[文章编号] 1007－7405(2018)02－0156－05

基于ATS的直播加速缓存系统设计

马邦阳[1]，魏伟刚[2]，浦云明[2]，尤志宁[2]，王 巍[2]，陈凯萌[2]

（1. 集美大学网络中心，福建 厦门 361021；2. 集美大学计算机工程学院，福建 厦门 361021）

[摘要] 由于网络直播快速发展和在线用户数量过大，服务器容易出现过载状态，响应速度也随之下降，直播卡顿和延时问题成为了困扰大多数用户的问题。因此提出，基于ATS（apache traffic server），采用Boost库开发插件并搭建网络代理服务器，通过重定向服务器将用户的直播请求导入ATS服务器，ATS以异步处理和并发的方式处理用户请求。当多个用户请求同一个直播资源时，ATS缓存系统只需要回源一次，并把数据输出给多个用户，有效地减少了广域网和源站服务器的负载，提高了直播数据的传输效率。实验结果表明，ATS缓存加速系统可以解决直播的卡顿和延时问题。

[关键词] 网络直播；ATS；数据回源/输出；缓存加速；卡顿

[中图分类号] TP 311

The Design of Live Accelerated Cache System Based on ATS

MA Bangyang[1], WEI Weigang[2], PU Yunming[2], YOU Zhining[2], WANG Wei[2], CHEN Kaimeng[2]

(1. Network Center, Jimei University, Xiame 361021, China;
2. Comupter Engineering College, Jimei University, Xiamen 361021, China)

Abstract: With the rapid development of live webcast and the large number of live users, the internet server overload often happened, so the response time becomes slowly, the delay and live Kartun becomes a problem for most users. Therefore, the main goal of this project develops a plug-in software for the proxy server using boost library based on ATS. The system redirects the user's request to the ATS server. ATS handles the user request in asynchronous and concurrency way. When multiple users request the same live resource, ATS live cache system only needs to fetch once, and then the data output to multiple users, effectively reducing the WAN and the source server load, Improve the transmission efficiency of live data. As result of the experiment, the ATS live accelerated cache system solved the problem of internet live delay and Kartun.

Keywords: internet live; ATS; fetch/output; cache accelerate; Kartun

0 引言

网络上直播数据的请求和传输都要通过网络直播服务器处理，当用户过多，服务器即处于过载状态，响应速度也会随之下降，出现网络延迟。王彦辉等[1]基于立方体网络结构，进行了网络传输延迟分析，认为网络中存储和转发都要耗费一定的系统资源，因而信息传输节点数决定了通信效率的高低。越来越多研究关注用户访问和服务器的交互作用。李锐等[2]从负载均衡性方面出发，研究了分

[收稿日期] 2017－08－07 [修回日期] 2017－11－09
[基金项目] 厦门市科技计划项目（3502Z20173028）；福建省科技计划项目（2017H0026，2017R0076，2017J05104）
[作者简介] 马邦阳（1979—），男，实验师，从事网络系统设计和安全研究。

布式缓存副本策略，匹配服务器处理性能最大化利用缓存资源。张国强等[3]研究了 ICN（信息中心网络）缓存网络中缓存大小规划、应用无关的缓存空间共享机制、缓存决策策略、网络内置缓存对象的可用性以及缓存网络的理论模型和性能优化方法等。蔡君等[4]研究了 ICN 全网内置缓存架构，为使被缓存的内容对象在空间和时间上分布更合理，提出了一种基于节点社团重要度的缓存策略（CSNIC），保证了不同流行度内容对象在各社团内节点处的时间分布的合理性。现有的相关研究工作，各类缓存策略都侧重于提高系统整体性能，而在企业实际的应用中，为解决网络中来回传送数据的问题，一般使用缓存设备实现本地存储，同时缓存设备与源服务器进行刷新检验，以保证存储在缓存中数据的时效性。

ATS（apache traffic server）是模块化的 Http 网络代理缓存服务器。ATS 缓存包括一个高速的对象数据库，数据库根据用户请求的 URL 和请求头部中相关的参数来组成对应的 Key，用于索引对象[5]。李润知等[8]研究了缓存的新鲜度及数据分片点击率等指标，评估节点缓存空间利用率，提出了频度限制的缓存替换算法。在新鲜度检测中，如果内容新鲜，则 ATS 会向客户端发送回复标头；否则流量服务器将打开到源服务器的连接，请求内容并检测资源新鲜度。如果资源新鲜，ATS 将该对象发给用户，如果资源不新鲜，则 ATS 缓存新鲜的对象并转发给用户。在任何阶段出现错误，Http 状态机将跳转到“Send Reply Header”状态并发送回复，如果回复错误，则事务关闭。当客户端向服务器请求一个资源时，当资源在其他网络位置，服务器向客户端打回一个重定向的回复，客户端根据该回复去正确的 URL 获取资源[9]。

ATS 自身没有直播代理的功能，因此，系统需要开发直播代理的插件。ATS 插件开发是基于协程 TSCont、钩子 Hook 和事件池来实现的。本文采用基于 ATS 的网络直播加速缓存系统设计，以期解决网络直播的延时和卡顿问题，同时也减少网络的负载。

1　系统模块设计

1.1　系统模块

在 ATS 缓存系统中，主要是设计事务处理模块，图 1 为系统获取关键字流程，主要功能是调用 Boost 库获取关键字、检索关键字、源站请求资源、给用户发送资源等[10]。

1）获取关键字：由于 ATS 中的资源都是以关键字唯一表示，而关键字又是从用户的请求中获取的，因此，处理模块首先请求、获取关键字。

2）检索关键字：获取关键字之后，对关键字进行检索，查看该资源是否存在于本地，若存在，则给用户发送该资源，如果不存在，则 ATS 将会去源站进行请求，先获取该资源，再将该资源发送给用户。

3）去源站请求资源：当用户请求的资源本地不存在时，ATS 将会去源站资源请求。

4）给用户发送资源。

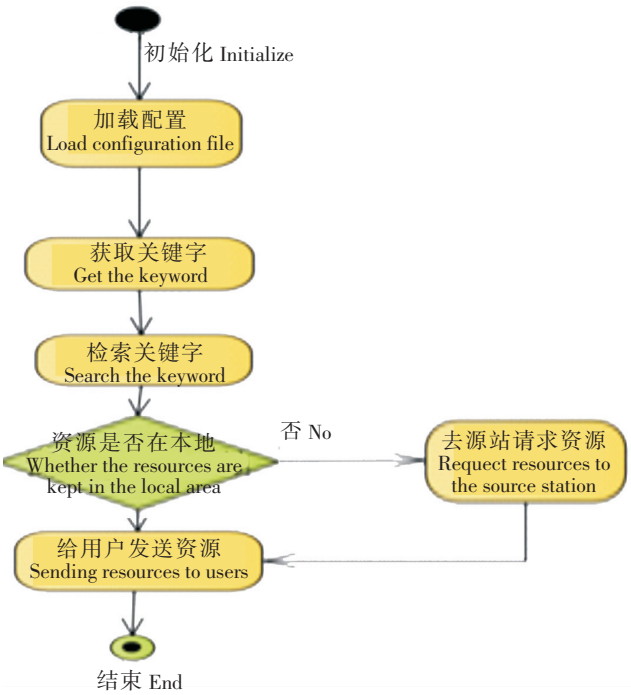

图 1　系统获取关键字流程

Fig.1　System keywords transaction

1.2　获取关键字处理

获取关键字是事务处理的第一步，为后续模块处理做准备，主要是创建功能对象和回调

函数，发送到 ATS 的请求事务一旦触发就进入回调函数。功能入口代码为：

```
if( GLB_CONFIG_MANAGER( ) . get_global_block( ) . pattern_store_switch)
//判断过去关键字模块功能是否开启{
    m_store_url_sp = hm_store_url::create( ) ;  //创建功能对象
    m_store_contp = TSContCreate( hm_store_url::store_url_event_cb,0) ;
      //创建回调函数 store_url_event_cb
    TSContDataSet( m_store_contp,m_store_url_sp. get( ) ) ;
    TSHttpHookAdd( TS_HTTP_POST_REMAP_HOOK,m_store_contp) ;
      //添加全局钩子,触发钩子进入回调函数}
```

代码中，GLB_CONFIG_MANAGER( ) . get_global_block( ) . pattern_store_switch 的返回值为一个布尔值，它代表 store_url 功能的开关，1 表示开，0 表示关。store_url 功能是提取关键字功能，用来表示和存储资源。

URL 关键字提取规则在配置文件中配置，URL 的匹配和存储都是以正则表达式来处理的。当 ATS 接收到请求时将会调用该功能，并将结果写到日志文件 storeurl. log 中。日志字段包括：用户请求时间、用户所在的网络 IP 地址、用户请求的 URL、提取的关键字。

1. 3 检索关键字处理

项目中创建一个 Map 用来存放已存在于本地的资源，创建代码为：

```
Typedef boost::unordered_map < std::string,video_live_stream::pointer > video_stream_hash_container;
video_stream_hash_container m_stream_grp;
```

检索部分的代码为：

```
int video_stream_manage::setup_request( main_state * ms, int &client_num) {
  simple_recursive_guard lock( m_stream_mutex) ;   //程序加锁
  video_stream_hash_iterator iter = m_stream_grp. find( ms - >store_key) ;
  if( iter = = m_stream_grp. end( ) )
    return CTS_HLRS_MISS;  //资源不存在
  else
    return CTS_HLRS_HIT;//返回检索结果 hit,资源存在}
```

当用户所请求的资源不存在于本地时，返回 CTS_HLRS_MISS,否则返回 CTS_HLRS_HIT，系统会自动统计请求一个资源的用户数量。然后将其写入日志文件 videolive. log 中，日志信息包括：用户所在网络 IP、状态码、同时请求该资源的用户数、用户请求的原 URL。状态码为 HTTP_LIVE_HIT 表示用户请求的资源存在于本地，为 HTTP_LIVE_MISS 表示用户请求的资源本地没有，为 HTTP_LIVE_FETCH 表示 ATS 无法从源站获取资源。

1. 4 数据的回源处理

数据的回源指的是当 ATS 接收到用户的数据请求，本地又没有该数据资源，此时 ATS 需要到源站请求数据到本地。程序插件需构建请求头部、创建与源站的连接、解析源站回复的数据。

1）构建请求头部，请求头部的构建主要有两个方法：

① voidvideo_live_stream::init_fetch_request(main_state * ms)

该方法的主要功能是初始化构造头部过程中所需要用到的参数，如源站的 Host 和请求的 URL。需要注意的是在该方法中获取到了用户的 IP 地址和端口号，并存放在 m_client_addr 成员内。

② voidvideo_live_stream::build_request_line(TSHttpTxn txnp,std::string &request_line)

该方法是创建对源站进行请求的 Request，将所需要用到的参数、主机地址、资源的 URL 等尽心

拼接。并将结果放入 Request_Line。

2）创建与源站的连接，创建连接使用了 ATS 的 API。

```
voidvideo_fetch_request::start_connect(const struct sockaddr *addr){
    //创建用于处理数据传输的回调函数;
    //建立读区源站回复的数据通道;
    //建立写入连接的数据通道;}
```

通过该方法建立与源站的连接，M_Req_Reader 存放着与源站建立连接的 Request。将该 Request 写入连接 M_Http_Vc 就可建立与源站的连接。

3）解析源站回复的数据，包括回复头部信息的解析和内容解析。

```
int video_fetch_request::handle_vconn_read_ready();
```

1.5　给用户发送资源处理

给用户发送数据首先要建立连接，然后建立数据传输通道，最后发送数据。

1）连接的建立：通过 ATS 提供的 API 接口来建立与客户端的连接。

```
voidvideo_live_session::start_connect(){
  m_contp = TSContCreate(video_live_session::state_video_live_event,TSMutexCreate());    //封装回调函数
  TSContDataSet(m_contp,this);    //设置回调函数的传入数据
  TSHttpTxnIntercept(m_contp,m_txnp);    //接管事务}
```

TSHttpTxnIntercept 函数的作用是 ATS 服务器接管事务，同时生成一个与客户端的连接 TSVConn。

2）创建数据传输通道。

```
int video_live_session::handle_accept_event(TSCont contp,TSEvent event,
void *edata){
    //PluginVC
    m_net_vc =(TSVConn)edata;
    //edata 是调用 TSHttpTxnIntercept 生成的连接,其类型是 TSVConn
    m_req_buffer = TSIOBufferCreate();
    m_resp_buffer = TSIOBufferCreate();
    m_resp_reader = TSIOBufferReaderAlloc(m_resp_buffer);      //封装缓存区
    m_read_vio = TSVConnRead(m_net_vc,contp,m_req_buffer,INT64_MAX);
    return 0;}
m_write_vio = TSVConnWrite(m_net_vc,m_contp,m_resp_reader,INT64_MAX);
```

上述代码创建了数据传输通道 M_ Write_ Vio，M_ Resp_ Reader 为封装过后的数据缓存区。只要将数据写入该缓存区就可以实现向用户发送数据。

3）发送数据。

```
voidvideo_live_session::produce_output(const void *data,int64_t length)
```

2　实验结果及分析

系统最低硬件环境：CPU 2.40 GHz 以上；RAM 4 G 以上；Disk 空间 512 G 以上。当服务用户较多时，硬件要求也要随之提高，否则服务器容易过载。管理员应监控服务器的负荷情况，特别是直播高峰期时应做适当的负载均衡。软件环境：CentOS 6.5、Python2.7、CURL 开发工具、Tcpdump 开发工具、Gcc、G++、Boost1.56 开发工具。

本研究采集了基于 ATS 缓存系统的一天、一周和一月的流量数据，数据来源于上海移动某运营

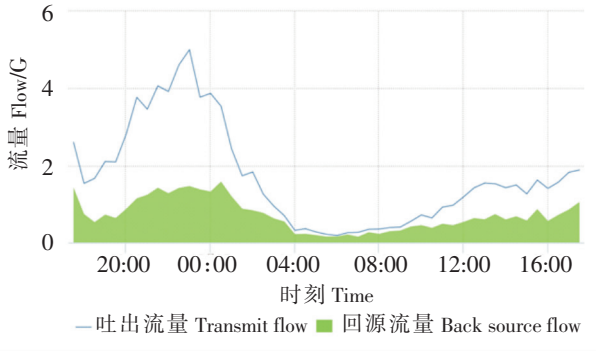

图 2 一天流量情况

Fig.2 Day system data traffic

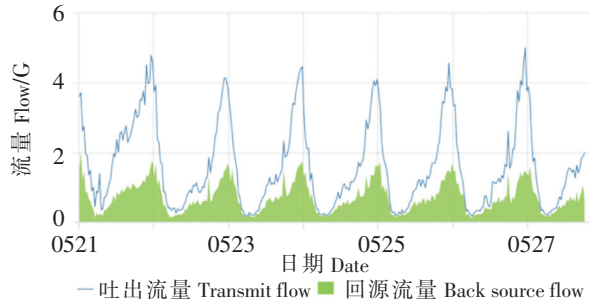

图 3 一周流量情况

Fig.3 Week system data traffic

商的直播数据统计和回源情况，参见图 2 ~ 图 4。可以看出直播加速效果较好，吐出流量减去回源流量等于本系统服务的流量，且吐出流量是回源流量的两倍以上，高峰期可达到更高，如节假日、晚上 21 时至 23 时。从实验测试数据看，ATS 缓存服务达到了预期的设计要求。

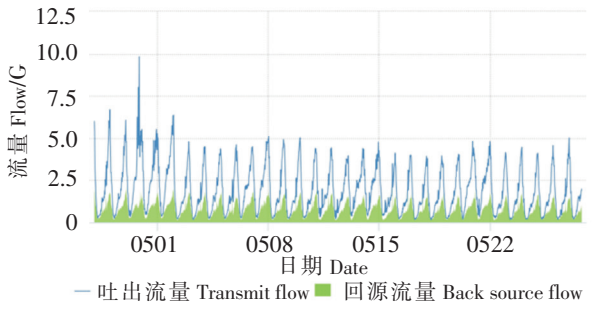

图 4 一个月流量情况

Fig.4 Month system data traffic

3 结束语

系统在上海移动网进行了测试，数据分析表明，网络直播缓存加速达到预期效果，解决了直播的延时和卡顿问题，同时也减少了网络的负载。系统还需进一步的研究，主要包括：1）网络数据块的数据传输，实现 *.ts 和 *.data 等数据类型的直播；2）开发大文件的缓存、加速和反向代理服务等功能；3）热点功能，当用户过多时，如果服务器将每个用户的请求都进行缓存，服务器资源消耗相当大，可以研究热点功能的开设，将访问数量较多的资源进行缓存。

[参 考 文 献]

[1] 王彦辉，张德全. 两类重要网络的传输延迟分析 [J]. 计算机工程与应用，2010，46(18)：87-88.

[2] 李锐，唐旭，石小龙，等. 网络 GIS 中最佳负载均衡的分布式缓存副本策略 [J]. 武汉大学学报（信息科学版），2015，40(10)：1287-1293.

[3] 张国强，李扬，林涛，等，信息中心网络中的内置缓存技术研究 [J]. 软件学报，2014，25(1)：154-175.

[4] 蔡君，余顺争，刘外喜. 基于节点社团重要度的 ICN 缓存策略 [J]. 通信学报，2015，36(6)：2015222. DOI：10.11959/J.issn.1000-436x.2015222.

[5] SARA ALOUF，NICAISE CHOUNGMO FOFACK，NEDKO NEDKOV. Performance models for hierarchy of caches：application to modern DNS caches [J]. Performance Evaluation，2016，97：57-82.

[6] 张振伟，李志雄. 网络视频直播缓存系统设计 [J]. 计算机工程，2002，28(8)：227-230.

[7] 李云飞，谢伟凯，鲁晨平，等. 面向直播 HTTP Streaming 系统的 HTTP 缓存服务器行为优化 [J]. 计算机工程与应用，2012，48(1)：68-74.

[8] 李润知，郭纯一，范明，等. P2P 流媒体直播分布式缓存替换算法研究 [J]. 计算机工程与设计，2011，32(1)：58-61.

[9] 张晓军，吕洁，张蓓. HTTP 重定向在网关认证中的应用 [J]. 大连理工大学学报，2005，45(s1)：48-51.

[10] 罗剑锋. Boost 程序完全开发指南 [M]. 3 版. 北京：电子工业出版社，2015.

（责任编辑 朱雪莲 英文审校 黄振坤）