07-SRS流媒体服务器-HTTP-FLV框架分析

《FFmpeg/WebRTC/RTMP音视频流媒体高级开发教程》
零声学院 Darren老师：QQ326873713
课程链接：https://ke.qq.com/course/468797?tuin=137bb271

http-flv技术的实现

HTTP协议中有个约定：content-length字段，http的body部分的长度
服务器回复http请求的时候如果有这个字段，客户端就接收这个长度的数据然后就认为数据传输完成了，
如果服务器回复http请求中没有这个字段，客户端就一直接收数据，直到服务器跟客户端的socket连接断开。

http-flv直播就是利用了这个原理，服务器回复客户端请求的时候不加content-length字段，在回复了http
内容之后，紧接着发送flv数据，客户端就一直接收数据了。

请求SRS返回的是：

```
 HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Type: video/x-flv
Server: SRS/3.0.141(OuXuli)
Transfer-Encoding: chunked
```

配置文件

主要分为两部分

（1）配置http 服务

（2）配置http-flv服务

配置文件如下所示：

```
listen              1935;
max_connections     1000;
#srs_log_tank       file;
#srs_log_file       ./objs/srs.log;
# 前台运行
daemon off;
# 打印到终端控制台
srs_log_tank        console;
http_api {
    enabled         on;
    listen          1985;
}
http_server {
    enabled         on;
    listen          8081; # http监听端口 (1)配置的http服务器，注意端口，如果是云服务器一定要注意开放相应端口
    dir             ./objs/nginx/html;
}
stats {
    network         0;
    disk            sda sdb xvda xvdb;
}
vhost __defaultVhost__ {    # 使用默认的vhost
    # hls darren
    hls {
        enabled         on;
        hls_path        ./objs/nginx/html;
        hls_fragment    10;
        hls_window      60;
```

```
    }
    # http-flv darren
    http_remux {
        enabled     on;
        mount       [vhost]/[app]/[stream].flv;        # 支持flv的使用
        hstrs       on;
    }
}
```

开发步骤

在客户端进行推流验证

ffmpeg -re -i source.200kbps.768x320.flv -vcodec copy -acodec copy -f flv -y rtmp://111.229.231.225/live/livestream

在客户端拉流验证

ffplay http://111.229.231.225:8081/live/livestream.flv

ffplay rtmp://111.229.231.225/live/livestream

重点难点

- 数据怎么来
- 客户端怎么连接

SrsLiveStream::do_serve_http 处理客户端的数据发送

框架

每个http client连接对应一个SrsHttpConn，和SrsRtmpConn连接类似。

每个SrsHttpConn也会对应一个消费者SrsConsumer，即是SrsConsumer对应rtmp、http-flv都是通用的，作为中间数据的缓存

相关类说明

SrsBufferCache HTTP直播流编码器的缓存

SrsFlvStreamEncoder 将RTMP转成HTTP FLV流

SrsTsStreamEncoder 将RTMP转成HTTP TS流

SrsAacStreamEncoder 将RTMP含有的AAC成分转成HTTP AAC流

SrsMp3StreamEncoder 将RTMP含有的MP3成分转成HTTP MP3流
SrsBufferWriter 将流直接写入到HTTP响应
SrsLiveStream HTTP直播流，将RTMP转成HTTP-FLV或者其他格式，其实际是handler
SrsLiveEntry 直播入口，用来处理HTTP 直播流
SrsHttpStreamServer HTTP直播流服务，服务FLV/TS/MP3/AAC流
SrsHttpResponseWriter 负责将数据发送给客户端，本质是调用SrsStSocket进行发送
SrsHttpServeMux HTTP请求多路复用器，说白了就是路由，里面记录了path以及对应的handler。

#0 SrsHttpResponseWriter::writev (this=0x7ffff7f1ebd0, iov=0xaeaa80, iovcnt=240, pnwrite=0x0)
    at src/service/srs_service_http_conn.cpp:784
#1 0x00000000004fde62 in SrsBufferWriter::writev (this=0x7ffff7f1e860, iov=0xaeaa80, iovcnt=240, pnwrite=0x0)
    at src/app/srs_app_http_stream.cpp:511
#2 0x000000000040f109 in SrsFlvTransmuxer::write_tags (this=0xb92fb0, msgs=0xaea310, count=80)
    at src/kernel/srs_kernel_flv.cpp:538
#3 0x00000000004fd0b1 in SrsFlvStreamEncoder::write_tags (this=0xb51490, msgs=0xaea310, count=80)
    at src/app/srs_app_http_stream.cpp:345
#4 0x00000000004ff0dc in SrsLiveStream::do_serve_http (this=0xa3d9f0, w=0x7ffff7f1ebd0, r=0xb92840)
    at src/app/srs_app_http_stream.cpp:677
#5 0x00000000004fe108 in SrsLiveStream::serve_http (this=0xa3d9f0, w=0x7ffff7f1ebd0, r=0xb92840)
    at src/app/srs_app_http_stream.cpp:544
#6 0x000000000049c86f in SrsHttpServeMux::serve_http (this=0xa11fe0, w=0x7ffff7f1ebd0, r=0xb92840)
    at src/protocol/srs_http_stack.cpp:711
#7 0x0000000000562080 in SrsHttpServer::serve_http (this=0xa11e00, w=0x7ffff7f1ebd0, r=0xb92840)
    at src/app/srs_app_http_conn.cpp:300
#8 0x000000000049d6be in SrsHttpCorsMux::serve_http (this=0xb37440, w=0x7ffff7f1ebd0, r=0xb92840)
    at src/protocol/srs_http_stack.cpp:859
#9 0x0000000000561086 in SrsHttpConn::process_request (this=0xb93ff0, w=0x7ffff7f1ebd0, r=0xb92840)
    at src/app/srs_app_http_conn.cpp:161

#10 0x0000000000560ce8 in SrsHttpConn::do_cycle (this=0xb93ff0) at
src/app/srs_app_http_conn.cpp:133
---Type <return> to continue, or q <return> to quit---
#11 0x00000000004d10fb in SrsConnection::cycle (this=0xb93ff0) at
src/app/srs_app_conn.cpp:171
#12 0x0000000000509c88 in SrsSTCoroutine::cycle (this=0xb93f10) at
src/app/srs_app_st.cpp:198
#13 0x0000000000509cfd in SrsSTCoroutine::pfn (arg=0xb93f10) at
src/app/srs_app_st.cpp:213
#14 0x00000000005bdd9d in _st_thread_main () at sched.c:337
#15 0x00000000005be515 in st_thread_create (start=0x5bd719 <_st_vp_schedule+170>,
arg=0x900000001, joinable=1,
    stk_size=1) at sched.c:616

SrsHttpResponseWriter 往客户端写入数据

http监听

服务器启动时http端口的监听过程如下：
run_master()-->SrsServer::listen()--->SrsServer::listen_http_stream()

当http client产生连接时，大体的流程和rtmp类似，只是对于http-flv而言，在

rtmp推流时

推流的时候根据url创建对应的handler，拉流的时候根据url找到对应处理的handler

这里的流程目的是创建一个http-flv的source?
#0 SrsLiveStream::SrsLiveStream (this=0xa3da40, s=0xa3bbd0, r=0xa3ad40, c=0xa3d520)
    at src/app/srs_app_http_stream.cpp:514
#1 0x00000000005010bb in SrsHttpStreamServer::http_mount (this=0xa11fd0, s=0xa3bbd0,
r=0xa3ad40)
    at src/app/srs_app_http_stream.cpp:912
#2 0x00000000005620f5 in SrsHttpServer::http_mount (this=0xa11e00, s=0xa3bbd0,
r=0xa3ad40)
    at src/app/srs_app_http_conn.cpp:308
#3 0x00000000004cd3cc in SrsServer::on_publish (this=0xa11ea0, s=0xa3bbd0, r=0xa3ad40)
    at src/app/srs_app_server.cpp:1608

#4 0x00000000004e6a9b in SrsSource::on_publish (this=0xa3bbd0) at src/app/srs_app_source.cpp:2466
#5 0x00000000004d89f2 in SrsRtmpConn::acquire_publish (this=0xa30d00, source=0xa3bbd0)
    at src/app/srs_app_rtmp_conn.cpp:940
#6 0x00000000004d7a74 in SrsRtmpConn::publishing (this=0xa30d00, source=0xa3bbd0) at src/app/srs_app_rtmp_conn.cpp:822
#7 0x00000000004d5229 in SrsRtmpConn::stream_service_cycle (this=0xa30d00) at src/app/srs_app_rtmp_conn.cpp:534
#8 0x00000000004d4141 in SrsRtmpConn::service_cycle (this=0xa30d00) at src/app/srs_app_rtmp_conn.cpp:388
#9 0x00000000004d2f09 in SrsRtmpConn::do_cycle (this=0xa30d00) at src/app/srs_app_rtmp_conn.cpp:209
#10 0x00000000004d10fb in SrsConnection::cycle (this=0xa30d78) at src/app/srs_app_conn.cpp:171
#11 0x0000000000509c88 in SrsSTCoroutine::cycle (this=0xa30f90) at src/app/srs_app_st.cpp:198
#12 0x0000000000509cfd in SrsSTCoroutine::pfn (arg=0xa30f90) at src/app/srs_app_st.cpp:213
#13 0x00000000005bdd9d in _st_thread_main () at sched.c:337
#14 0x00000000005be515 in st_thread_create (start=0x5bd719 <_st_vp_schedule+170>, arg=0x700000001, joinable=1,
    stk_size=1) at sched.c:616

http-flv播放

[2020-08-06 17:42:20.027][Trace][10457][554] HTTP client ip=175.0.54.116, request=0, to=15000ms
[2020-08-06 17:42:20.027][Trace][10457][554] HTTP GET http://111.229.231.225:8081/live/livestream.flv, content-length=-1
[2020-08-06 17:42:20.027][Trace][10457][554] http: mount flv stream for sid=/live/livestream, mount=/live/livestream.flv
[2020-08-06 17:42:20.027][Trace][10457][554] flv: source url=/live/livestream, is_edge=0, source_id=-1[-1]
[2020-08-06 17:42:20.027][Trace][10457][554] create consumer, active=0, queue_size=0.00, jitter=30000000
[2020-08-06 17:42:20.027][Trace][10457][554] set fd=10, SO_SNDBUF=46080=>175000, buffer=350ms

[2020-08-06 17:42:20.027][Trace][10457][554] FLV /live/livestream.flv, encoder=FLV, nodelay=0, mw_sleep=350ms, cache=0, msgs=128

每个播放的SrsFlvStreamEncoder是独立的

#0 SrsFlvStreamEncoder::SrsFlvStreamEncoder (this=0xa57820) at src/app/srs_app_http_stream.cpp:250
#1 0x00000000004fe2fd in SrsLiveStream::do_serve_http (this=0xa3da20, w=0x7ffff7eb5bd0, r=0xa5d7c0)
    at src/app/srs_app_http_stream.cpp:562
#2 0x00000000004fe108 in SrsLiveStream::serve_http (this=0xa3da20, w=0x7ffff7eb5bd0, r=0xa5d7c0)
    at src/app/srs_app_http_stream.cpp:544
#3 0x000000000049c86f in SrsHttpServeMux::serve_http (this=0xa11fe0, w=0x7ffff7eb5bd0, r=0xa5d7c0)
    at src/protocol/srs_http_stack.cpp:711
#4 0x0000000000562080 in SrsHttpServer::serve_http (this=0xa11e00, w=0x7ffff7eb5bd0, r=0xa5d7c0)
    at src/app/srs_app_http_conn.cpp:300
#5 0x000000000049d6be in SrsHttpCorsMux::serve_http (this=0xa52930, w=0x7ffff7eb5bd0, r=0xa5d7c0)
    at src/protocol/srs_http_stack.cpp:859
#6 0x0000000000561086 in SrsHttpConn::process_request (this=0xa5d120, w=0x7ffff7eb5bd0, r=0xa5d7c0)
    at src/app/srs_app_http_conn.cpp:161
#7 0x0000000000560ce8 in SrsHttpConn::do_cycle (this=0xa5d120) at src/app/srs_app_http_conn.cpp:133
#8 0x00000000004d10fb in SrsConnection::cycle (this=0xa5d120) at src/app/srs_app_conn.cpp:171
#9 0x0000000000509c88 in SrsSTCoroutine::cycle (this=0xa5d1c0) at src/app/srs_app_st.cpp:198
#10 0x0000000000509cfd in SrsSTCoroutine::pfn (arg=0xa5d1c0) at src/app/srs_app_st.cpp:213
#11 0x00000000005bdd9d in _st_thread_main () at sched.c:337