TCP/HTTP 负载均衡利器
— HAProxy 介绍及实践

Godbach

nylzhaowei@gmail.com

2015/06/07

大纲

- 简介
- 连接模式（Connection Mode）
- 功能列表（Feature List）
- 实例
- Debug & Analysis
- Reference

简介

- TCP/HTTP Load Balancer
- Reliable, High Performance
- 最新 stable 版本： 1.5.12
- 最新 dev 版本：1. 6-dev0
- 官网：www.haproxy.org

连接模式（1）

- Proxy 典型配置： frontend + backend(s)
- 一个来自 Client 端的 request，HAProxy 要维持两个 TCP 连接：
  - Client -> HAProy
  - HAProxy -> Real Server
- 支持 HTTP KeepAlive 和 Pipeline
- IPv4/IPv6 Dual stack

连接模式（2）

HAProxy 在处理 frontend 和 backend 的 HTTP transaction 时，共有以下五种模式：

```
By default HAProxy operates in keep-alive mode with regards to persistent
connections: for each connection it processes each request and response, and
leaves the connection idle on both sides between the end of a response and the
start of a new request.

HAProxy supports 5 connection modes :
  - keep alive    : all requests and responses are processed (default)
  - tunnel        : only the first request and response are processed,
                    everything else is forwarded with no analysis.
  - passive close : tunnel with "Connection: close" added in both directions.
  - server close  : the server-facing connection is closed after the response.
  - forced close  : the connection is actively closed after end of response.
```

功能列表

- 负载均衡算法
- Persistence
- Content Switching
- Content Rewriting
- HTTP Compression：支持 HTTP 压缩，gzip&deflate
- HTTP Basic Authentication: 简单认证
- SSL Offload： 配置 cert，支持 TCPS/HTTPS
- Transparent Proxy
- Health Check: 对 real server 做健康检查
- Log

功能列表 – 负载均衡算法

- 基本算法
  - roundrobin：动态 rr 算法，支持动态修改 rs 的 weight
  - static-rr: 静态 rr 算法，参考 roundrobin
  - leastconn: 最少连接数
  - first: 优先使用 server id 最小的，超过 maxconn 时选择下一个 server，适合非 HTTP 的长连接。可结合 cloud 使用。
- Hash 类算法
  - source：源 IP
  - uri:
  - url_param: URL 中某个指定参数的 value hash
  - hdr(<name>): 指定任何一个 header name，以其 value hash
  - ……

功能列表 – Persistence

- source ip:
- appsession:
- cookie: insert/rewrite/prefix…
- SSL session ID
- …

功能列表 – Content Switching

- 前面提到，一个 Proxy 由 frontend + backend(s) 组成
- 其中 frontend 一个，backend 可以有多个。通过配置，实现不同的请求交给不同的 backend 处理，即所谓的 Content Switching
- 核心配置：

  use_backend <backend> [{if | unless} <condition>]

功能列表 – Content Rewriting

- 支持修改 HTTP Request 以及 Response Header
- 配置项 http-request/http-response

```
add-header <name> <fmt> | set-header <name> <fmt> |
del-header <name> | set-nice <nice> | set-log-level <level> |
replace-header <name> <match-regex> <replace-fmt> |
replace-value <name> <match-regex> <replace-fmt> |
```

实例

- 典型配置
- Hot Configuration
- Mutli-Process
- HA Synchronization
- Statistics

实例 -- 典型配置

```
global
    node hap
    pidfile /var/log/haproxy/hap.pid
    stats socket /var/log/haproxy/hap.socket level admin
    maxconn 4096
    daemon
    quiet

defaults
    mode http
    option splice-auto
    option http-keep-alive

    timeout client 50s
    timeout server 50s
    timeout connect 5s
    timeout http-keep-alive 50s
    timeout http-request 50s

frontend fe
    bind :80
    use_backend be unless

backend be
    balance roundrobin
    server 1 2.2.2.1:80 id 1 cookie rs1 weight 1 maxconn 0
    server 2 2.2.2.2:80 id 2 cookie rs2 weight 1 maxconn 0
~
```

实例 -- Hot Configuration

- Unix/TCP Socket Command
- 配置实例：

```
global
    stats socket /var/run/haproxy.sock mode 600 level admin
    stats socket ipv4@192.168.0.1:9999 level admin
    stats timeout 2m
```

- 设置命令示例

$ echo “show stat” | socat stdio unix-connect:/path/to/hap.socket

- 支持 Command: 查看info、sess，修改 server 配置，设置 maxconn，stick table 等等
- REF: http://cbonte.github.io/haproxy-dconv/configuration-1.5.html#9.2

实例 -- Mutli-Process

- 利用多进程获取高性能
- nbproc: 指定启动的多进程个数
- bind-process: 当前 fe/be 在哪些进程上启动
- process: bind option, 当前 bind 在那些进程启动
- 配置实例（**避免多进程下 epoll 惊群**）

```
1
2 global
3       nbproc 4
4
5 frontend fe
6       bind 1.1.1.1:80 process 1
7       bind 1.1.1.1:80 process 2
8       bind 1.1.1.1:80 process 3
9       bind 1.1.1.1:80 process 4
```

实例 -- HA Synchronization

- HA 环境下同步 stick table
- 配置 peers section

```
peers mypeers
        peer local 1.1.1.1:10000
        peer remote 1.1.1.2:10000
```

- 引用定义的 peers

```
stick-table type ip size 20k peers mypeers
```

- 启动 HAProxy 进程

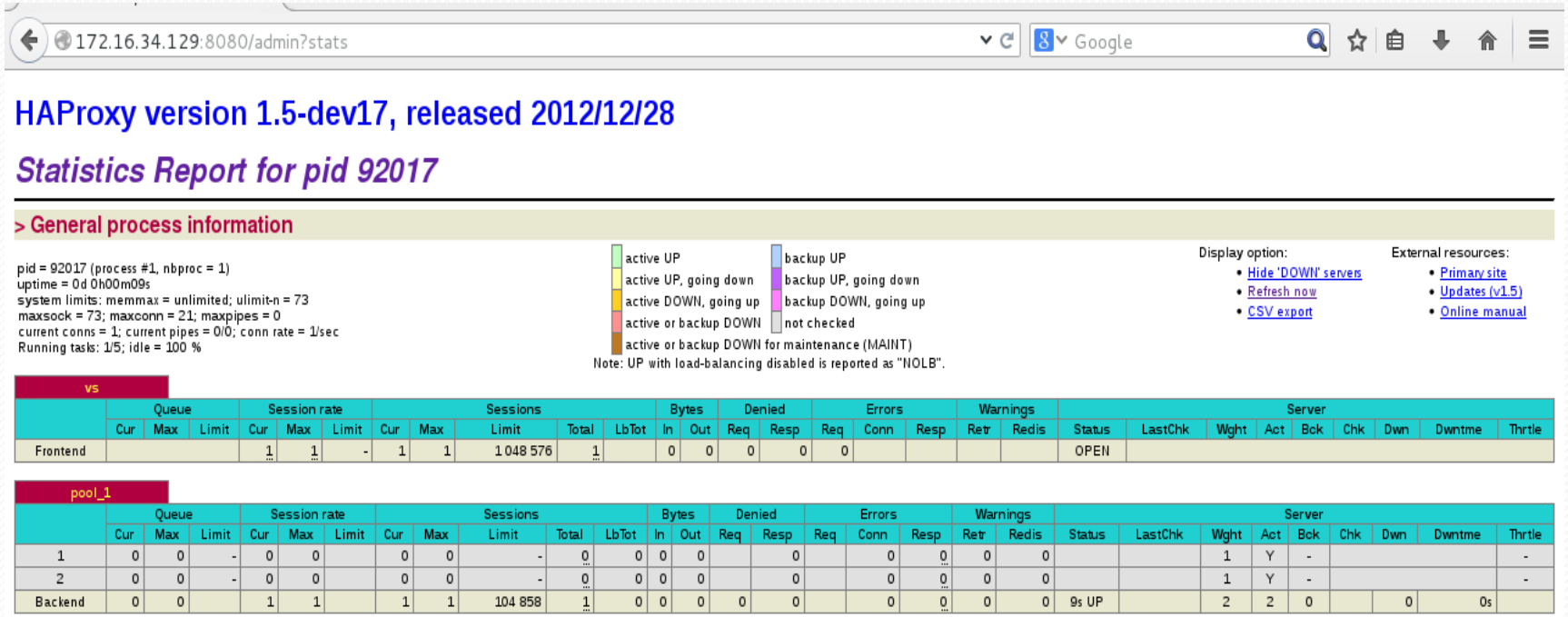
本地： $ haproxy -f h.cfg -L local

远程： $ haproxy -f h.cfg -L remote

实例 -- Statistics

- HAProxy 自带的统计信息 Web 展示，关键配置如下

```
backend be
        stats enable
        stats uri /admin?stats
```



Debug&Analysis

- 编译带调试信息版本的 HAProxy
  - Makefile 中添加编译选 -DDEBUG_XXX
  - 启动时 -d option
- 抓包判断出问题的环节
  - 分别在 HAProxy 端和 Real Server 端抓包，并对比判断
  - 条件允许的话，client 端抓包
- 通过 Unix Socket Command，执行 show stat
  - 主要观察 frontend 和 backend 上的相关统计数据

Reference

- HAProxy 官网： https://www.haproxy.org/
- HAProxy version 1.5 配置手册：
  http://cbonte.github.io/haproxy-dconv/configuration-1.5.html

Q&A