此处不方便透露主机名及IP地址，所涉及所有截图中的IP及主机名将被抹去

一：测试环境

1. 软件

压力测试工具dnsperf

测试命令：dnsperf -d domain.list -s IP -l300

测试域名在domain.list中配置



测试目标sample/dns\_server



参数说明:

-d: 指定网卡设备

-i：指定dns的服务IP

-p：指定dns的服务端口

-g：指定加载的ebpf的字节码成勋

-Q：指定网卡队列，同时worker数量和网卡队列数相同

1. 硬件

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | CPU | 核数 | 内存 | 网卡 | 角色 | 运行程序 |
| 机器A | Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz | 24 | 64G | Intel Corporation 82599EB 10-Gigabit SFP+ | 被打压机器 | sample/dns\_server |
| 机器B | Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz | 24 | 64G | Intel Corporation 82599EB 10-Gigabit SFP+ | 打压机器 | dnsperf |
| 机器C | Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz | 24 | 64G | Intel Corporation 82599EB 10-Gigabit SFP+ | 打压机器 | dnsperf |
| 机器D | Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz | 24 | 64G | Intel Corporation 82599EB 10-Gigabit SFP+ | 打压机器 | dnsperf |

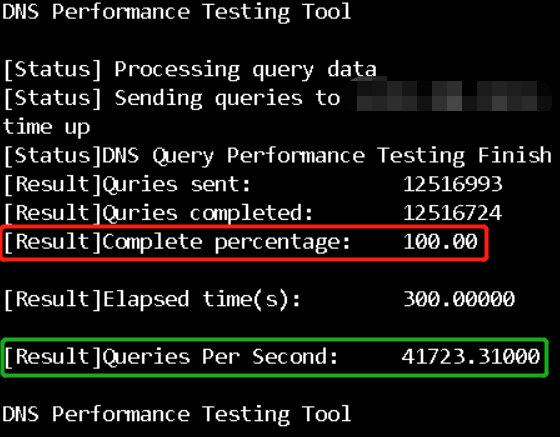
1. sample/dns\_server

此测试程序只是做了简单的dns解包和封包处理。目的是为了接近收发包计算量，为了简单没有单独开发发包工具，用了dnsperf当做打压工具，所以写了个简单的dns\_server。同时因为收报为轮询式的，CPU会直接占满，为了可以观察CPU的利用率所以在收到包为0后进行了nanosleep 1000ns。

xdp\_packet的工作线程命名方式为”xdp\_ wrk\_” + worke\_id

三、测试方法

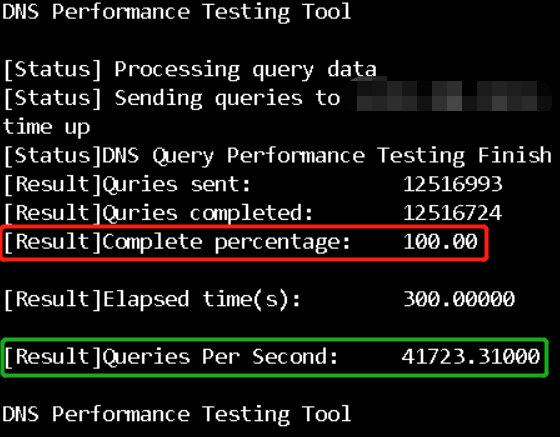
１、在打压机器上逐渐增加dnsperf的进程数，观察所有dnsperf的输出结果如果，如果Complete percentage不是100.00则代表压力超出承受范围，需要通过减少dnsperf进程数的方式来减少压力



2、统计每个dnsperf的输出结果，以单个进程输出结果为例

红框中“Complete percentage”必须为100代表所有请求都得到了正确的响应。

绿框中为qps，将多个同时运行的dnsperf的输出结果中的”Queries Per Second”加在一起就是测试结果的qps



1. 单个打压机器所能产生的压力是有限的，所以需要增加打压机器，并且需要保证所有的机器的输出结果”Complete percetnage”为100.00时达到最大每秒发包数，将同时运行的所有的打压机器上的所有的dnsperf的输出结果中的”Queries Per Second”加在一起即为被打压机器的处理能力

三、测试过程

1. 单核单网卡队列压力测试，一个被打压机器

机器A：



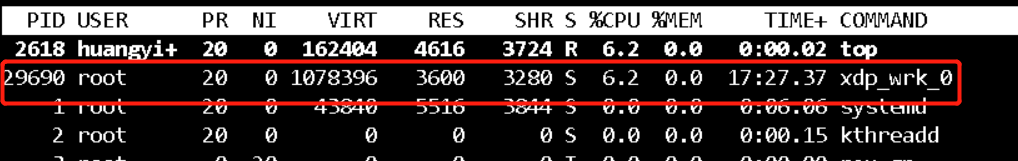
-i 是测试网卡eth2的IP地址

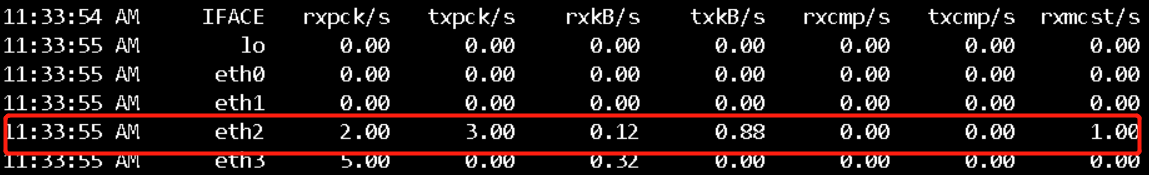
2、三个打压机器分别是机器B、机器C、机器D,每个机器上同时运行9个dnsperf进程

通过反复测试，如果在增加进程数”Complete percentage”将低于100.00

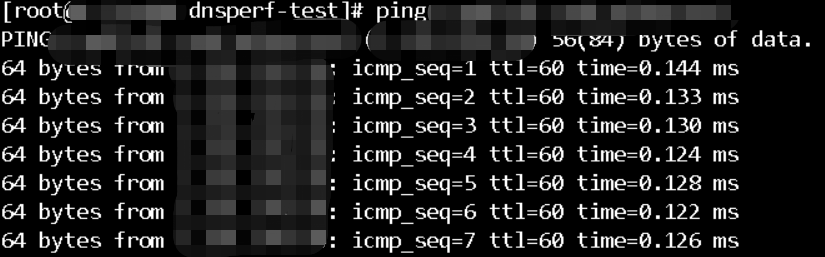
3、记录测试之前，各机器状态

* 机器A：CPU及网卡情况

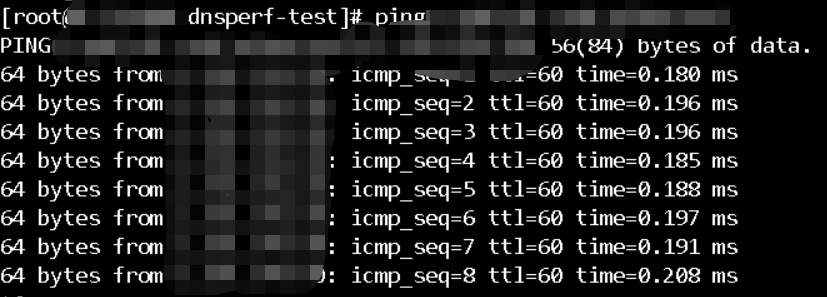




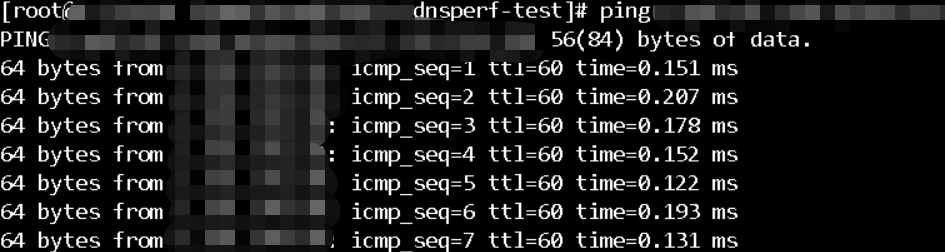
* 机器B到机器A的延时



* 机器C到机器A的延时



* 机器D到机器A的延时



1. 启动测试

先在机器A上启动dns\_server

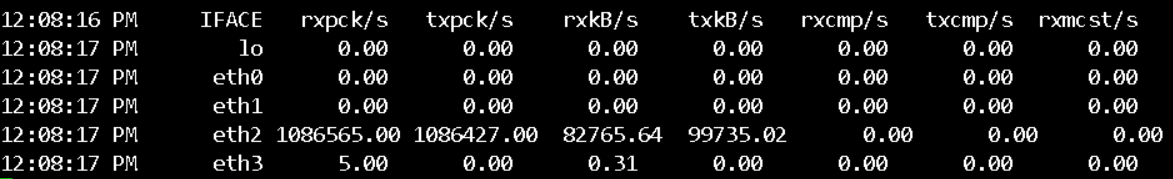
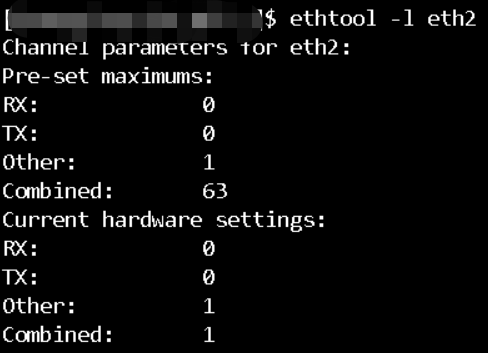
然后同时在机器B、C、D、上分别启动9个dnsperf并将结果保存在dnsperf.log中，脚本如下

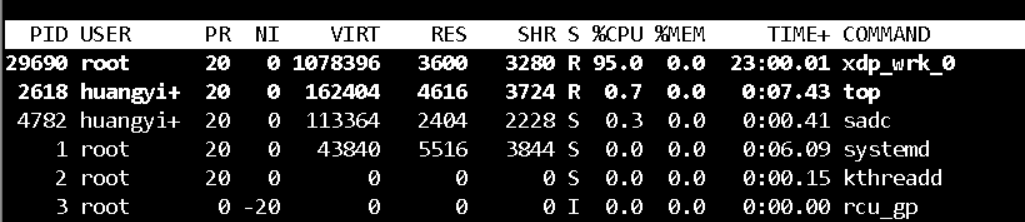


测试域名如下



1. 观察测试过程中机器A的状态

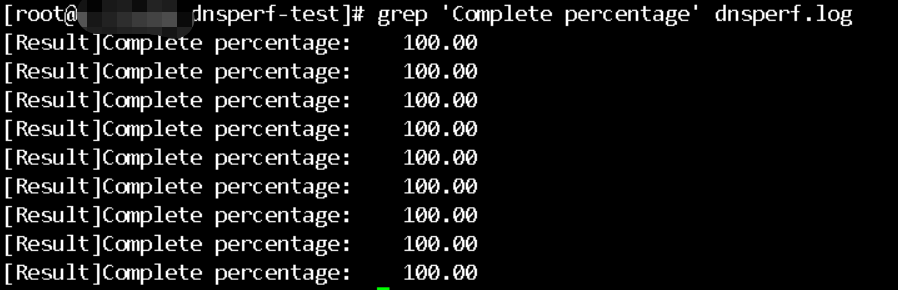




1. 统计dnsperf的输出结果并计算总QPS

* 首先确定“Complete percentage”

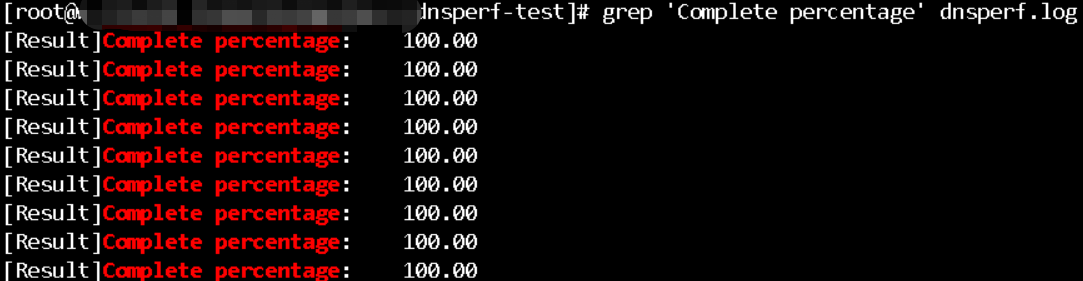
机器B



机器C



机器D



所有机器上的所有9个dnsperf都是100.00

* 然后计算总QPS

机器B



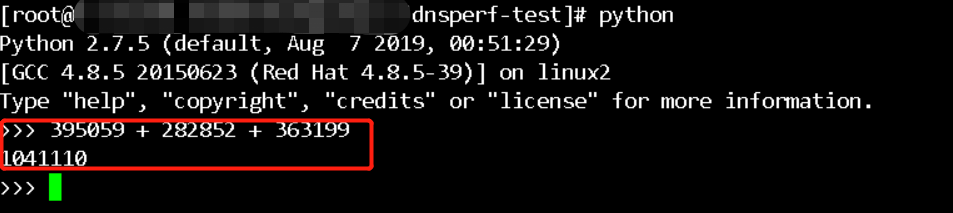
机器C



机器D



把三个机器的QPS加在一起即为总QPS



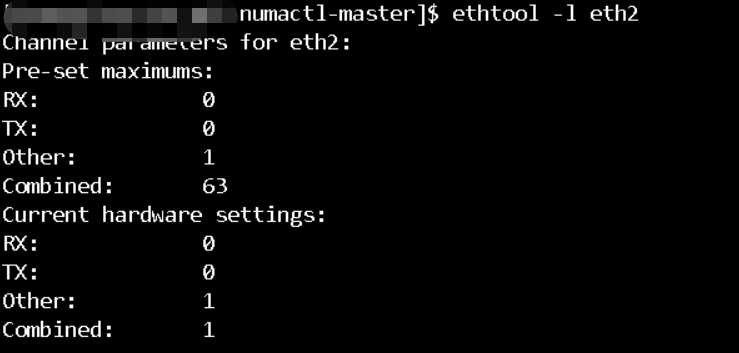
**总结:**

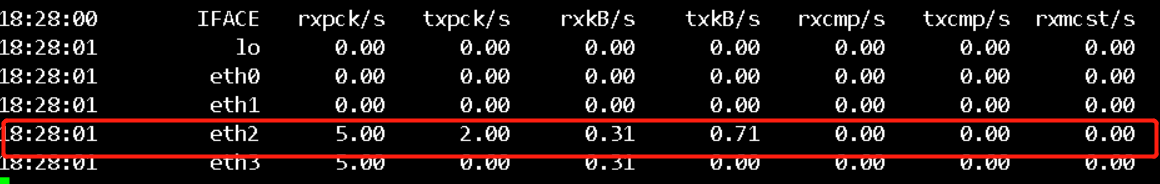


**此测试网卡与xdp\_wrk\_0属于同一numa节点，且在单核单网卡队列的情况下， sample/dns\_server的CPU使用率95%左右，最大可承载1041110QPS即100万+QPS，但在接下来的过程中发现了bug,这个bug会导致在分配内存时设置numa策略失败，但程序可以正常执行，numa策略设置失败时，使用的仍然是本地优先分配的策略**

相同测试环境及相同机器配置下的修正numa设置策略失败问题再次测试如下：

1. 打压力之前机器A的状态

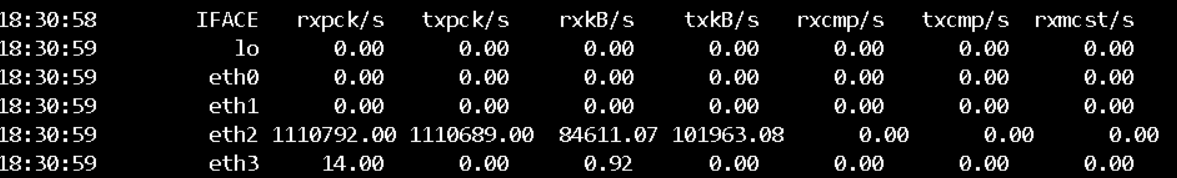






1. 在机器B、C、D启动dnsperf, 同时分别启动了10个dnsperf,
2. 测试过程中机器A的状态

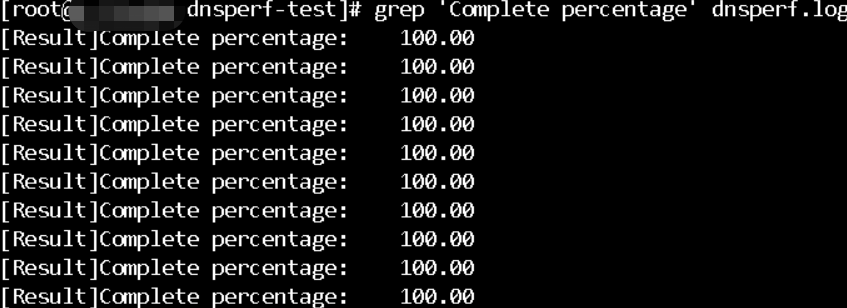




1. 测试结束统计dnsperf的输出结果并统计QPS

* 首先检查”Complete percentage”

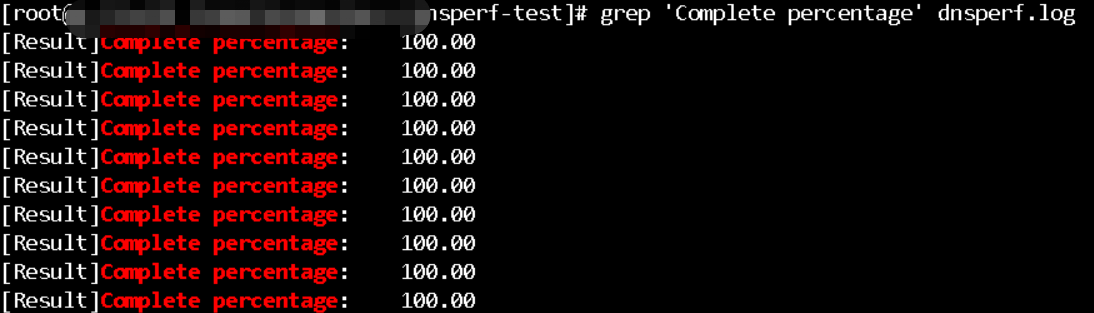
机器B



机器C



机器D



所有打压力的机器上的”Complete percentage”都是100.00

* 统计QPS

机器B



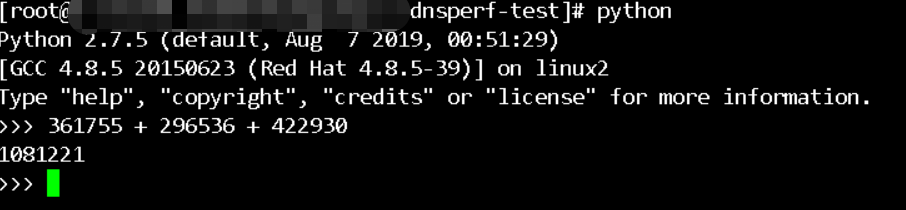
机器C



机器D



汇总:



**总结：**

**在修复了numa策略设置失败的问题之后且同样是单核单网卡队列的配置下QPS是1081221,比修复前的1041110提升了3万，提升并不明显，应该是因为修复前修复后用的numa节点及CPU核与网卡的numa节点都一致造成的。**