利用 eBPF sockmap/redirection 提升 socket 性能

环境信息

机器：Ubuntu 20.04.6 LTS

操作系统版本：5.8.0-050800-generic

软件源

更新操作系统的 /etc/apt/sources.list 软件源，如下所示：

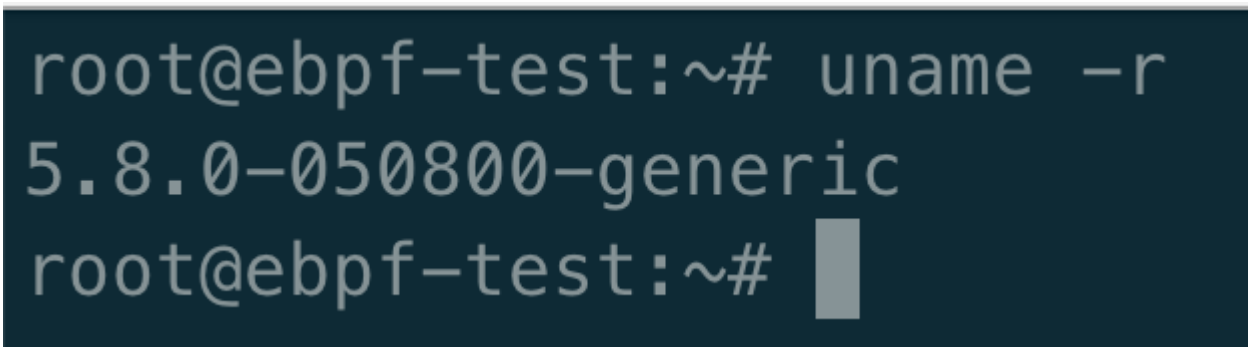
```
root@ebpf-test:~# cat /etc/apt/sources.list
deb http://mirrors.aliyun.com/ubuntu/ focal main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted universe multiverse
```

升级内核(5.8.0)

如果操作系统版本较低，没有满足 5.8.0+ 内核，可以使用以下方式安装 5.8.0 内核。

```
wget https://raw.githubusercontent.com/pimlie/ubuntu-mainline-kernel.sh/master/ubuntu-mainline-kernel.sh
chmod +x ubuntu-mainline-kernel.sh
bash ubuntu-mainline-kernel.sh --no-checksum  -i 5.8.0
```

升级完内核后，重启操作系统即可生效。



eBPF 工具

安装 eBPF 工具。具体操作如下所示：

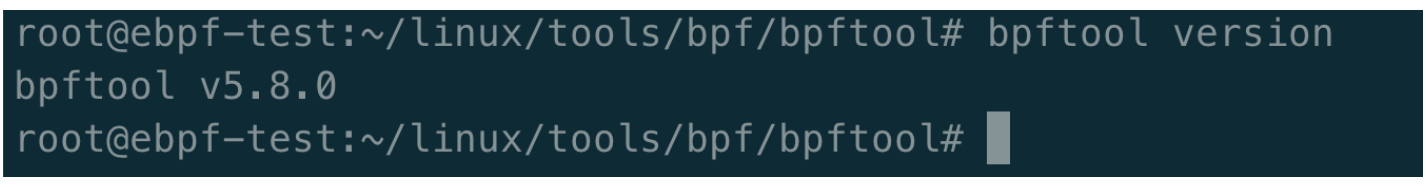
```
sudo apt-get update -y && sudo apt-get upgrade -y

sudo apt-get install -y git cmake make gcc python3 libncurses-dev gawk flex bison openssl libssl-dev dkms libelf-dev libudev-dev libpci-dev libiberty-dev autoconf

sudo apt-get install -y binutils-dev clang gcc-multilib
```

安装 bpftool 5.8 版本

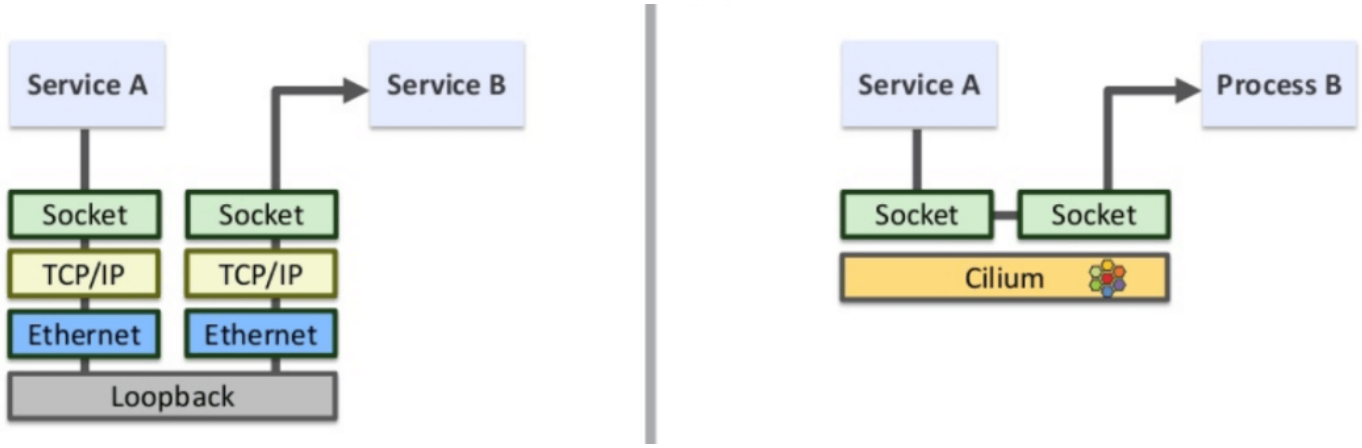
```
git clone -b v5.8 https://github.com/torvalds/linux.git --depth 1
cd /root/linux/tools/bpf/bpftool
make && make install
```



实现原理

对于源端和目的端都在同一台机器的应用来说，可以使用 BPF 程序做 socket level 重定向（redirection），

进而绕过整个 TCP/IP 协议栈，直接将数据发送到 socket 对端，提升 socket 性能。

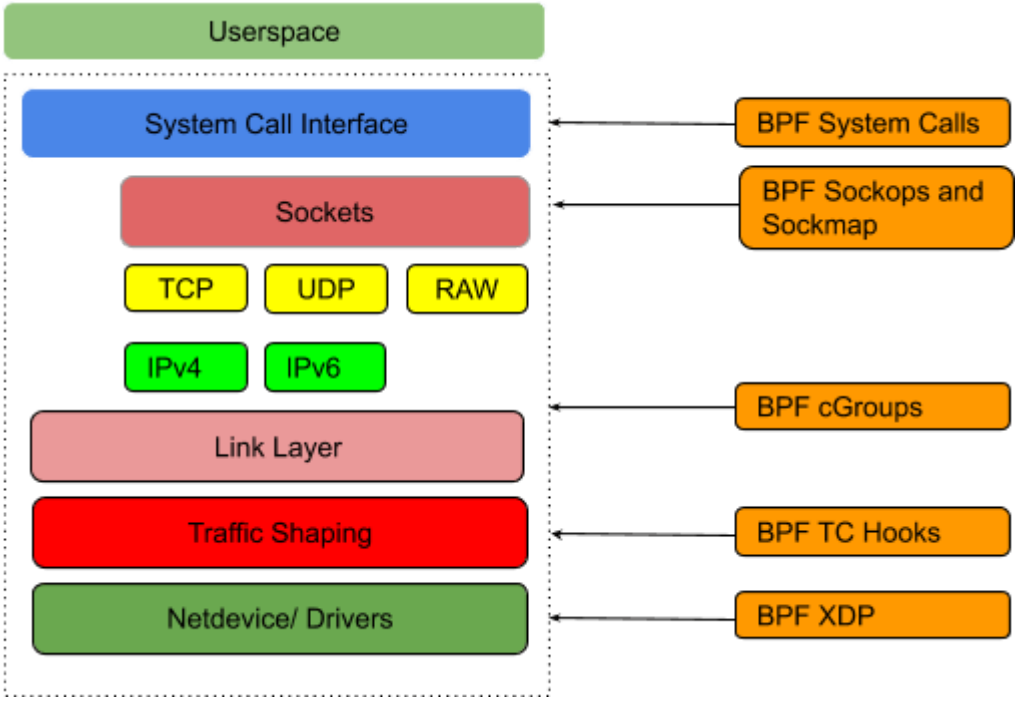


核心概念：

sockmap：这是一个存储 socket 信息的映射表。

a. 一段 BPF 程序**监听所有的内核 socket 事件**，并将新建的 socket 记录到这个 map；
b. 另一段 BPF 程序**拦截所有 `sendmsg` 系统调用**，然后去 map 里查找 socket 对端，之后调用 BPF 函数绕过 TCP/IP 协议栈，直接将数据发送到对端的 socket queue。

cgroups：指定要**监听哪个范围内的 sockets 事件**，进而决定了稍后要对哪些 socket 做重定向。sockmap 需要关联到某个 cgroup，然后这个 cgroup 内的所有 socket 就都会执行加载的 BPF 程序。



本文将主要关注下面两种能拦截到 socket 操作（例如 TCP `connect`、`sendmsg` 等）的类型：

- `BPF_PROG_TYPE_SOCK_OPS`：socket operations 事件触发执行。
- `BPF_PROG_TYPE_SK_MSG`：`sendmsg()` 系统调用触发执行。

创建一个全局的**映射表**（map）来**记录所有的 socket 信息**。基于这个 sockmap，编写两段 BPF 程序分别完成以下功能：

- 程序一：拦截所有 TCP connection 事件，然后将 socket 信息存储到这个 map；
- 程序二：拦截所有 `sendmsg()` 系统调用，然后从 map 中查 询这个socket 信息，之后直接将数据**重定向到对端**。

测试

下载源码，编译、加载、运行。

```
git clone https://github.com/ArthurChiao/socket-acceleration-with-ebpf
cd bpf
bash load.sh
```

load.sh 具体操作如下所述：

load.sh

```
#!/bin/bash

set -x
set -e

# 挂载 bpf 文件系统
sudo mount -t bpf bpf /sys/fs/bpf/

# 编译 bpf_sockops 程序
clang -O2 -g -target bpf -c bpf_sockops.c -o bpf_sockops.o

# 加载并附加 bpf_sockops 程序
sudo bpftool prog load bpf_sockops.o /sys/fs/bpf/bpf_sockops
sudo bpftool cgroup attach /sys/fs/cgroup/unified/ sock_ops pinned
/sys/fs/bpf/bpf_sockops
```

```
15
16 # 提取 bpf_sockops 程序使用的 sockhash map 的 id
17 # 然后这个 map id 被固定到 bpf 虚拟文件系统
18 MAP_ID=$(sudo bpftool prog show pinned /sys/fs/bpf/bpf_sockops | grep -o -E
   'map_ids [0-9]+' | cut -d ' ' -f2-)
19 sudo bpftool map pin id $MAP_ID /sys/fs/bpf/sock_ops_map
20
21 # 加载 bpf_redir 程序并将其附加到 sock_ops_map
22 clang -O2 -g -Wall -target bpf -c bpf_redir.c -o bpf_redir.o
23
24 sudo bpftool prog load bpf_redir.o /sys/fs/bpf/bpf_redir map name
   sock_ops_map pinned /sys/fs/bpf/sock_ops_map
25 sudo bpftool prog attach pinned /sys/fs/bpf/bpf_redir msg_verdict pinned
   /sys/fs/bpf/sock_ops_map
```

查看系统中已经加载的所有 BPF 程序：

```
</>                                                    Plain Text | 收起
1 sudo bpftool prog show
```

```
root@ebpf-test:~/socket-acceleration-with-ebpf# sudo bpftool prog show
65: sock_ops  name bpf_sockmap  tag d9aec8c151998c9c  gpl
        loaded_at 2023-04-14T14:28:54+0800  uid 0
        xlated 672B  jited 388B  memlock 4096B  map_ids 6
        btf_id 10
70: sk_msg  name bpf_redir  tag 550f6d3cfcae2157  gpl
        loaded_at 2023-04-14T14:28:54+0800  uid 0
        xlated 224B  jited 156B  memlock 4096B  map_ids 6
        btf_id 14
805: cgroup_skb  tag 6deef7357e7b4530  gpl
        loaded_at 2023-04-14T15:14:57+0800  uid 0
        xlated 64B  jited 66B  memlock 4096B
806: cgroup_skb  tag 6deef7357e7b4530  gpl
        loaded_at 2023-04-14T15:14:57+0800  uid 0
        xlated 64B  jited 66B  memlock 4096B
807: cgroup_skb  tag 6deef7357e7b4530  gpl
        loaded_at 2023-04-14T15:14:57+0800  uid 0
        xlated 64B  jited 66B  memlock 4096B
808: cgroup_skb  tag 6deef7357e7b4530  gpl
        loaded_at 2023-04-14T15:14:57+0800  uid 0
        xlated 64B  jited 66B  memlock 4096B
809: cgroup_skb  tag 6deef7357e7b4530  gpl
        loaded_at 2023-04-14T15:14:57+0800  uid 0
        xlated 64B  jited 66B  memlock 4096B
810: cgroup_skb  tag 6deef7357e7b4530  gpl
        loaded_at 2023-04-14T15:14:57+0800  uid 0
        xlated 64B  jited 66B  memlock 4096B
1935: sock_ops  name bpf_sockmap  tag d9aec8c151998c9c  gpl
        loaded_at 2023-04-14T15:21:32+0800  uid 0
        xlated 672B  jited 388B  memlock 4096B  map_ids 931
        btf_id 1490
1940: sk_msg  name bpf_redir  tag 550f6d3cfcae2157  gpl
```

查看系统中所有的 map，以及 map 详情：

```
root@ebpf-test:~/socket-acceleration-with-ebpf# sudo bpftool map show
6: sockhash  name sock_ops_map  flags 0x0
        key 24B  value 4B  max_entries 65535  memlock 5767168B
931: sockhash  name sock_ops_map  flags 0x0
        key 24B  value 4B  max_entries 65535  memlock 5767168B
root@ebpf-test:~/socket-acceleration-with-ebpf#
```

在一个窗口中启动 `socat` 作为服务端，监听在 1000 端口：

</> Plain Text | 收起

```
# start a TCP listener at port 1000, and echo back the received data
sudo socat TCP4-LISTEN:1000,fork exec:cat
```

另一个窗口用 `nc` 作为客户端来访问服务端，建立 socket：

```
# connect to the local TCP listener at port 1000
nc localhost 1000
```

观察我们在 BPF 代码中打印的日志：

```
sudo cat /sys/kernel/debug/tracing/trace_pipe
```

卸载 bpf 程序

unload.sh 具体操作如下所述：

unload.sh

```
#!/bin/bash
set -x

# 卸载 bpf_redir 程序
sudo bpftool prog detach pinned /sys/fs/bpf/bpf_redir msg_verdict pinned /sys/fs/bpf/sock_ops_map
sudo rm /sys/fs/bpf/bpf_redir

# 卸载 bpf_sockops_v4 程序
sudo bpftool cgroup detach /sys/fs/cgroup/unified/ sock_ops pinned /sys/fs/bpf/bpf_sockops
sudo rm /sys/fs/bpf/bpf_sockops

# 删除 map
sudo rm /sys/fs/bpf/sock_ops_map
```

参考

1. 利用 ebpf sockmap/redirection 提升 socket 性能
2. eBPFsockmap/redirection加速源代码
3. 升级Ubuntu20.04内核参考