基于ebpf的应用性能加速实践

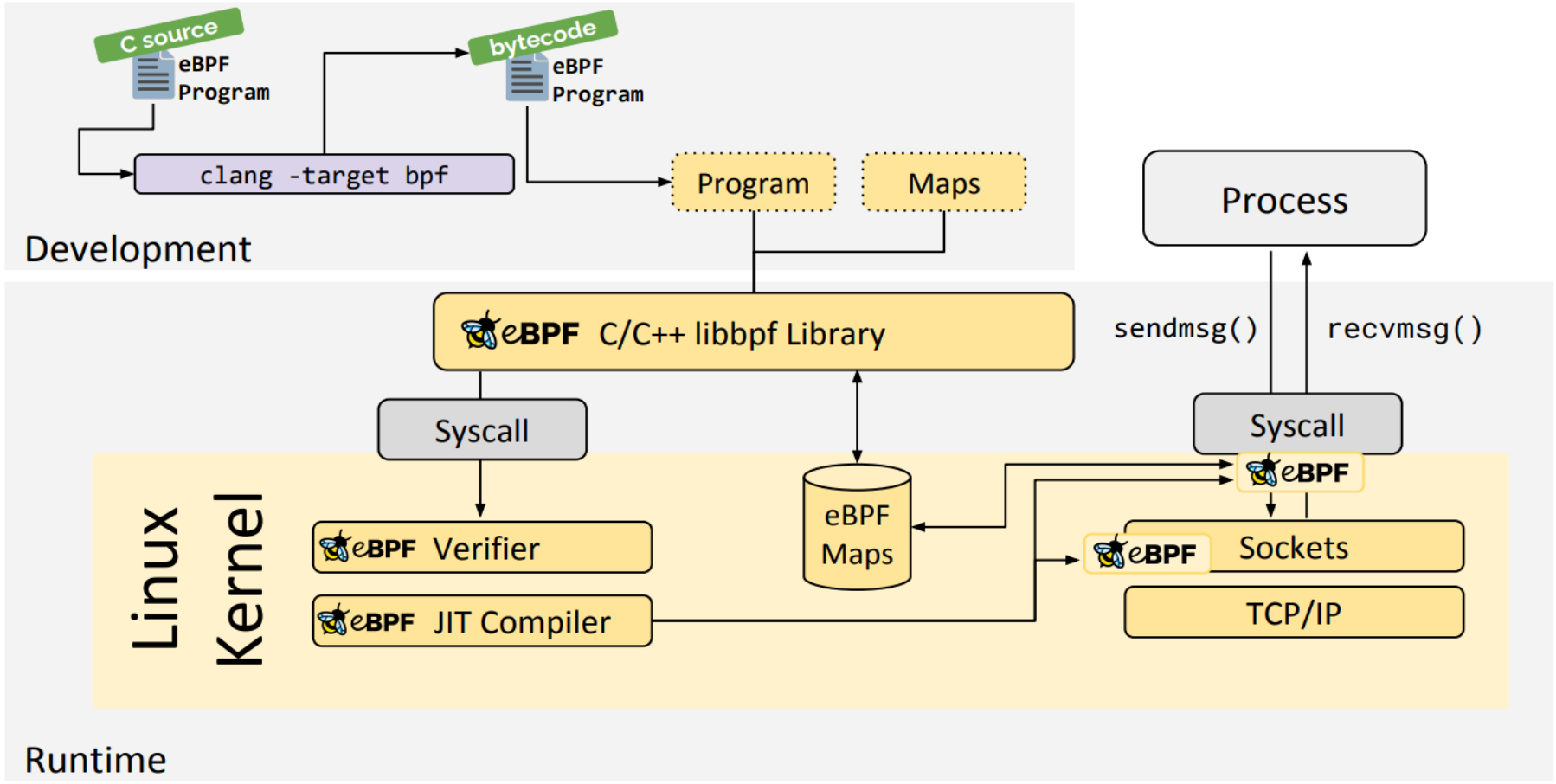
魏勇军 华为 技术专家

何凤清 华为 高级工程师

目录

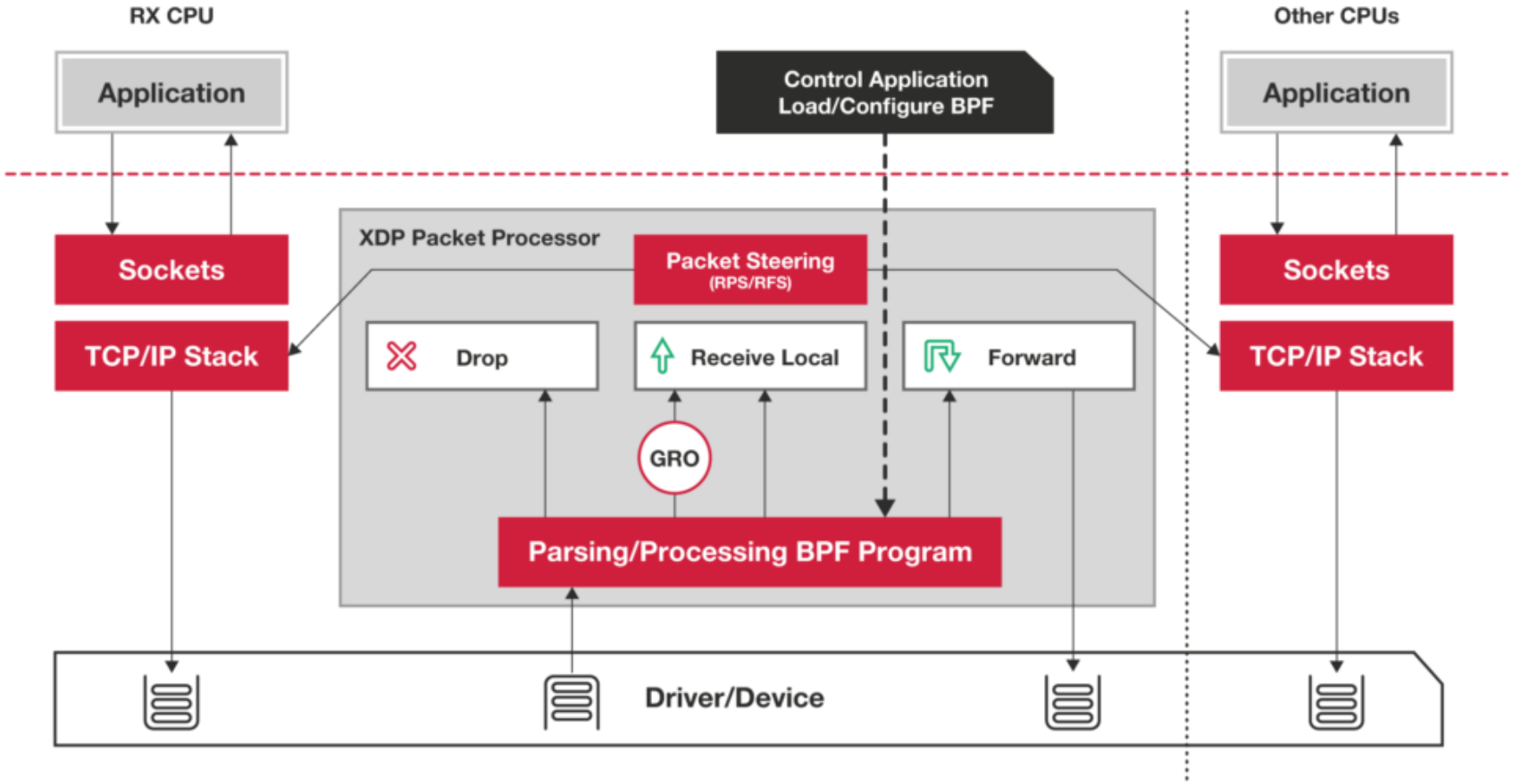
eBPF总体框架

eBPF 是一项革命性技术，它能在内核中运行沙箱程序，来运行用户态编写的代码。基于现有的（而非增加新的）抽象层来打造更加智能、 功能更加丰富的基础设施软件，而不会增加系统的复杂度，也不会牺牲执行效率和安全性。



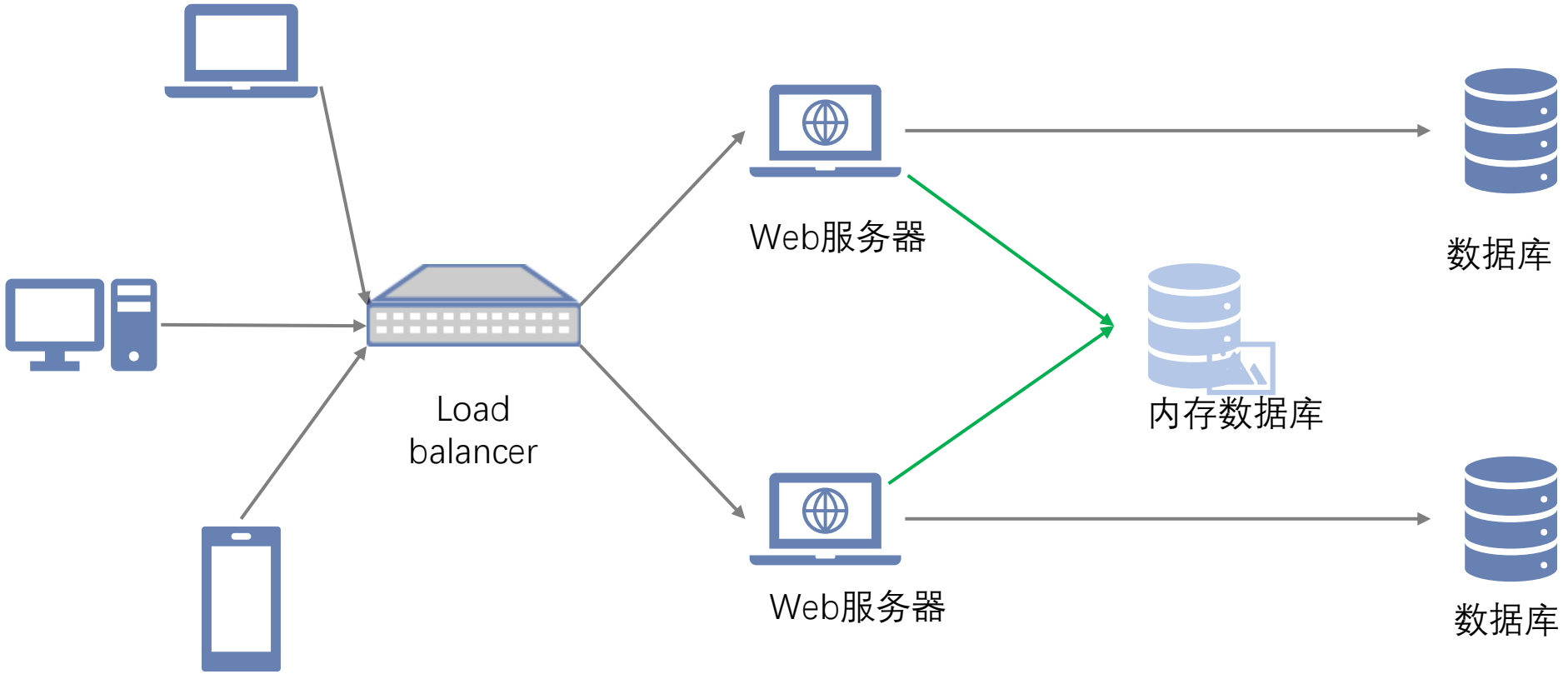
XDP技术框架

XDP（eXpress Data Path）是一个linux内核数据包处理组件，该组件在网卡驱动中直接对收到的报文执行BPF程序，然后根据BPF程序的返回值对报文执行不同的操作。BPF程序执行数据包解析、表查找、创建/管理有状态过滤器、封装/解封数据包等处理。



目录

内存型数据库



数据存在于内存中，提供快速的数据查询服务，通常作为磁盘型数据库的缓存

常见的内存数据库

	Redis	Memcached	LevelDB	GMDB
贡献公司	VMware	LiveJournal	Google	华为
开发语言	C	C++	C++	C++
客户端语言	C, C++, C#, Perl, Ruby, Python, Lua, PHP, Bash等	C, C++, Perl, PHP, Python, Lua, Ruby, C#	没有内置的C/S架构	C
数据类型	string, list, hash, set, 有序set	序列化数据	简单的byte数组	schema自定义数据
进程线程	单线程	多线程	多线程	多线程
持久化	快照/AOF	×	快照	×
压缩	√	×	√	×
复制	√	×	×	√
过期时间	√	√	×	×
集群	√	×	×	√
订阅	√	×	×	√
传输层协议	TCP	Udp(get),TCP(set)		

内核协议栈消耗大量cpu资源

Function	% CPU utilization
native_queued_spin_lock_slowpath	17.68%
__udp_enqueue_schedule_skb	10.95%
clear_page_erms	5.04%
udp4_lib_lookup2	3.23%
_raw_spin_lock	3.04%
fib_table_lookup	2.90%
napi_gro_receive	2.27%
nfp_net_rx	1.97%
i40e_napi_poll	1.32%
udp_queue_rcv_one_skb	1.14%

Table 1: Top ten most CPU-consuming functions on a Memcached server under network load

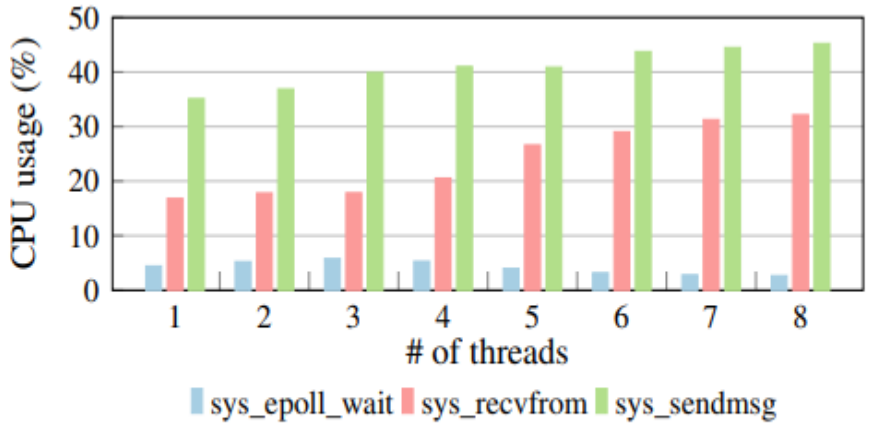
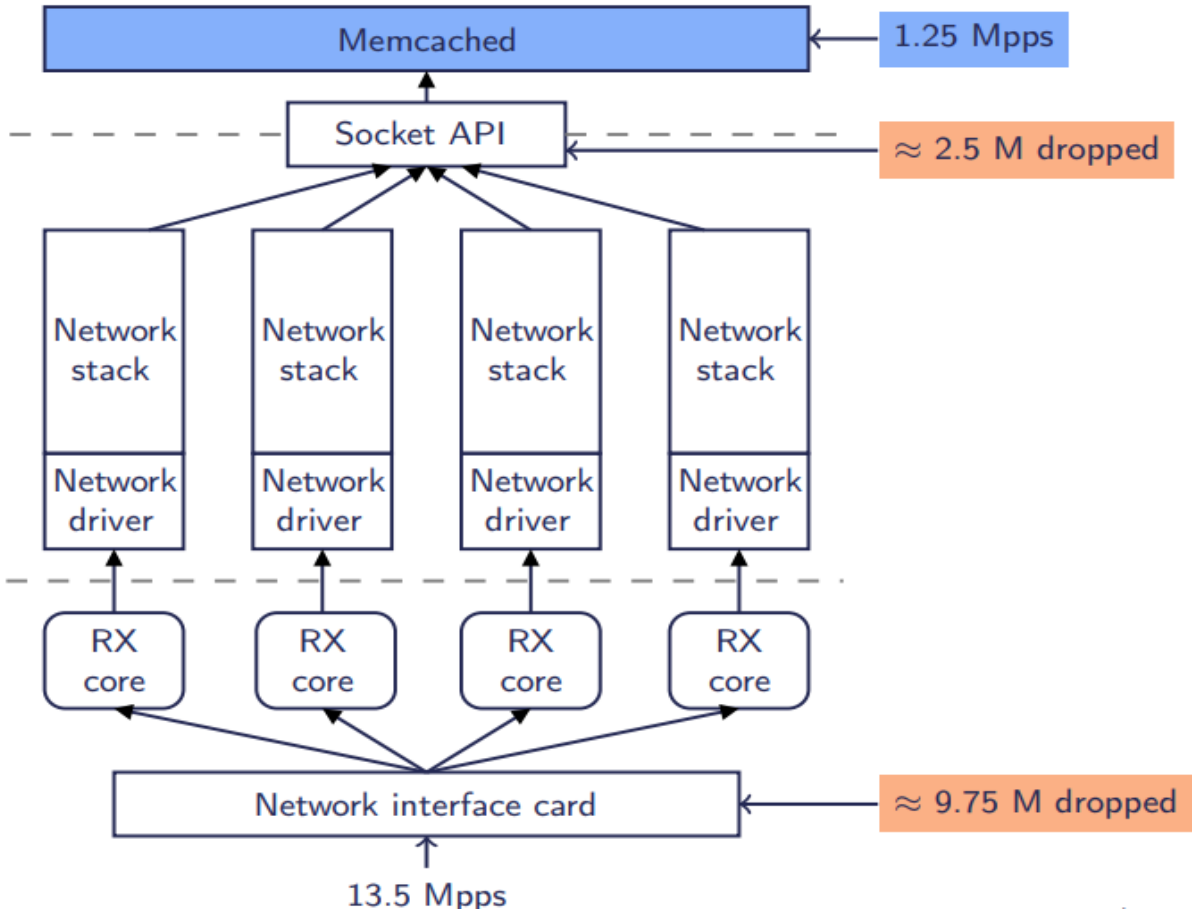


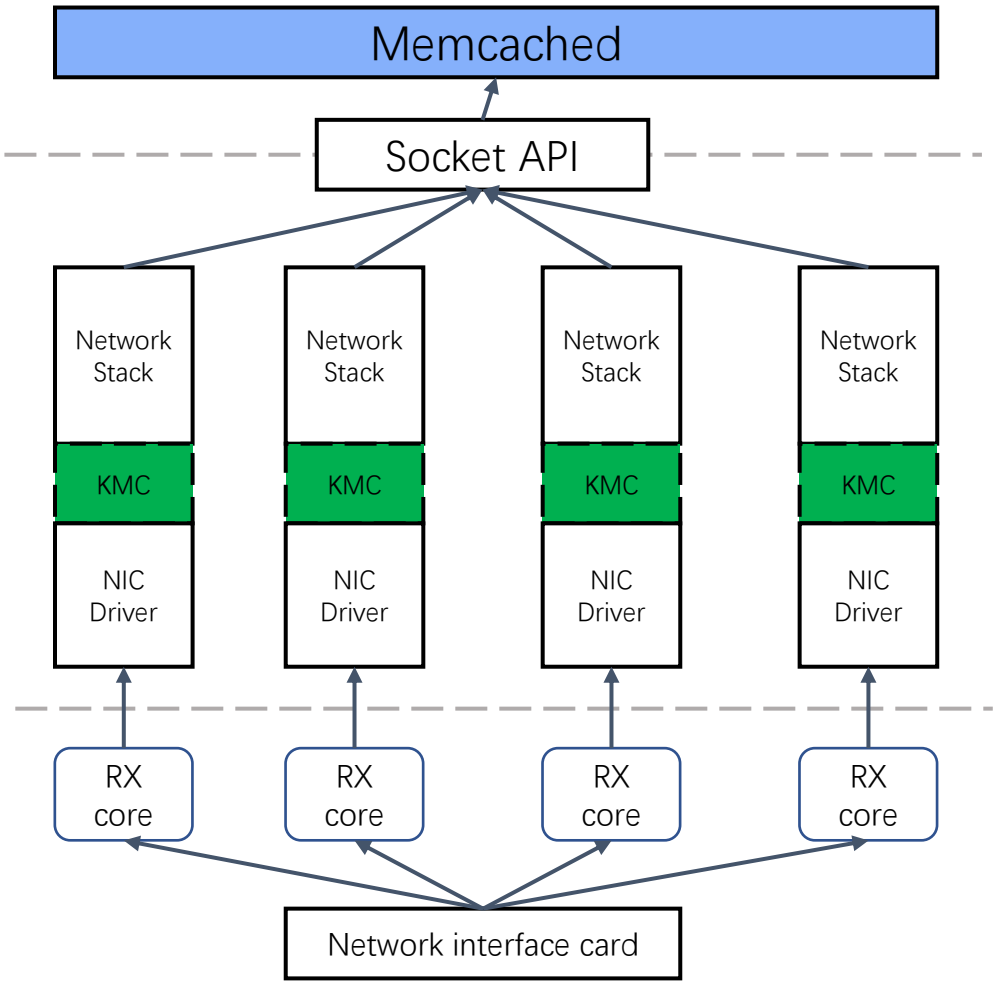
Figure 2: CPU usage of the three most used system calls by Memcached

大量的overhead消耗在内核协议栈以及系统调用上。
https://www.usenix.org/system/files/nsdi21-ghigoff.pdf

处理不及时导致大量报文丢失



基于ebpf的kernel memory cache

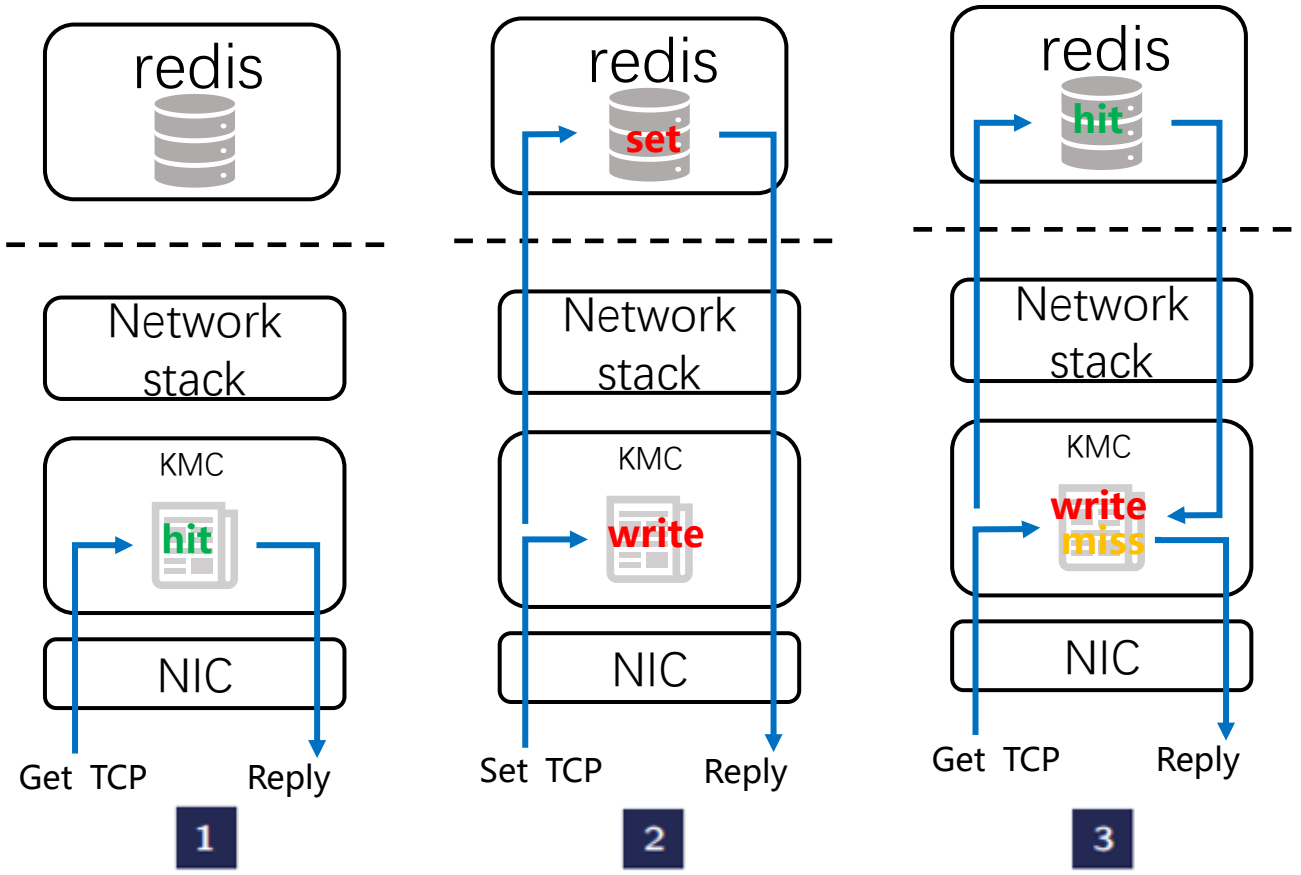


1、用hash map缓存数据

2、用ebpf程序更新、查找数据

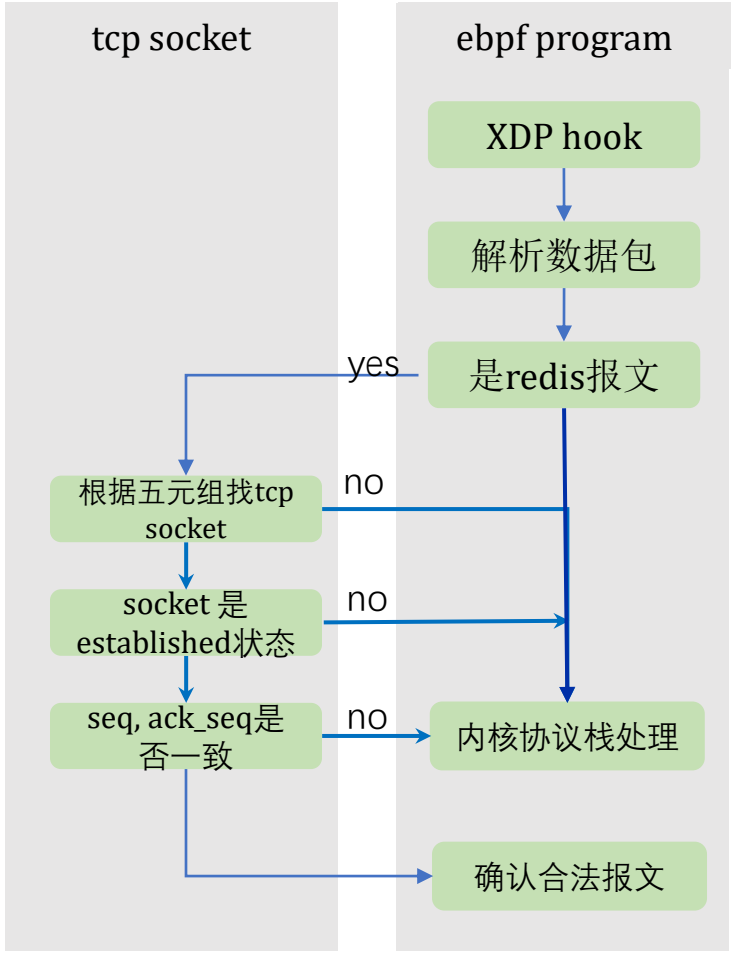
3、xdp redirect转发报文

用Kernel memory cache加速redis数据库

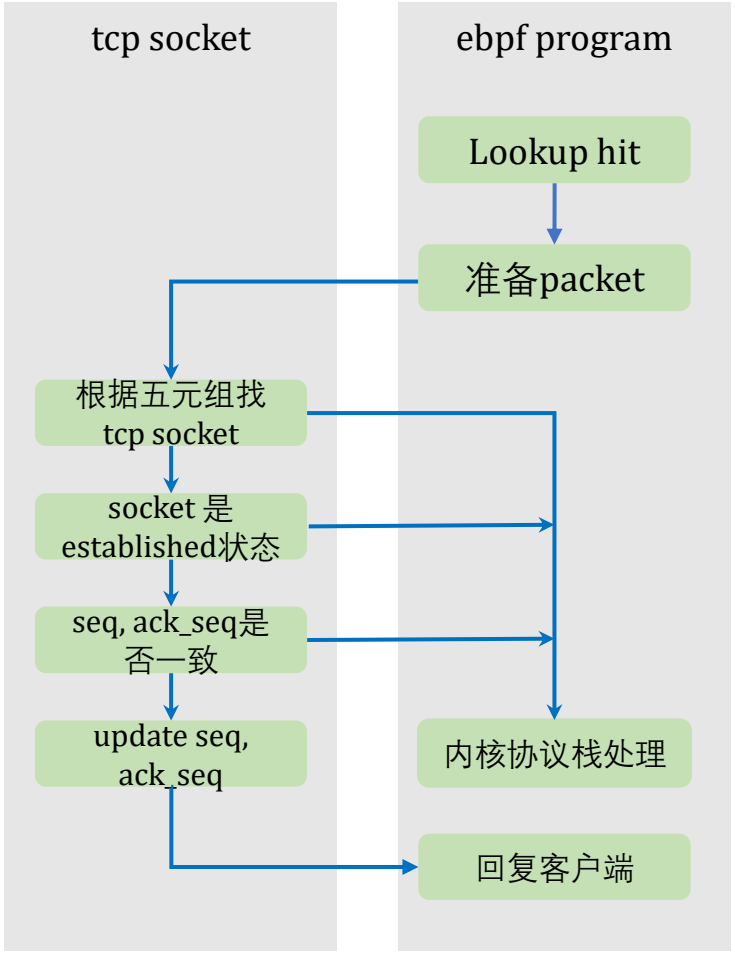


1、get命令，cache中命中，直接回复client端
2、set命令，更新cache，然后送到协议栈处理
3、get命令，cache中miss，从redis server中查找，回复client端

与TCP socket交互



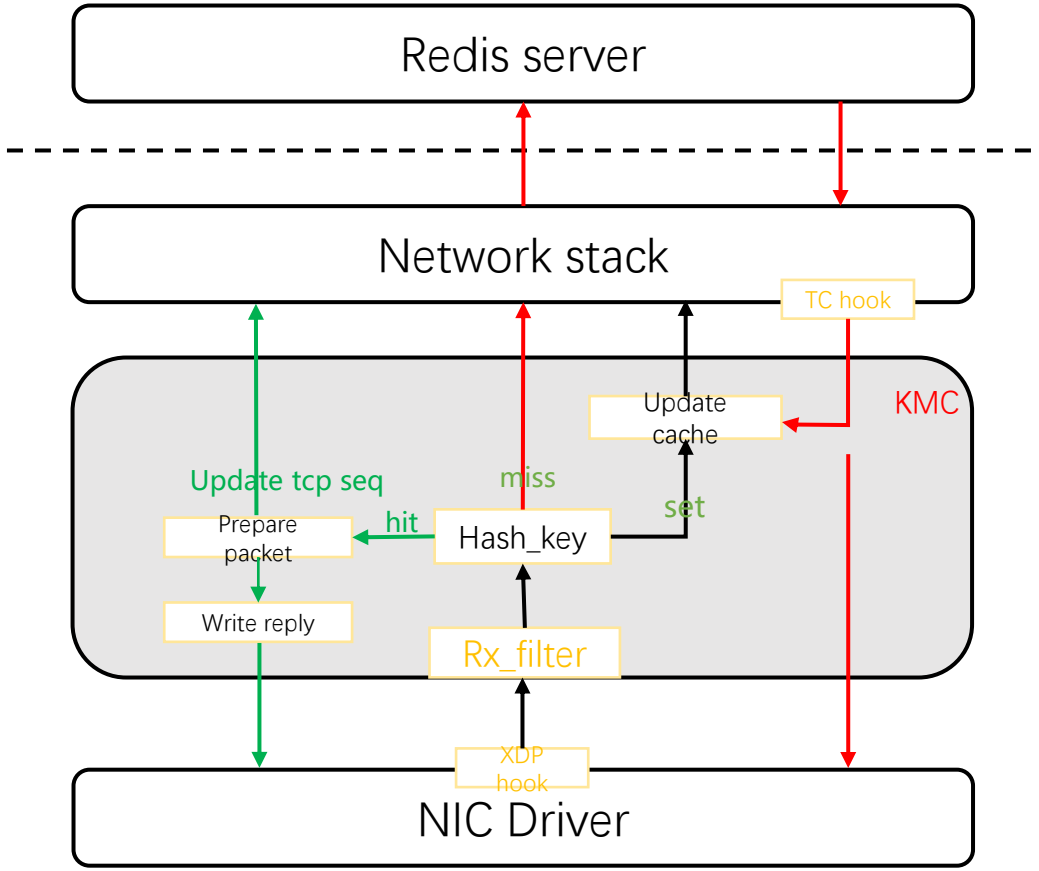
收包确认是否Redis合法报文流程



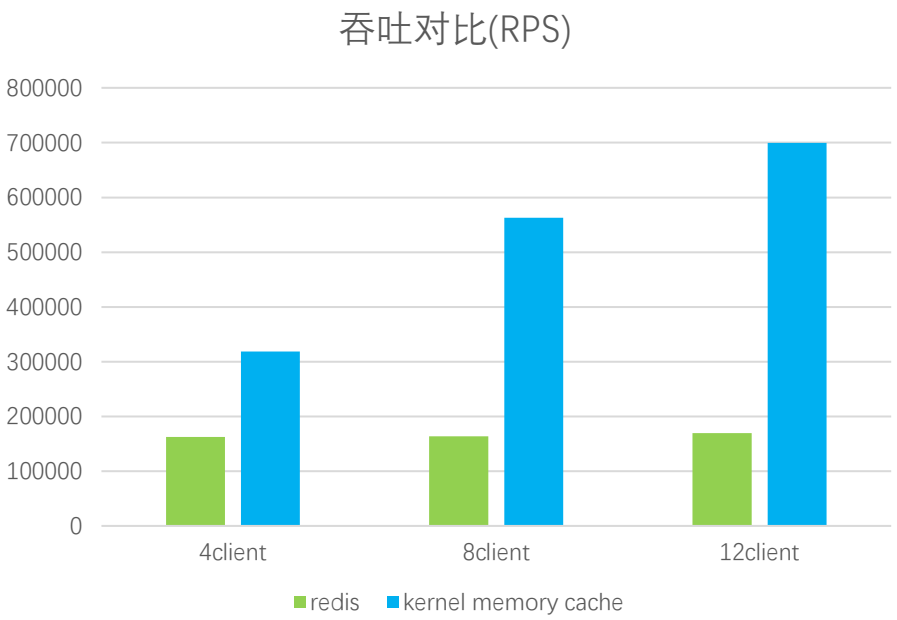
直接答复client端时更新tcp sequence

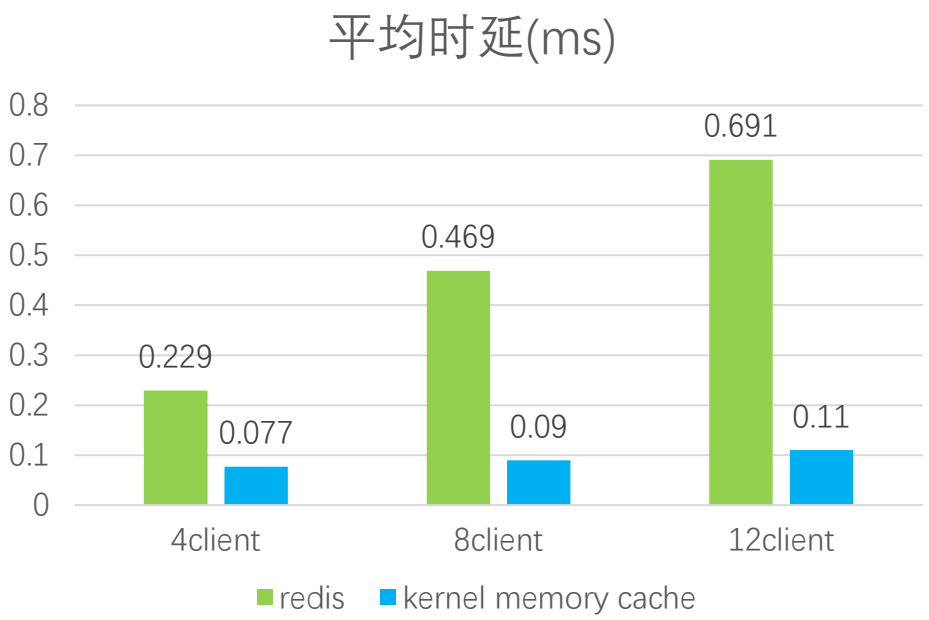
Rcv_nxt
Bytes_received
Snd_nxt
Copied_seq
Bytes_sent
Bytes_acked
Write_seq

BPF program全景视图



性能对比





理想情况下，redis数据库上吞吐提升至少337%，redis数据库上get命令平均时延减少84%。

todo

OpenEuler

1、解决内核态与用户态各有数据备份的问题

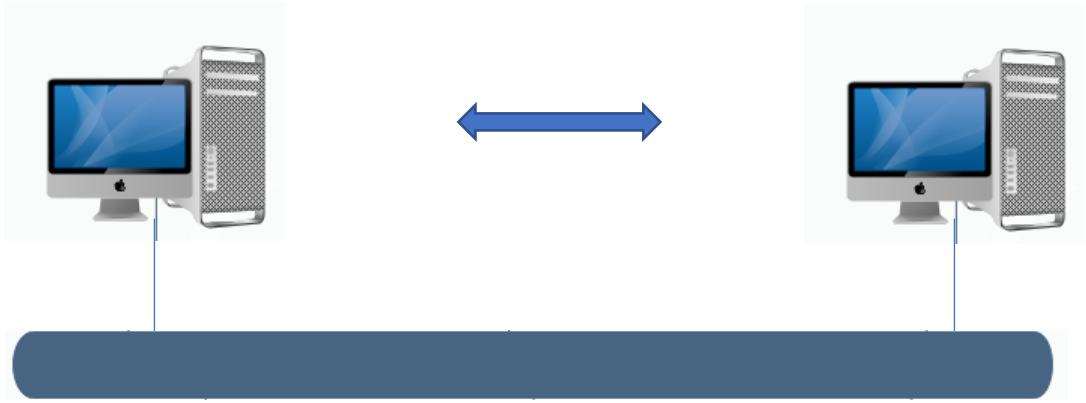
2、支持多数据库

3、支持TLS安全协议

目录

问题1： 网络压缩数据传输限制

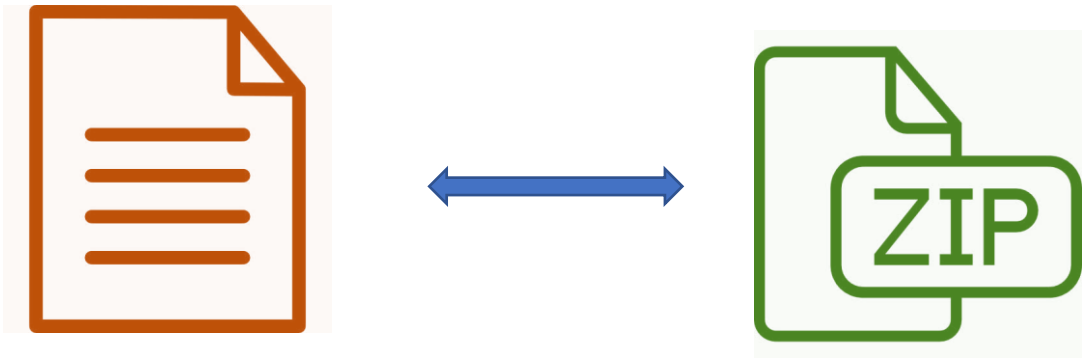
带宽限制



网络上数据传输速度受限于网络物理带宽
10Gb, 25Gb etc
压缩：

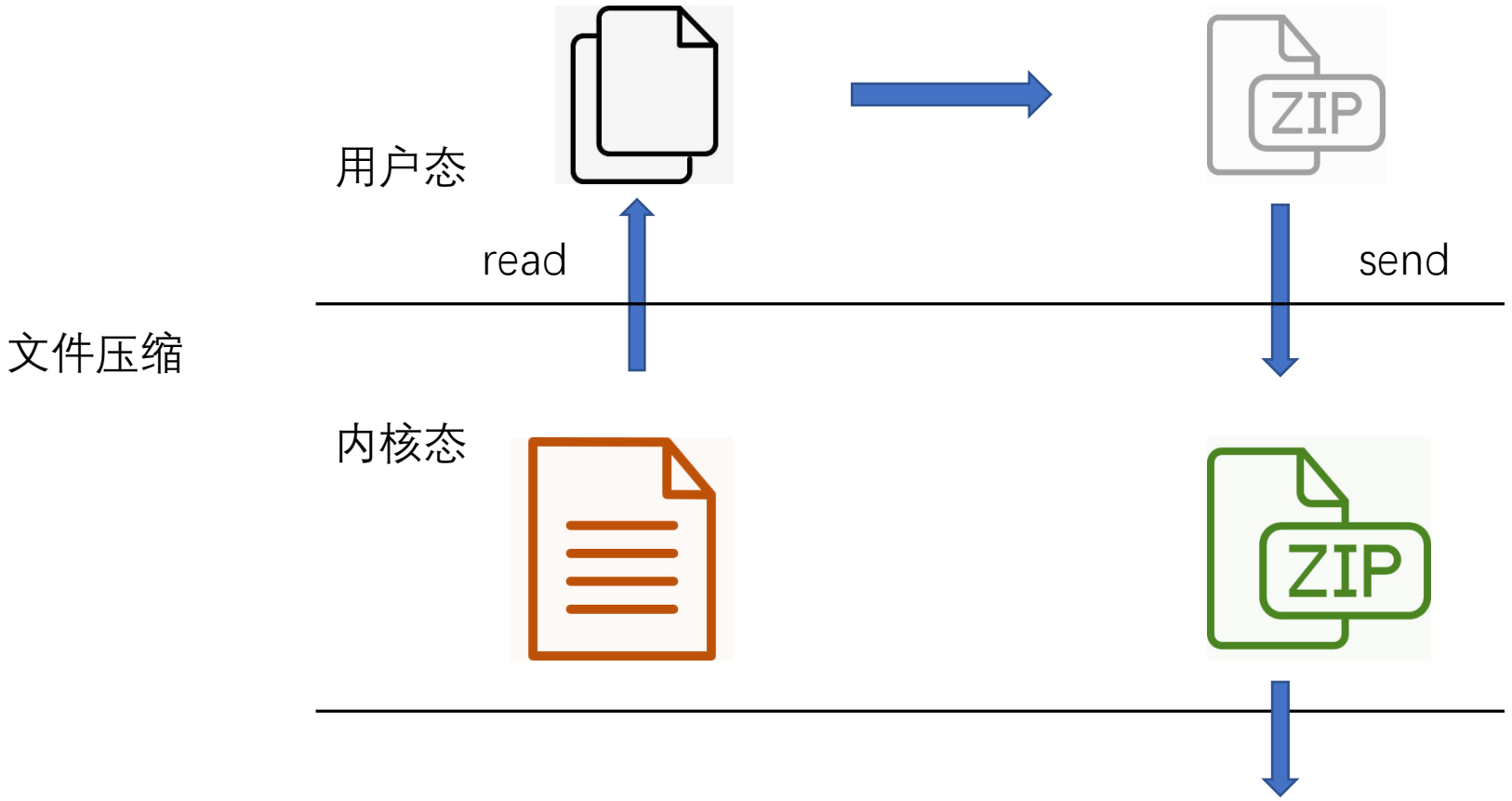
- 节省带宽
- 减少传输时间

压缩解压



- 压缩速度限制
  zstd ~4-5Gbps
- CPU利用率

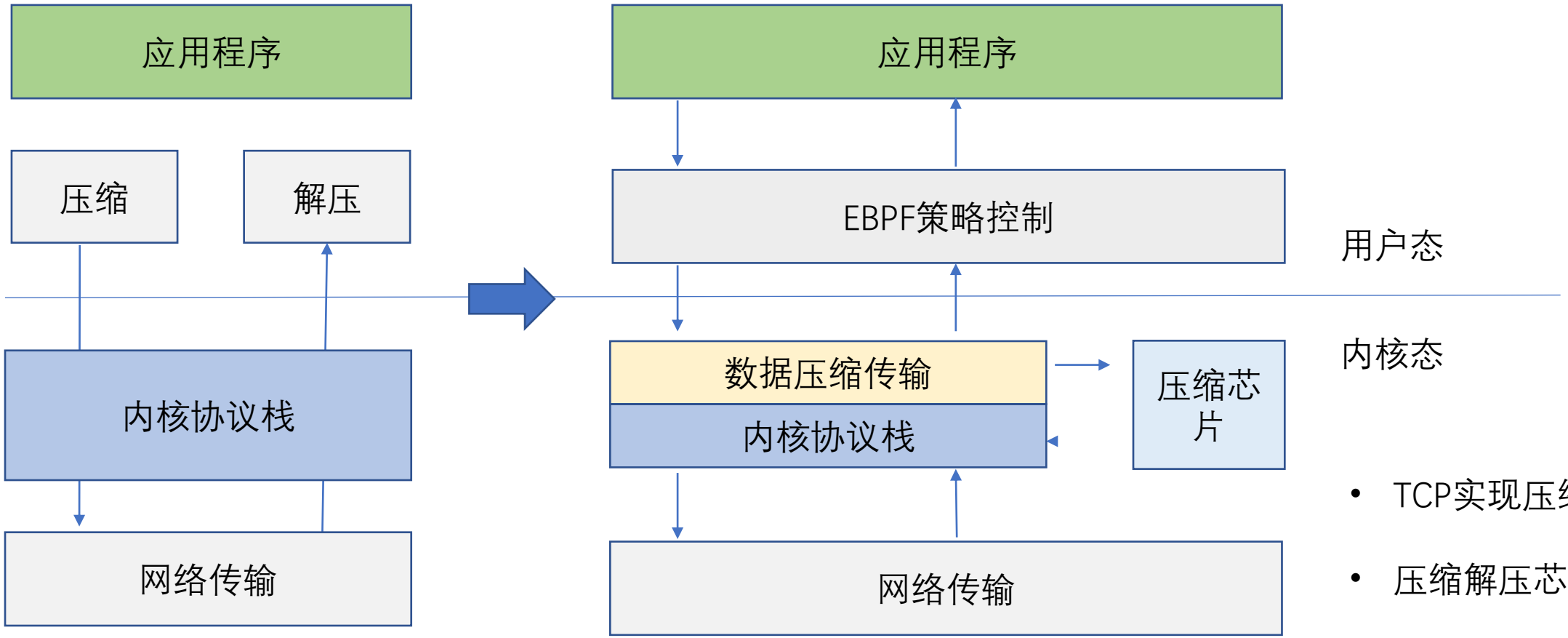
问题2：文件压缩传输



用户态压缩传输

多次数据拷贝

基于EBPF的网络压缩卸载



- TCP实现压缩卸载机制
- 压缩解压芯片加速
- EBPF策略控制

TCP压缩卸载机制

- sockopt
  - TCP_COMP_TX
  - TCP_COMP_RX

建链协商

- SYN

  添加 EXP_TCP_COMP选项，通知对方需要使用TX or RX 压缩, 压缩算法
- SYN/ACK

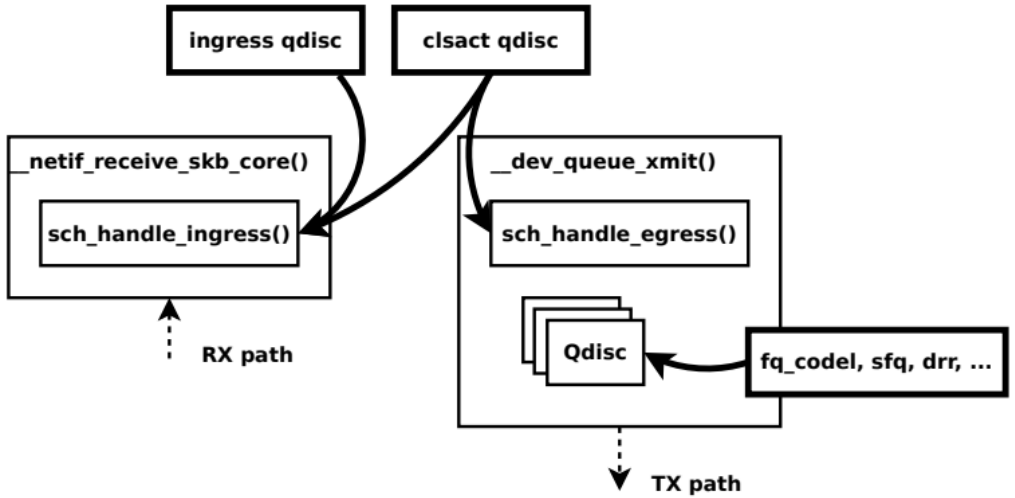
  添加 EXP_TCP_COMP选项，通知对方可以使用TX or RX 压缩, 压缩算法
- ACK

  根据两端的TX/RX状态设置本连接开启/关闭压缩

- 压缩解压
  - 块式压缩解压
    - 适用于部分压缩
  - 流式压缩解压
    - 适合于全压缩

- 系统调用
  - sendmsg
    - 发送前压缩
  - recvmsg
    - 接收前解压
  - sendfile
    - 内核压缩发送

EBPF实现策略



基于服务器端口的policy

BPF_SOCK_OPS_STATE_CB

TCP_SYN_SENT
TCP_SYN_RECV

探测点

- cls_bpf

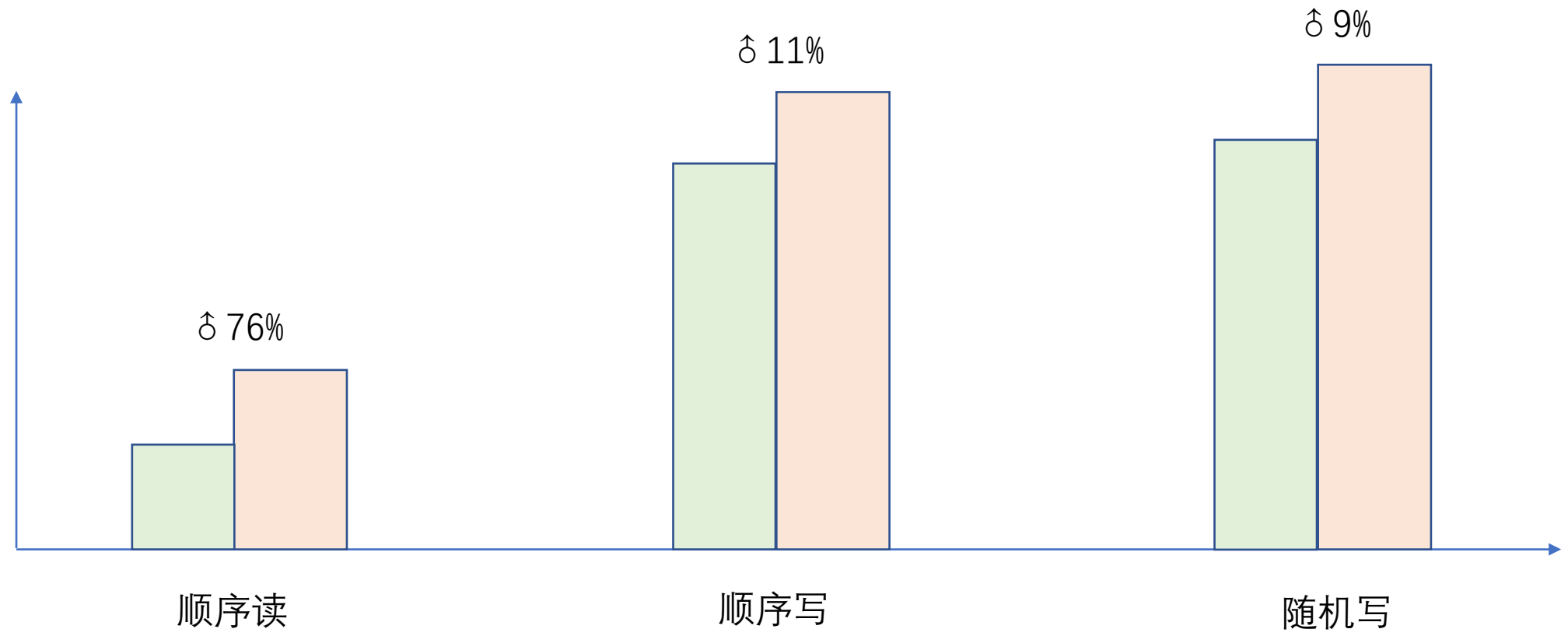
bpf_setsockopt

- 开启关闭压缩
- 设置压缩算法

探测数据

- 流量模型　　请求响应模式，单向数据传输等
- 流量探测　　流量是否超过压缩速率
- 可压缩性探测　　数据是否可压缩

性能对比




OpenEuler