Linux 性能優化實踐

36 - 套路篇: 怎麼評估系統的網路效能

性能指標回顧

- 頻寬 (帶寬)
  - 链路的最大传输速率，单位是 b/s（比特 / 秒）
  - 带宽跟物理网卡配置是直接关联的
- 吞吐量（Throughput）
  - 没有丢包时的最大数据传输速率，单位通常为 b/s （比特 / 秒）或者 B/s（字节 / 秒）
- 延时 (Latency)
  - 从网络请求发出后，一直到收到远端响应，所需要的时间延迟
- Packet Per Second (PPS)
  - 以网络包为单位的传输速率

网络基准测试 - 開始之前

- 测试之前，先弄清楚，你要评估的网络性能/应用程序，究竟属于协议栈的哪一层?
- 範例
  - 基于 HTTP 或者 HTTPS 的 Web 应用程序
    - HTTP/HTTPS 效能
  - 游戏服务器，支援大量的同時在線人數
    - TCP/UDP 的性能
  - 把 Linux 作为一个软交换机或者路由器
    - 网络包的处理能力（即 PPS）

各协议层的性能测试

轉發功能

- 网络接口层和网络层
  - 主要负责网络包的封装、寻址、路由以及发送和接收
- Packet Per Second（包 / 秒）
- Tools
  - Hping3
  - pktgen

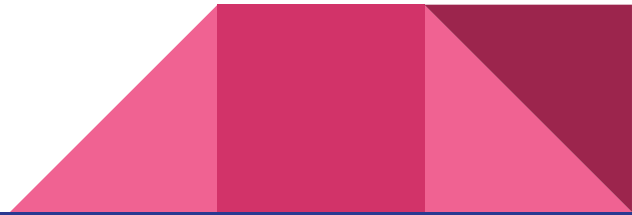
hping3

-S参数表示设置TCP协议的SYN（同步序列号），-p表示目的端口为80

-i u100表示每隔100微秒发送一个网络帧

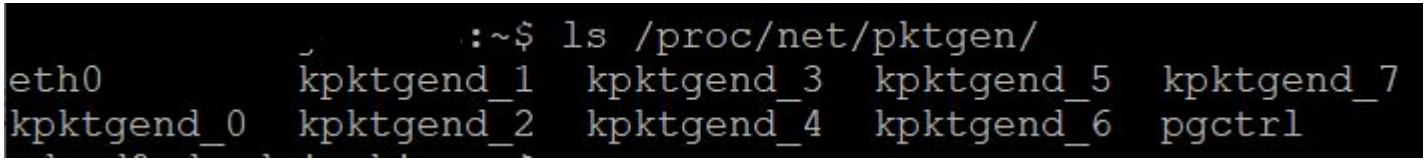
注：如果你在实践过程中现象不明显，可以尝试把100调小，比如调成10甚至1

$ hping3 -S -p 80 -i u100 192.168.0.30



pktgen

- 加载 pktgen 内核模块
  - $ modprobe pktgen
-

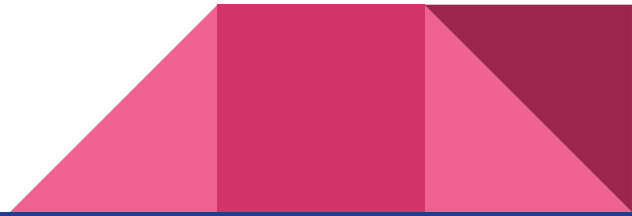


pktgen Example (1 / 4)

```
# 定义一个工具函数，方便后面配置各种测试选项
function pgset() {
    local result
    echo $1 > $PGDEV
    result=`cat $PGDEV | fgrep "Result: OK:"`
    if [ "$result" = "" ]; then
        cat $PGDEV | fgrep Result:
    fi
}
```

pktgen Example（2 / 4)

```
# 为0号线程绑定eth0网卡

PGDEV=/proc/net/pktgen/kpktgend_0

pgset "rem_device_all"  # 清空网卡绑定

pgset "add_device eth0"  # 添加eth0网卡
```



pktgen Example（3 / 4)

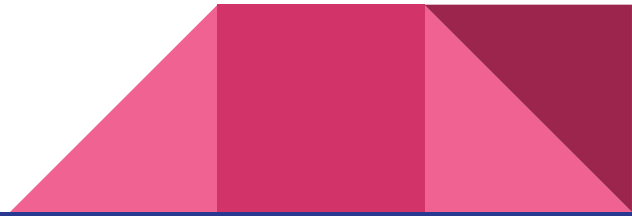
```
# 配置eth0网卡的测试选项

PGDEV=/proc/net/pktgen/eth0

pgset "count 1000000"    # 总发包数量

pgset "delay 5000"       # 不同包之间的发送延迟(单位纳秒)

pgset "clone_skb 0"      # SKB包复制

pgset "pkt_size 64"      # 网络包大小

pgset "dst 192.168.0.30" # 目的IP

pgset "dst_mac 11:11:11:11:11:11"  # 目的MAC
```

pktgen Example（4 / 4)

启动测试

PGDEV=/proc/net/pktgen/pgctrl

pgset "start"



pktgen Test Result (1 / 2)

$ cat /proc/net/pktgen/eth0

- Params:
- Current:
- Result:

```
Params: count 1000000  min_pkt_size: 64  max_pkt_size: 64
     frags: 0  delay: 5000  clone_skb: 0  ifname: eth0
     flows: 0 flowlen: 0
     queue_map_min: 0  queue_map_max: 0
     dst_min: 10.16.33  dst_max:
     src_min:   src_max:
     src_mac: c0:3f:d5:51:39:79 dst_mac: 00:02:d1:3c:6e:c0
     udp_src_min: 9  udp_src_max: 9  udp_dst_min: 9  udp_dst_max: 9
     src_mac_count: 0  dst_mac_count: 0
     Flags:
```

pktgen Test Result (2/2)

```
Current:
     pkts-sofar: 1000000  errors: 0
     started: 126290717360083us  stopped: 126290767361127us idle: 4667704us
     seq_num: 1000001  cur_dst_mac_offset: 0  cur_src_mac_offset: 0
     cur_saddr: 10.16.106.140  cur_daddr: 10.16.33.0
     cur_udp_dst: 9  cur_udp_src: 9
     cur_queue_map: 0
     flows: 0
Result: OK: 5000043(c332339+d4667704) usec, 1000000 (64byte,0frags)
  199998pps 102Mb/sec (102398976bps) errors: 0
```

TCP/UDP 性能

- 傳輸層
- Tools
  - iperf
  - netperf

iperf Example (1 / 2)

- 在目标机器上启动 iperf 服务端
  - # -s表示启动服务端，-i表示汇报间隔，-p表示监听端口
  - $ iperf3 -s -i 1 -p 10000

- 在另一台机器上运行 iperf 客户端，运行测试
  - # -c表示启动客户端，192.168.0.30为目标服务器的IP
  - # -b表示目标带宽(单位是bits/s)
  - # -t表示测试时间
  - # -P表示并发数，-p表示目标服务器监听端口
  - $ iperf3 -c 192.168.0.30 -b 1G -t 15 -P 2 -p 10000

iperf Example (2 / 2)

```
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-15.00  sec   103 MBytes  57.7 Mbits/sec                  sender
[  4]   0.00-15.00  sec   103 MBytes  57.7 Mbits/sec                  receiver
[  6]   0.00-15.00  sec   108 MBytes  60.3 Mbits/sec                  sender
[  6]   0.00-15.00  sec   108 MBytes  60.3 Mbits/sec                  receiver
[SUM]   0.00-15.00  sec   211 MBytes   118 Mbits/sec                  sender
[SUM]   0.00-15.00  sec   211 MBytes   118 Mbits/sec                  receiver
```

```
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-15.02  sec  0.00 Bytes  0.00 bits/sec                  sender
[  5]   0.00-15.02  sec   103 MBytes  57.7 Mbits/sec                  receiver
[  7]   0.00-15.02  sec  0.00 Bytes  0.00 bits/sec                  sender
[  7]   0.00-15.02  sec   108 MBytes  60.2 Mbits/sec                  receiver
[SUM]   0.00-15.02  sec  0.00 Bytes  0.00 bits/sec                  sender
[SUM]   0.00-15.02  sec   211 MBytes   118 Mbits/sec                  receiver
```

HTTP 性能

- 應用層
- Tools
  - ab
  - webbench

ab Example

-c表示并发请求数为1000,

-n表示总的请求数为10000

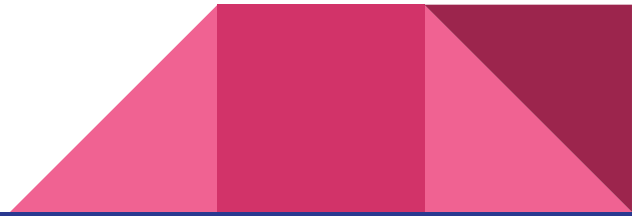
$ ab -c 1000 -n 10000 http://192.168.0.30/

- Results
  - Request
  - Connection Times
  - Processing percentage

```
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:        [redacted]HTTP-Server
Server Hostname:        10.16.105.21
Server Port:            80

Document Path:          /
Document Length:        1558 bytes

Concurrency Level:      1000
Time taken for tests:   3.866 seconds
Complete requests:      10000
Failed requests:        288
   (Connect: 0, Receive: 0, Length: 288, Exceptions: 0)
Total transferred:      18113187 bytes
HTML transferred:       15131296 bytes
Requests per second:    2586.89 [#/sec] (mean)
Time per request:       386.565 [ms] (mean)
Time per request:       0.387 [ms] (mean, across all concurrent requests)
Transfer rate:          4575.86 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    6  74.0      0    1004
Processing:    23  185 662.0     38    3857
Waiting:        0   99 443.6     38    3857
Total:         24  191 669.7     38    3864

Percentage of the requests served within a certain time (ms)
  50%     38
  66%     39
  75%     40
  80%     41
  90%     47
  95%   1036
  98%   3195
  99%   3413
 100%   3864 (longest request)
```

應用負載性能

- 要求性能工具本身可以模拟用户的请求负载
- Tools
  - wrk
  - TCPCopy
  - JMeter
  - LoadRunner

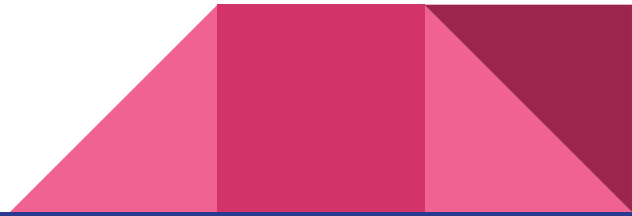


wrk Example

```
~/temp/wrk$ ./wrk -c 1000 -t 2 http://10.16.105.21/
Running 10s test @ http://10.16.105.21/
  2 threads and 1000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    50.60ms  211.45ms   2.00s    94.46%
    Req/Sec     4.03k     1.70k    7.87k    64.47%
  79125 requests in 10.02s, 141.11MB read
  Socket errors: connect 0, read 29, write 0, timeout 263
Requests/sec:   7900.22
Transfer/sec:     14.09MB
```

小結

- 低层协议是高层协议的基础
- 从上到下，对每个协议层进行性能测试，然后根据性能测试的结果，结合 Linux 网络协议栈的原理，找出导致性能瓶颈的根源，进而优化网络性能。



思考

- 你是如何评估网络性能的？
- 在评估网络性能时，你会从哪个协议层、选择哪些指标，作为性能测试最核心的目标？
- 你又会用哪些工具，测试并分析网络的性能呢？