09-SRS流媒体服务器-Forward模式

1 Forward集群原理和配置

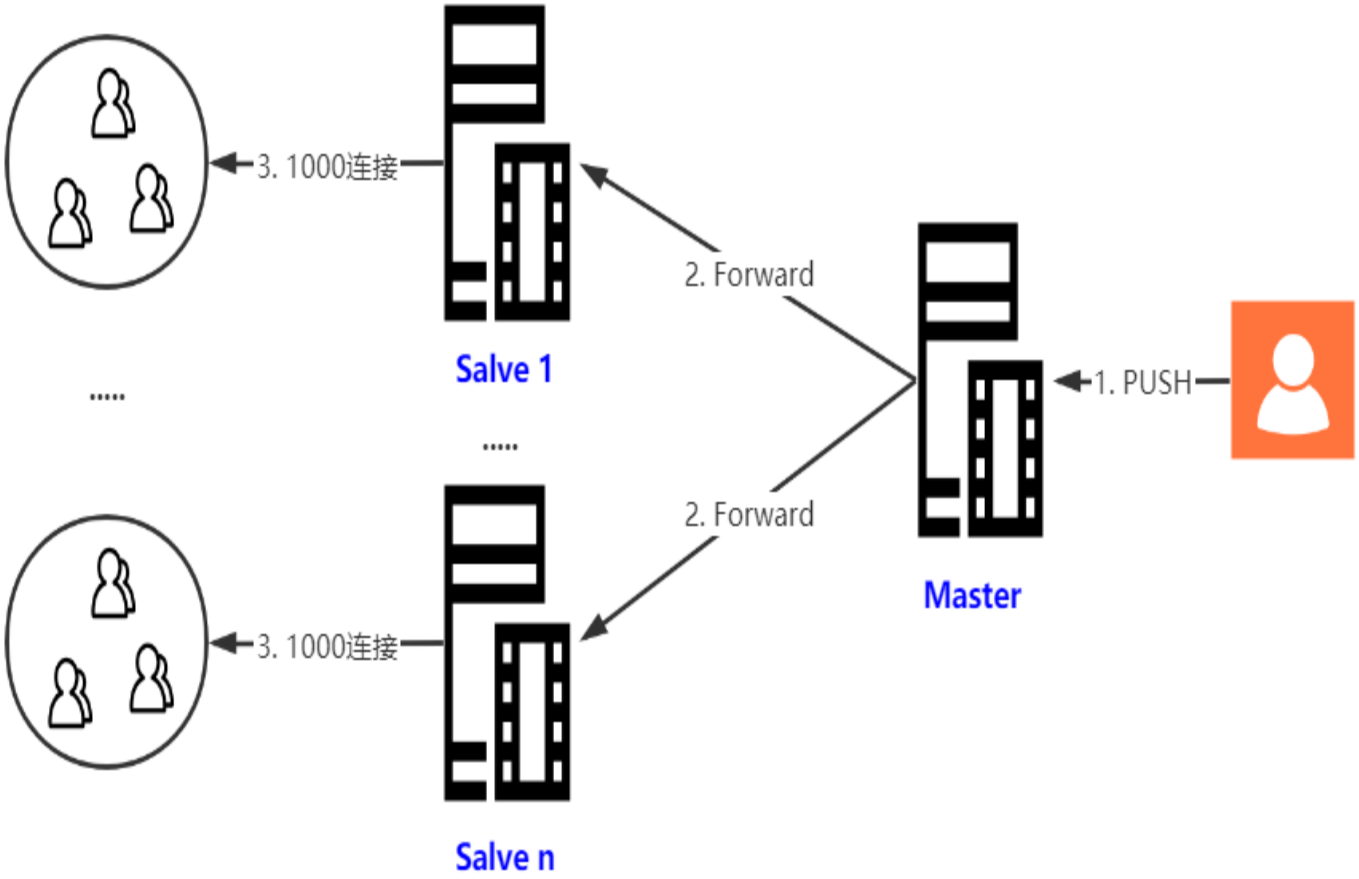
Forward 翻译成中文是向前、前头的、发送等（来自google翻译），还有好多词性。

在SRS中可以理解为把Master节点获得直播流**广播**给所有的Slave节点。我觉得广播这个词可能要比forward更容易理解。也可以理解为转发，即master把接收到的所有流都转发给Slave节点，即master节点由多少路直播流，那么在每个slave节点也会多少路直播流。
注：在SRS中还有另外一种集群方式，edge方式。注意两种方式的用词不同。

在Forward模式中，**中心节点叫Master，边缘节点叫Slave**；在edge模式中，**中心节点叫origin（源站），边缘节点叫做edge**。

适用场景

Forward适合搭建小型集群，为什么这么说呢？因为每个slave节点都和master节点有相同数量的直播流，那么请看下图（从右往左，1、2、3是步骤）：



推流者推流给master，那么master就会Forward到每一个Slave。那么在slave节点上不论需不需要都会有master过来的流。试想一下，如果推流者的数量为10，那么master到slave之间的带宽就是：带宽=10 * slave的个数 *直播流码率，随着slave的增多，master的出口带宽会不断提高。而现实是，在某些slave节点其实根本没有人看……这样就造成了master到slave之间的带宽浪费。

所以说Forward适合用于搭建小型集群。

RTMP流转发（Forward）部署实例

SRS可以将送到SRS的流转发给其他RTMP服务器，实现简单集群/热备功能，也可以实现一路流热备（譬如编码器由于带宽限制，只能送一路流到RTMP服务器，要求RTMP服务器能将这路流也转发给其他RTMP备用服务器，实现主备容错集群）。

假设服务器的IP是：111.229.231.225

Forward就是SRS将流拷贝输出给其他的RTMP服务器，以SRS转发给SRS为例：

- 主SRS：Master，编码器推流到主SRS，主SRS将流处理的同时，将流转发到备SRS
- 备SRS：Slave，主SRS转发流到备SRS，就像编码器推送流到备用SRS一样。 我们的部署实例中，主SRS侦听1935端口，备SRS侦听19350端口。

第一步，编写主SRS配置文件。详细参考Forward
将以下内容保存为文件，譬如`conf/forward.master.conf`，服务器启动时指定该配置文件(srs的conf文件夹有该文件)。

```
# conf/forward.master.conf
listen              1935;
max_connections     1000;
pid                 ./objs/srs.master.pid;
srs_log_tank        file;
srs_log_file        ./objs/srs.master.log;
vhost __defaultVhost__ {
    forward {
        enabled on;
        destination 127.0.0.1:19350 127.0.0.1:19351; # forward目的地址，怎么给出多个地址?
    }
}
```

多个forward时比如：192.168.1.6:1935 192.168.1.7:1935;

第二步，启动主SRS，主SRS将流转发到从SRS。详细参考Forward
./objs/srs -c conf/forward.master.conf
第三步，编写从SRS配置文件。详细参考Forward
将以下内容保存为文件，譬如`conf/forward.slave1.conf, conf/forward.slave2.conf`，服务器启动时指定该配置文件(srs的conf文件夹有该文件)。

```
# conf/forward.slave1.conf
listen              19350;
pid                 ./objs/srs.slave1.pid;
srs_log_tank        file;
srs_log_file        ./objs/srs.slave1.log;
vhost __defaultVhost__ {
}
```

```
# conf/forward.slave2.conf
```

```
listen              19351;
pid                 ./objs/srs.slave2.pid;
srs_log_tank        file;
srs_log_file        ./objs/srs.slave2.log;
vhost __defaultVhost__ {
}
```

第四步，启动从SRS，主SRS将流转发到从SRS。详细参考Forward

./objs/srs -c conf/forward.slave1.conf

./objs/srs -c conf/forward.slave2.conf

注意：启动srs后查看下srs是否启动成功，错误可以查看日志。

[winlin@dev6 srs]$ sudo netstat -anp|grep srs

tcp 0 0 0.0.0.0:1935 0.0.0.0:* LISTEN 7826/srs

tcp 0 0 0.0.0.0:19350 0.0.0.0:* LISTEN 7834/srs

tcp 0 0 0.0.0.0:19351 0.0.0.0:* LISTEN 7834/srs

第五步，启动推流编码器。详细参考Forward

使用FFMPEG命令循环推流：

```
#!/bin/bash
for((;;)); do \
    ffmpeg -re -i ./doc/source.200kbps.768x320.flv \
        -vcodec copy -acodec copy \
        -f flv -y rtmp://111.229.231.225/live/livestream; \
        sleep 1; \
done
find
```

涉及的流包括：

- 编码器推送的流：rtmp://111.229.231.225/live/livestream
- 主SRS转发的流：rtmp://111.229.231.225:19350/live/livestream
- 主SRS转发的流：rtmp://111.229.231.225:19351/live/livestream
- 观看主SRS的流：rtmp://111.229.231.225/live/livestream
- 观看从1 SRS的流：rtmp://111.229.231.225:19350/live/livestream
- 观看从2 SRS的流：rtmp://111.229.231.225:19351/live/livestream

第六步，观看主SRS的RTMP流。详细参考Forward

RTMP流地址为：`rtmp://111.229.231.225/live/livestream`
可以使用VLC观看。
或者使用在线SRS播放器播放：srs-player
备注：请将所有实例的IP地址111.229.231.225都换成部署的服务器IP地址。
第七步，观看从SRS的RTMP流。详细参考Forward
RTMP流地址为：`rtmp://111.229.231.225:19350/live/livestream`
`rtmp://111.229.231.225:19351/live/livestream`
可以使用VLC观看。
或者使用在线SRS播放器播放：srs-player-19350
备注：请将所有实例的IP地址111.229.231.225都换成部署的服务器IP地址。

改进版本
使用Haproxy做负载均衡，在master前加上Haproxy，

2 Forward集群源码分析

从原理上来分析，要实现forward功能：
1. 读取配置文件获取forward server的地址
2. 创建RTMP推流客户端
3. 从source里面拉取消息，然后推送给forward server

从配置文件入手

```
# conf/forward.master.conf
listen              1935;
max_connections     1000;
pid                 ./objs/srs.master.pid;
srs_log_tank        file;
srs_log_file        ./objs/srs.master.log;
vhost __defaultVhost__ {
   forward {
       enabled on;
       destination 127.0.0.1:19350 127.0.0.1:19351; # forward目的地址，怎么给出多个地址?
    }
}
```

从

```
bool SrsConfig::get_forward_enabled(string vhost)
{
    static bool DEFAULT = false;

    SrsConfDirective* conf = get_vhost(vhost);
    if (!conf) {
        return DEFAULT;
    }

    conf = conf->get("forward");
    if (!conf) {
        return DEFAULT;
    }

    conf = conf->get("enabled");
    if (!conf || conf->arg0().empty()) {
        return DEFAULT;
    }

    return SRS_CONF_PERFER_FALSE(conf->arg0());
}

SrsConfDirective* SrsConfig::get_forwards(string vhost)
{
    SrsConfDirective* conf = get_vhost(vhost);
    if (!conf) {
        return NULL;
    }

    conf = conf->get("forward");
    if (!conf) {
        return NULL;
    }

    return conf->get("destination");
}
```

断点分析

SrsConfig::get_forward_enabled

(gdb) bt
#0 SrsConfig::get_forward_enabled (this=0xa0fcf0, vhost="__defaultVhost__") at
src/app/srs_app_config.cpp:4821
#1 0x00000000004e1aa2 in SrsOriginHub::create_forwarders (this=0xa3a5b0) at
src/app/srs_app_source.cpp:1467
#2 0x00000000004e053c in SrsOriginHub::on_publish (this=0xa3a5b0) at
src/app/srs_app_source.cpp:1120
#3 0x00000000004e6a0b in SrsSource::on_publish (this=0xa3a120) at
src/app/srs_app_source.cpp:2460
#4 0x00000000004d89f2 in SrsRtmpConn::acquire_publish (this=0xa2fca0, source=0xa3a120)
    at src/app/srs_app_rtmp_conn.cpp:940
#5 0x00000000004d7a74 in SrsRtmpConn::publishing (this=0xa2fca0, source=0xa3a120)
    at src/app/srs_app_rtmp_conn.cpp:822
#6 0x00000000004d5229 in SrsRtmpConn::stream_service_cycle (this=0xa2fca0) at
src/app/srs_app_rtmp_conn.cpp:534
#7 0x00000000004d4141 in SrsRtmpConn::service_cycle (this=0xa2fca0) at
src/app/srs_app_rtmp_conn.cpp:388
#8 0x00000000004d2f09 in SrsRtmpConn::do_cycle (this=0xa2fca0) at
src/app/srs_app_rtmp_conn.cpp:209
#9 0x00000000004d10fb in SrsConnection::cycle (this=0xa2fd18) at
src/app/srs_app_conn.cpp:171
#10 0x0000000000509c88 in SrsSTCoroutine::cycle (this=0xa2ff50) at
src/app/srs_app_st.cpp:198
#11 0x0000000000509cfd in SrsSTCoroutine::pfn (arg=0xa2ff50) at
src/app/srs_app_st.cpp:213
#12 0x00000000005bdd9d in _st_thread_main () at sched.c:337
#13 0x00000000005be515 in st_thread_create (start=0x5bd719 <_st_vp_schedule+170>,
arg=0x700000001, joinable=1,
    stk_size=1) at sched.c:616

SrsConfig::get_forwards

#0 SrsConfig::get_forwards (this=0xa0fcf0, vhost="__defaultVhost__") at
src/app/srs_app_config.cpp:4843

#1 0x00000000004e1b0d in SrsOriginHub::create_forwarders (this=0xa3a5b0) at src/app/srs_app_source.cpp:1471
#2 0x00000000004e053c in SrsOriginHub::on_publish (this=0xa3a5b0) at src/app/srs_app_source.cpp:1120
#3 0x00000000004e6a0b in SrsSource::on_publish (this=0xa3a120) at src/app/srs_app_source.cpp:2460
#4 0x00000000004d89f2 in SrsRtmpConn::acquire_publish (this=0xa2fca0, source=0xa3a120)
    at src/app/srs_app_rtmp_conn.cpp:940
#5 0x00000000004d7a74 in SrsRtmpConn::publishing (this=0xa2fca0, source=0xa3a120)
    at src/app/srs_app_rtmp_conn.cpp:822
#6 0x00000000004d5229 in SrsRtmpConn::stream_service_cycle (this=0xa2fca0) at src/app/srs_app_rtmp_conn.cpp:534
#7 0x00000000004d4141 in SrsRtmpConn::service_cycle (this=0xa2fca0) at src/app/srs_app_rtmp_conn.cpp:388
#8 0x00000000004d2f09 in SrsRtmpConn::do_cycle (this=0xa2fca0) at src/app/srs_app_rtmp_conn.cpp:209
#9 0x00000000004d10fb in SrsConnection::cycle (this=0xa2fd18) at src/app/srs_app_conn.cpp:171
#10 0x0000000000509c88 in SrsSTCoroutine::cycle (this=0xa2ff50) at src/app/srs_app_st.cpp:198
#11 0x0000000000509cfd in SrsSTCoroutine::pfn (arg=0xa2ff50) at src/app/srs_app_st.cpp:213
#12 0x00000000005bdd9d in _st_thread_main () at sched.c:337

SrsForwarder::on_meta_data

#0 SrsForwarder::on_meta_data (this=0xa3b960, shared_metadata=0xa45f00) at src/app/srs_app_forward.cpp:114
#1 0x00000000004dea69 in SrsOriginHub::on_meta_data (this=0xa3a5b0, shared_metadata=0xa45f00, packet=0xa45e60
    at src/app/srs_app_source.cpp:924
#2 0x00000000004e516f in SrsSource::on_meta_data (this=0xa3a120, msg=0xa45c80, metadata=0xa45e60)
    at src/app/srs_app_source.cpp:2116
#3 0x00000000004d91bb in SrsRtmpConn::process_publish_message (this=0xa2fca0, source=0xa3a120, msg=0xa45c80)
    at src/app/srs_app_rtmp_conn.cpp:1045
#4 0x00000000004d8dce in SrsRtmpConn::handle_publish_message (this=0xa2fca0, source=0xa3a120, msg=0xa45c80)
    at src/app/srs_app_rtmp_conn.cpp:993

#5 0x00000000005810b6 in SrsPublishRecvThread::consume (this=0x7ffff7f66800, msg=0xa45c80)
    at src/app/srs_app_recv_thread.cpp:389
#6 0x000000000057fbd4 in SrsRecvThread::do_cycle (this=0x7ffff7f66808) at src/app/srs_app_recv_thread.cpp:146
#7 0x000000000057fa25 in SrsRecvThread::cycle (this=0x7ffff7f66808) at src/app/srs_app_recv_thread.cpp:115
#8 0x0000000000509c88 in SrsSTCoroutine::cycle (this=0xa3a840) at src/app/srs_app_st.cpp:198
#9 0x0000000000509cfd in SrsSTCoroutine::pfn (arg=0xa3a840) at src/app/srs_app_st.cpp:213
#10 0x00000000005bdd9d in _st_thread_main () at sched.c:337
#11 0x00000000005be515 in st_thread_create (start=0xa3bc30, arg=0x7ffff7f66530, joinable=32767,
    stk_size=-134847200) at sched.c:616

SrsForwarder::on_video

SrsForwarder::forward

推流核心所在

SrsForwarder::on_meta_data

每个SrsForwarder对应一个forward server