentTimeMillis());

        } catch (KurentoException e) {
            log.error("KMS in {} is not reachable by OpenVidu Server", firstProps.getUri());
            if (kClient != null) {
                kClient.destroy();
            }
            throw new Exception();
        }
        return Arrays.asList(kms);
    }

    @Override
    @PostConstruct
    protected void postConstructInitKurentoClients() {
        try {
            List<KmsProperties> kmsProps = new ArrayList<>();
            for (String kmsUri : this.openviduConfig.getKmsUris()) {
                String kmsId = KmsManager.generateKmsId();
                kmsProps.add(new KmsProperties(kmsId, kmsUri));
            }
            this.initializeKurentoClients(kmsProps, true);
        } catch (Exception e) {
            // Some KMS wasn't reachable
            log.error("Shutting down OpenVidu Server");
            System.exit(1);
        }
    }

}
```

initializeKurentoClients 方法里面清楚的写了，只会取KMS_URIS的第0条数据。那如果我们想处理条，这里就需要返回一个集合，很简单，我们定义一个自己的KmsManager类,重写initializeKurentoClients方法。如下:

```
public class FixedJDKmsManager extends KmsManager {

    @Override
    public List<Kms> initializeKurentoClients(List<KmsProperties> kmsProperties, boolean dis
onnectUponFailure) throws Exception {

        ArrayList<Kms> results = new ArrayList<>();

        for(KmsProperties kmsp:kmsProperties){
            KurentoClient kClient = null;
            Kms kms = new Kms(kmsp, loadManager);
```

```
            try {
                kClient = KurentoClient.create(kmsp.getUri(), this.generateKurentoConnectionListen
r(kms.getId()));
                this.addKms(kms);
                kms.setKurentoClient(kClient);

                // TODO: This should be done in KurentoClient connected event
                kms.setKurentoClientConnected(true);
                kms.setTimeOfKurentoClientConnection(System.currentTimeMillis());

                results.add(kms);
            } catch (KurentoException e) {
                log.error("KMS in {} is not reachable by OpenVidu Server", kmsp.getUri());
                if (kClient != null) {
                    kClient.destroy();
                }
                throw new Exception();
            }
        }

        return results;
    }

    @Override
    @PostConstruct
    protected void postConstructInitKurentoClients() {
        try {
            List<KmsProperties> kmsProps = new ArrayList<>();
            for (String kmsUri : this.openviduConfig.getKmsUris()) {
                String kmsId = KmsManager.generateKmsId();
                kmsProps.add(new KmsProperties(kmsId, kmsUri));
            }
            this.initializeKurentoClients(kmsProps, true);
        } catch (Exception e) {
            // Some KMS wasn't reachable
            log.error("Shutting down OpenVidu Server");
            System.exit(1);
        }
    }

}
```

配置使用自己定义的kmsManager

**怎么实现加权轮询负载均衡**

和kms_urls同理，我们可以在配置文件添加kms_weights:

```
KMS_WEIGHT=[1]
```

并且，人为的控制kms_urls的size与kms_weight同等

然后在openviduConfig里面，将其与kms_urls类似的道理初始化。核心代码如下：

```
public class OpenviduConfig{

    ...

    private List<Integer> kmsWeights;


    public List<Integer> getKmsWeights() {
        return kmsWeights;
    }

    public void setKmsWeights(List<Integer> weights) {
        this.kmsWeights = weights;
        log.info("=====kms权重被重置为:{}", this.kmsUrisList);
    }

    public List<Integer> initWeight() {

        String property = "KMS_WEIGHT";

        return asKmsWeights(property, getValue(property));
    }


    protected void checkConfigurationProperties(boolean loadDotenv) {

      ...
      kmsUrisList = checkKmsUris();

      kmsWeights = initWeight();
      ...

    }

  ...

}
```

到此，我们将自己想要的加权参数放入容器了，但是openvidu还没有选择kurento，接下来需要找到penvidu寻找对应kurento的地方，根据权值去取对应的url：

熟读源码后可以发现，在KurentoSessionManager.joinRoom方法里面，调用getLessLoadedConnetedAndRunningKms方法，这个方法就是选择kms的方法。同样的道理，我们可以重写一个方法去照自己的逻辑选择kms我写的如下：

```
public synchronized Kms getLoadBalanceConnectedAndRunningKms() throws NoSuchElemenException {
        List<KmsLoad> kmsLoads = getKmsLoads().stream().filter(kmsLoad -> kmsLoad.kms.isKuentoClientConnected()
                && mediaNodeStatusManager.isRunning(kmsLoad.kms.getId()) ).collect(Collectors.oList());

        if (kmsLoads.isEmpty()) {
```

```
            throw new NoSuchElementException();
        } else {
            //todo 这里编写kms的负载均衡
            this.openviduConfig.getKmsWeights();
//            Kms kms = kmsLoads.get(Integer.parseInt(kmsNode)).kms;

            //第一次:初始化数据
            KmsWeightRobin.initKmsLoads(kmsLoads,this.openviduConfig.getKmsWeights());
            KmsLoad kmsWeightRobin = KmsWeightRobin.getKmsWeightRobin();
            log.info("==========>>>>>>加权轮询后,选择kms:{}",kmsWeightRobin.kms.getUri());
            return  kmsWeightRobin.getKms();

        }
    }


public class KmsWeightRobin {

//    static Map<String,Integer> ipMap=new HashMap<>();

    static Map<KmsManager.KmsLoad,Integer> kmsMap=new HashMap<>();

    static List<KmsManager.KmsLoad> kmsLoads = null;

    static List<Integer> weights = null;

    public  static synchronized void initMap() {

        kmsMap.clear();

        for(int i = 0;i<kmsLoads.size();i++){
            kmsMap.put(kmsLoads.get(i),weights.get(i));
        }
    }



    static Integer pos=0;
    public static KmsManager.KmsLoad getKmsWeightRobin(){
        Map<KmsManager.KmsLoad,Integer> ipServerMap=new ConcurrentHashMap<>();
        ipServerMap.putAll(kmsMap);

        Set<KmsManager.KmsLoad> ipSet=ipServerMap.keySet();
        Iterator<KmsManager.KmsLoad> ipIterator=ipSet.iterator();

        //定义一个list放所有server
        ArrayList<KmsManager.KmsLoad> ipArrayList=new ArrayList<KmsManager.KmsLoad>();

        //循环set，根据set中的可以去得知map中的value，给list中添加对应数字的server数量
        while (ipIterator.hasNext()){
            KmsManager.KmsLoad serverName=ipIterator.next();
            Integer weight=ipServerMap.get(serverName);
            for (int i = 0;i < weight ;i++){
                ipArrayList.add(serverName);
```

```
            }
        }
        KmsManager.KmsLoad serverName=null;
        if (pos>=ipArrayList.size()){
            pos=0;
        }
        serverName=ipArrayList.get(pos);
        //轮询+1
        pos ++;
        return  serverName;
    }


//    public static Boolean initMapFlag = false;

    public static void initKmsLoads(List<KmsManager.KmsLoad> kmsLoads,List<Integer> weig
ts){
        if(ObjectUtils.isEmpty(KmsWeightRobin.kmsLoads)){
            KmsWeightRobin.kmsLoads = kmsLoads;
        }
        if(ObjectUtils.isEmpty(KmsWeightRobin.weights)){
            KmsWeightRobin.weights = weights;
        }
//        if(!initMapFlag){
            initMap();
//            initMapFlag = true;
//        }


    }


    public static void updateKmsLoads(List<KmsManager.KmsLoad> kmsLoads,List<Integer>
eights){


        return;
    }


}
```

这样，在选择kms的时候，就可以让容器按照weight去选择了。

**待解决或者改进**

目前的负载方案，使用了一个openvidu多个kms.但是对于openvidu还是单点的，我曾将想过将open idu做高可用（多点）。但是目前openvidu的很多信息都是存储在jvm的内存里面（比如session等信 ）。当然也有内存共享等可以解决，但是目前还没有做openvidu的多点方案。