**RabbitMQ testing for videocoin network**

**test 1: 1 Client, manual process creation, 1 distributor, 4 transcoders, 1 storage**

**Service Network Level:**



Streams and transcoding parameters:

1. The original 4k rtmp stream is created from a file with parameters:

lpclip1.mp4':

Metadata:

major\_brand : qt

minor\_version : 537199360

compatible\_brands: qt

creation\_time : 2018-03-07 15:16:57

comment : equirectangular projection

Duration: 00:00:51.41, start: 0.000000, bitrate: 50556 kb/s

Stream #0:0(eng): Video: h264 (Main) (avc1 / 0x31637661), yuv420p, 4096x2160, 50430 kb/s, 29.57 fps, 30 tbr, 10k tbn, 20k tbc (default)

Metadata:

creation\_time : 2018-03-07 15:16:57

handler\_name : VideoHandler

Stream #0:1(eng): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, mono, fltp, 127 kb/s (default)

Metadata:

creation\_time : 2018-03-07 15:16:57

handler\_name : SoundHandler

2, Distributor (IP) replicates this stream to 4 transcoders. And sends the original hls stream to storage

ffmpeg config:

nginx config:

3. Transcoder 1 (IP) accepts a 4k rtmp stream and transcodes it into a 1080p rtmp stream with the following parameters:

ffmpeg config:

ffmpeg -re -i rtmp: // IP: 1935 / app\_name / stream\_name -vcodec libx264 -vf scale = 1920x1080 -b: v 6M -b: a 192k -f flv rtmp: // IP: 1935 / app\_name / stream\_name1

nginx config:

4. Transcoder 2 (IP) accepts a 4k rtmp stream and transcodes it into a 720p rtmp stream with the following parameters:

ffmpeg config:

ffmpeg -re -i rtmp: // IP: 1935 / app\_name / stream\_name -vcodec libx264 -vf scale = 1280x720 -b: v 4M -b: a 192k -f flv rtmp: // IP: 1935 / app\_name / stream\_name2

nginx config:

5. Transcoder 3 (IP) accepts a 4k rtmp stream and transcodes it into a 480p rtmp stream with the following parameters:

ffmpeg config:

ffmpeg -re -i rtmp: // IP: 1935 / app\_name / stream\_name -vcodec libx264 -vf scale = 854x480 -b: v 2M -b: a 192k -f flv rtmp: // IP: 1935 / app\_name / stream\_name3

nginx config:

6. Transcoder 4 (IP) accepts a 4k rtmp stream and transcodes it into a 240p rtmp stream with the following parameters:

ffmpeg config:

ffmpeg -re -i rtmp: // IP: 1935 / app\_name / stream\_name -vcodec libx264 -vf scale=352x240 -b: v 1M -b: a 192k -f flv rtmp: // IP: 1935 / app\_name / stream\_name4

nginx config:

7. The storage (IP) accepts the 4k rtmp stream from the Distributor and the 1080p rtmp, 720p rtmp, 480p rtmp, 240p rtmp streams from transcoders 1-4 and creates the corresponding hls streams.

**Rabbit MQ Level**



1. The client creates Exchange and a set of queues for all participants in the pipeline as well as a rule for routing messages in the queue.
2. Each pipeline participant connects to listen to their queue.
3. The client sends the nginx config json configuration files for each of the participants in queue

message for storage

message to distributor

message N for transcoder N

1. Each member receives a configuration message and applies it to their nginx.

**Resume**

1. Messaging system work good
2. (solved) We have many issues with nginx/ffmpeg stable work with rtmp treatment, which we have not yet been able to solve
   1. on some instances localhost didn’t work
   2. non stable ffmpeg transcoding
   3. nginx - ffmpeg strange behavior on transcoders
3. Issues, that not solved (and may be newer will be solve) for current architecture
   1. rtmp delay between original stream and transcoded streams. Every time we have different amount of chunks on storage.
   2. original stream should have compatible format for hls
   3. bandwidth for distributor (50 in 250 out for 1 4k file)

All scripts are here:

<https://drive.google.com/drive/folders/1pHCyqCW0tIRXDB32VT9eh_gDMXgPPtqU?usp=sharing>

**Appendix A**

**source**

single file

ffmpeg -i lpclip1.mp4 -ss 00:00:01 -c copy -f flv rtmp://35.198.30.225:1935/stream1/test1

(not start without ss)

stream

ffmpeg -re -fflags +genpts -stream\_loop -1 -i lpclip1.mp4 -ss 00:00:01 -c copy -f flv rtmp://35.198.30.225:1935/stream1/test1

(crashes after first cycle )

All other files (including 4k ) started normal

**distributor**

application stream1 {

live on;

exec /usr/bin/ffmpeg -i rtmp://localhost:1935/stream1/test1

-c:a copy -c:v copy -f flv rtmp://35.242.172.58:1935/hls1/test1\_original

-c:a copy -c:v copy -f flv rtmp://35.233.205.84:1935/stream1/test1

-c:a copy -c:v copy -f flv rtmp://35.197.27.59:1935/stream1/test1

-c:a copy -c:v copy -f flv rtmp://35.197.4.197:1935/stream1/test1

-c:a copy -c:v copy -f flv rtmp://35.233.151.248:1935/stream1/test1;

}

worker\_processes 1;

pid /run/nginx.pid;

events {

worker\_connections 768;

# multi\_accept on;

}

# RTMP configuration

rtmp {

server {

listen 1935; # Listen on standard RTMP port

chunk\_size 4000;

include /etc/nginx/app-enabled/\*;

}

}

**transcoder 1080 (example)**

worker\_processes 1;

pid /run/nginx.pid;

events {

worker\_connections 768;

# multi\_accept on;

}

# error\_log /var/log/nginx/rtmp.log debug;

# RTMP configuration

rtmp {

server {

listen 1935; # Listen on standard RTMP port

chunk\_size 4000;

include /etc/nginx/app-enabled/\*;

}

}

application stream1 {

live on;

exec /usr/bin/ffmpeg -i rtmp://localhost:1935/stream1/test1

-vcodec libx264 -b:v 6000000 -vf scale=1920x1080 -f flv rtmp://35.242.172.58:1935/hls1/test1\_1080p;

}

**Storage**

worker\_processes 1;

pid /run/nginx.pid;

events {

worker\_connections 768;

# multi\_accept on;

}

# RTMP configuration

rtmp {

server {

listen 1935; # Listen on standard RTMP port

chunk\_size 4000;

include /etc/nginx/app-enabled/\*;

}

}

**for single file**

application hls1 {

live on;

hls on;

hls\_fragment 5s;

hls\_path /opt/data/hls1;

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 \_240 BANDWIDTH=800000,RESOLUTION=352x240;

hls\_variant \_480 BANDWIDTH=1500000,RESOLUTION=854x480;

hls\_variant \_720 BANDWIDTH=4000000,RESOLUTION=1280x720;

hls\_variant \_1080 BANDWIDTH=6000000,RESOLUTION=1920x1080;

hls\_variant \_original BANDWIDTH=8000000,RESOLUTION=1920x1080;

}