First steps for API

1. remove special job data from code of rabbitMQ sender
2. create json transmission for one job\_id to all nodes in the same manner, using API approach

**Related API data**

The user has at his disposal the following information:

1. Your own understanding of work

eg:

Transcode my stream or file into 3 additional streams with the appropriate parameters

2. The user formalizes this task by choosing options in a simple interface or through an interactive script.

The user must set (select as an option):

original stream parameters

stream\_0

resolution\_name: \_4k

resolution: 3840x2160

bitrate: 50,000,000

The number of transcoded streams

number\_of\_streams: 3

transcoding parameters for each stream (resolution and bitrate for now)

stream\_1

resolution\_name: \_1080p

resolution: 1920х1080

bitrate: 6000000

stream\_2

resolution\_name: \_720p

resolution: 1280х720

bitrate: 3000000

stream\_3

resolution\_name: 480p

resolution: 854х480

bitrate: 1000000

after exchanging with the authorization and billing system, the user receives job\_id for this job

work parameters are saved to a file

job\_id.txt

stream\_0

resolution\_name: \_4k

resolution: 3840x2160

bitrate:50000000

*number\_of\_streams: 3*

*stream\_1*

*resolution\_name: \_1080p*

*resolution: 1920х1080*

*bitrate: 6000000*

*stream\_2*

*resolution\_name: \_720p*

*resolution: 1280х720*

*bitrate: 3000000*

*stream\_3*

*resolution\_name: 480p*

*resolution: 854х480*

*bitrate: 1000000*

job\_id.txt received as a result of authorization and payment, is located on the disk of the client’s machine

3. The system processes the job\_id parameters and creates the corresponding pipeline

The pipeline.txt (json) is on the client’s disk

obtained from a preliminary survey of rabbitMQ

pipeline.txt

*distributor: ip*

*failover distributor: ip*

*transcoder 1: ip*

*transcoder 2: ip*

*transcoder 3: ip*

*failover transcoder: ip*

*storage: ip*

*failover storage: ip*

Thus, to send control messages, the client has 2 files,

which are combined into one

job\_id.txt и pipeline.txt ---> job\_id.conf

*stream\_0*

*resolution\_name: \_4k*

*resolution: 3840x2160*

*bitrate:50000000*

*number\_of\_streams: 3*

*stream\_1*

*resolution\_name: \_1080p*

*resolution: 1920х1080*

*bitrate: 6000000*

*stream\_2*

*resolution\_name: \_720p*

*resolution: 1280х720*

*bitrate: 3000000*

*stream\_3*

*resolution\_name: 480p*

*resolution: 854х480*

*bitrate: 1000000*

*distributor: ip*

*failover\_distributor: ip*

*transcoder\_1: ip*

*transcoder\_2: ip*

*transcoder\_3: ip*

*failover\_transcoder: ip*

*storage: ip*

*failover\_storage: ip*

**API usage**

**generating a control file for a distributor:**

The distributor copies the stream to the stack and to the transcoders according to the pipeline.

application in\_job\_id {

live on;

exec /usr/bin/ffmpeg -i rtmp://localhost:1935//in\_job\_id/s\_job\_id

-c copy -f flv rtmp://storage:1935/a\_job\_id/s\_job\_id\_original

-c copy -f flv rtmp:/*transcoder\_1*:1935/in\_job\_id/s\_job\_id

-c copy -f flv rtmp:/*transcoder\_2*:1935/in\_job\_id/s\_job\_id

-c copy -f flv rtmp:/*transcoder\_3*:1935/in\_job\_id/s\_job\_id

)

**generation of control file for N transcoder:**

application in\_job\_id {

live on;

exec /usr/bin/ffmpeg -i rtmp://localhost:1935/in\_job\_id/s\_job\_id

-vcodec libx264 -b:v stream\_N.bitrate -vf scale=stream\_N.resolution -f flv rtmp://storage//a\_job\_id/s\_job\_id\_stream\_N.resolution\_name;

}

**control file generation for storage:**

application a\_job\_id {

live on;

hls on;

hls\_fragment 5s;

hls\_path /opt/data/a\_job\_id;

hls\_nested off;

hls\_fragment\_naming timestamp; #hls\_fragment\_naming sequential;

hls\_fragment\_naming\_granularity 2;

hls\_fragment\_slicing aligned;

hls\_cleanup off;

hls\_continuous on;

hls\_variant \_stream\_1.resolution\_name BANDWIDTH=stream\_1.bitrate,RESOLUTION=stream\_1.resolution;

hls\_variant \_stream\_2.resolution\_name BANDWIDTH=stream\_2.bitrate,RESOLUTION=stream\_2.resolution;

hls\_variant \_stream\_2.resolution\_name BANDWIDTH=stream\_3.bitrate,RESOLUTION=stream\_3.resolution;

hls\_variant \_stream\_0.resolution\_name BANDWIDTH=stream\_0.bitrate,RESOLUTION=stream\_0.resolution;;

}

**client start job (related API results)**

On the client side, the executable file config\_ job\_id.sh is generated and launched.

send: distr\_json ---> distributor

send:trans1\_json ---> transcoder\_1

send:trans2\_json ---> transcoder\_2

send:trans3\_json ---> transcoder\_3

send storage.jcon ---> storage

ffmpeg -i <input file> c copy f flv rtmp://distributor:1935/in\_job\_id/s\_job\_id