网络性能调优分享

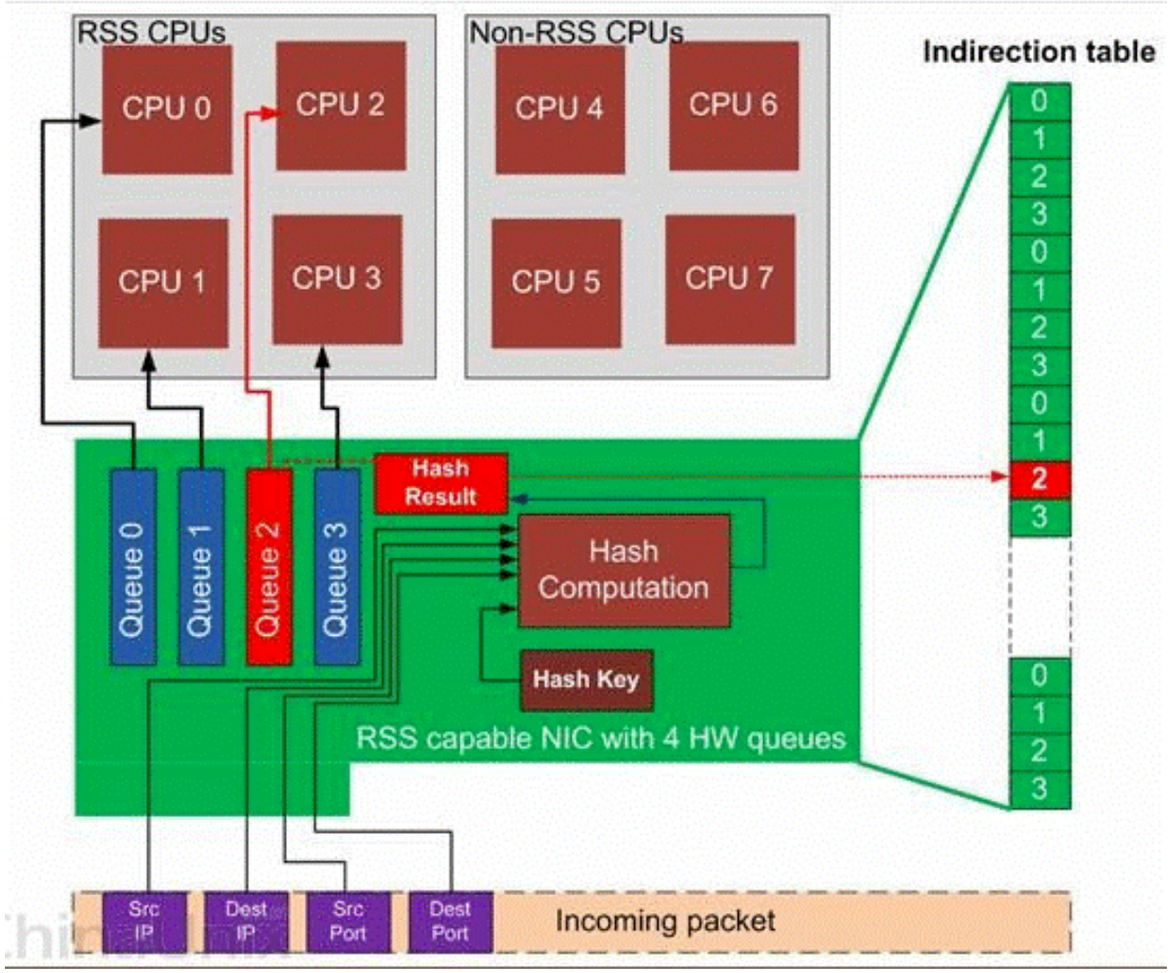
阿里云核心系统 鸣嵩
weibo @曹伟-鸣嵩

大纲

- 网卡硬件
- Linux网络报文处理
- 小报文性能瓶颈分析
- 小报文性能优化技术
- 协程

网卡硬件

- 千兆网卡
  - bnx2/tg3/igb
- MSI/MSI-X
- 多队列/多中断
- RSS



- 查看网卡型号

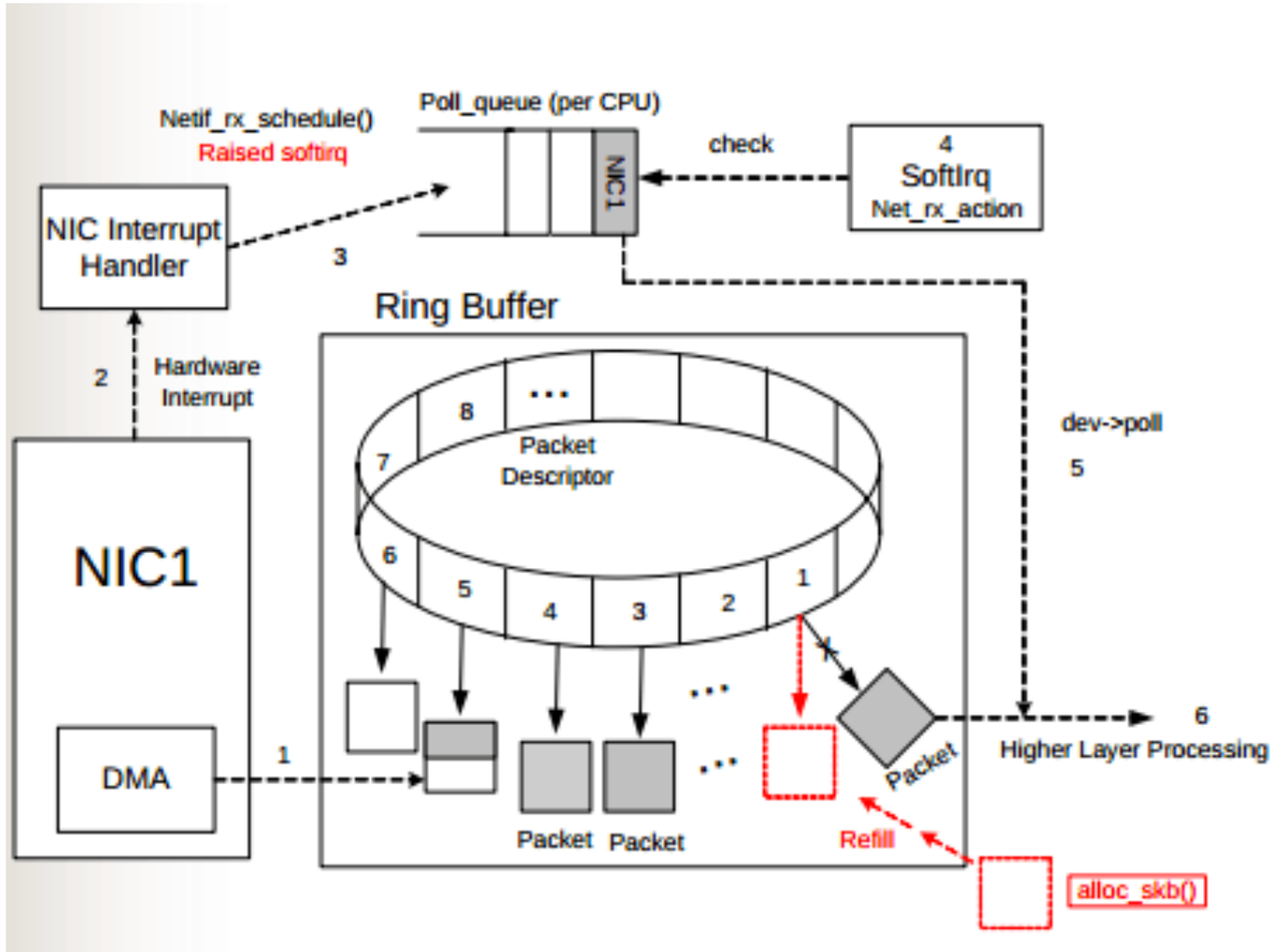
```
# ethtool -i eth0
```

- 查看网卡中断

```
# cat /proc/interrupts|grep "CPU\|eth0"
```

Linux系统接收报文流程

- NIC DMA
- 中断处理例程
- softirq
- 协议栈处理



- 查看软中断

```
# cat /proc/softirqs | grep "CPU\|NET"
```

- 实时看irq/softirq

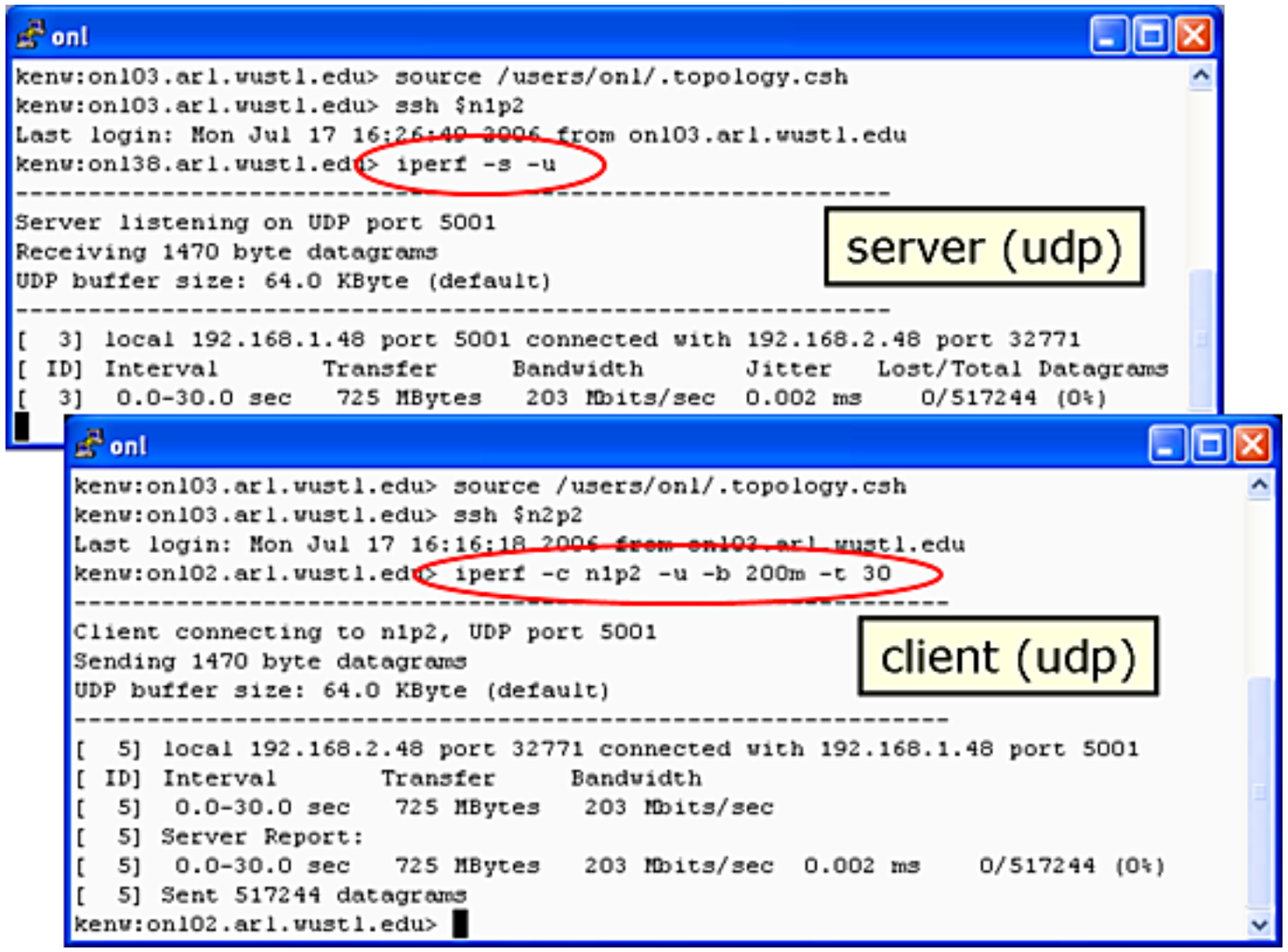
```
# mpstat -P ALL 1
```

网络性能指标

- 吞吐量 Mb/s
- PPS (packet per second)
- 丢包率
- 响应时间
- 资源使用率 cpu, memory, bus, etc

优化之前测试带宽

- iperf工具



检查各网卡pps

- watch ifconfig

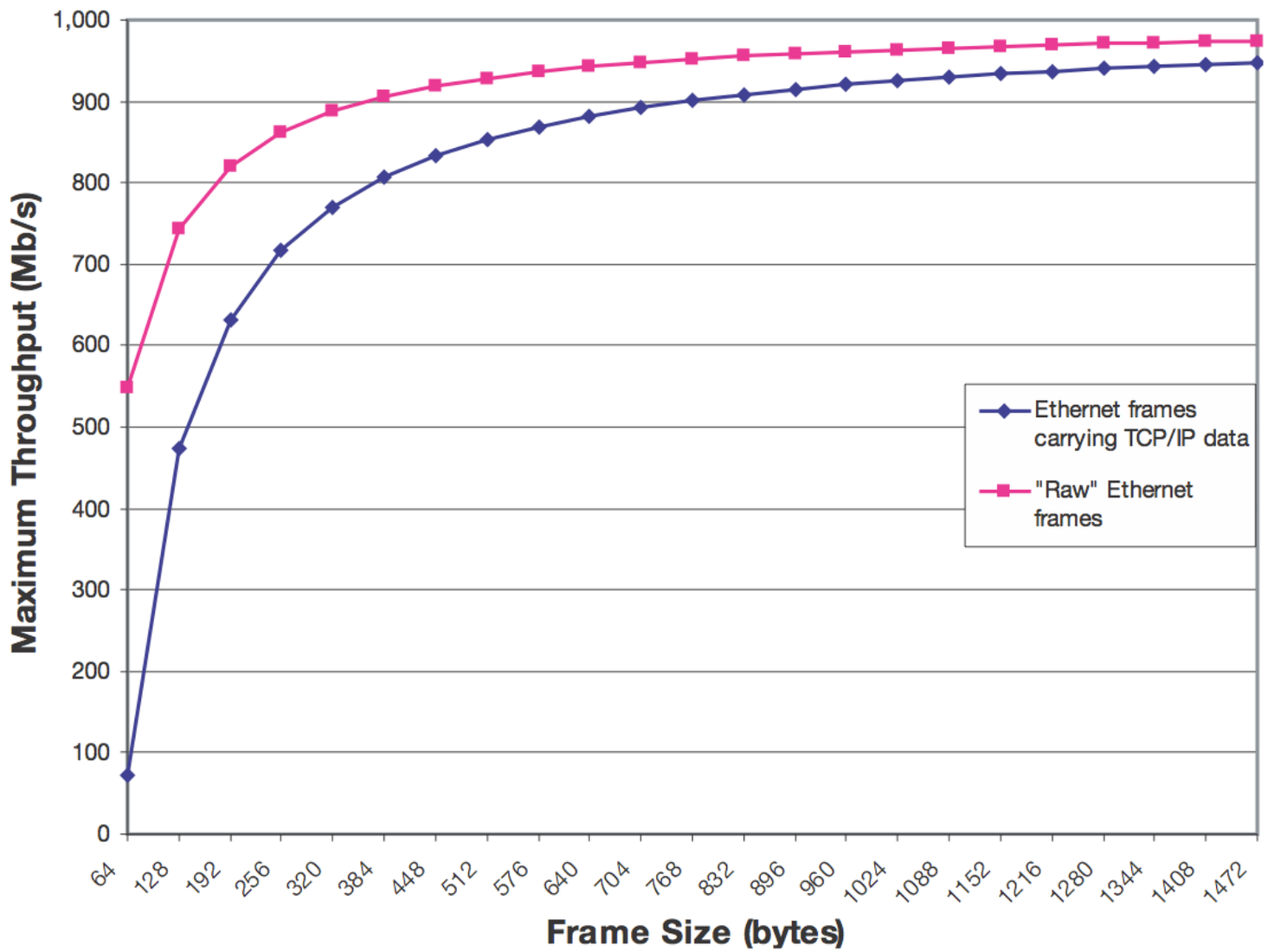
```
eth0      Link encap:Ethernet  HWaddr D4:3D:7E:6F:9B:24
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:9839833510 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11940770261 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1025760167992 (955.3 GiB)  TX bytes:1065853532243 (992.6 GiB)

eth1      Link encap:Ethernet  HWaddr D4:3D:7E:6F:9B:24
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:9948913891 errors:4799 dropped:0 overruns:0 frame:4799
          TX packets:11018319126 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1113642350186 (1.0 TiB)  TX bytes:985462796219 (917.7 GiB)
```

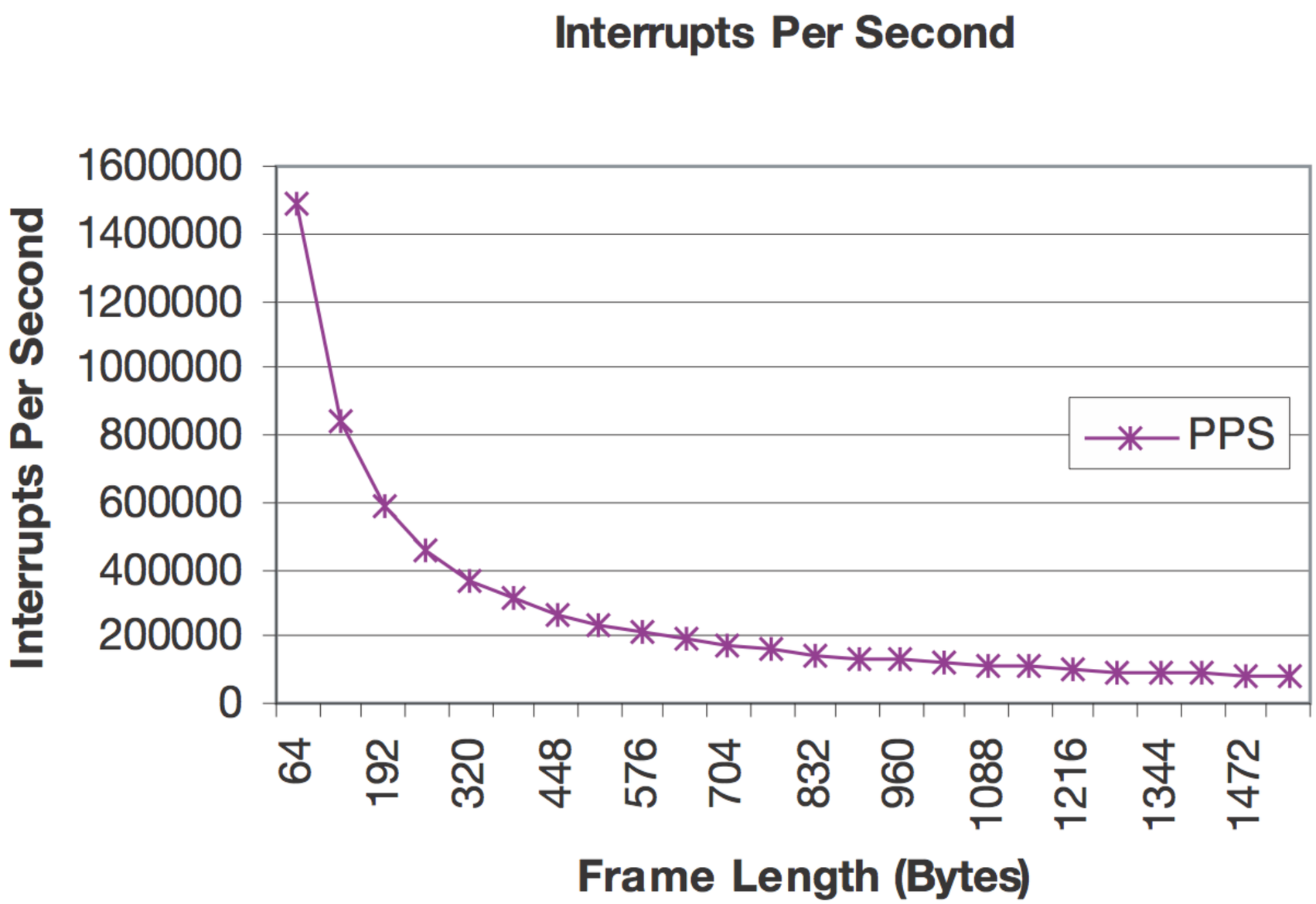
报文性能开销分析

- 每个报文固有的开销
  - 中断/上下文切换
  - 报文header
- 和报文长度相关的开销
  - CRC/Checksum计算

小报文理论上的瓶颈

	Minimum-sized "Raw" Ethernet frames	Minimum-sized Ethernet frames carrying TCP/IP data	Maximum-sized Ethernet frames carrying TCP/IP data
Preamble and Start-of-Frame Delimiter	8 bytes	8 bytes	8 bytes
Ethernet Header	14 bytes	14 bytes	14 bytes
TCP/IP Headers	N/A	40 bytes	40 bytes
Payload	46 bytes	6 bytes	1460 bytes
Ethernet Frame-Check-Sequence	4 bytes	4 bytes	4 bytes
Ethernet Inter-packet Gap	12 bytes	12 bytes	12 bytes
Total Packet Size	64 bytes	64 bytes	1518 bytes
Actual Bandwidth Consumed (i.e., packet size plus framing bytes)	84 bytes (672 bits)	84 bytes (672 bits)	1538 bytes (12,304 bits)
Link Speed	1 Gb/s	1 Gb/s	1 Gb/s
Theoretical Maximum Frame Rate	1,488,095 packets per second (approx.)	1,488,095 packets per second (approx.)	81,274 packets per second (approx.)
Theoretical Maximum Throughput	547 Mb/s (approx).	71 Mb/s (approx.)	949 Mb/s (approx).


Maximum Throughput (Mb/s)
1,000
900
800
700
600
500
400
300
200
100
0
64
128
192
256
320
384
448
512
576
640
704
768
832
896
960
1024
1088
1152
1216
1280
1344
1408
1472
Frame Size (bytes)
Ethernet frames carrying TCP/IP data
"Raw" Ethernet frames


Interrupts Per Second
Interrupts Per Second
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
PPS
64
192
320
448
576
704
832
960
1088
1216
1344
1472
Frame Length (Bytes)

提高小报文性能

- Packet polling
  - 不使用网卡中断
  - Linux NAPI
  - 20-25%性能提升
- 不使用TCP/UDP，私有协议，2层数据包

网络框架层合并小报文

- 应用层传递小报文到框架（异步/协程）
- 框架缓存小报文，延迟批量发出
- 框架层I/O线程
  - 与应用层线程的cpu affinity保持一致
  - 压缩 LZ4/Snappy
- Tradeoff 延迟v.s.吞吐量

协程

- ucontext实现
  - 大量sigprocmask调用
  - spin lock in kernel
  - 多线程下性能较差

```
ENTRY(__setcontext)
   /* Save argument since syscall will destroy it.  */
   pushq   %rdi
   cfi_adjust_cfa_offset(8)

   /* Set the signal mask with
      rt_sigprocmask (SIG_SETMASK, mask, NULL, _NSIG/8).  */
   leaq    oSIGMASK(%rdi), %rsi
   xorl    %edx, %edx
   movl    $SIG_SETMASK, %edi
   movl    $_NSIG8,%r10d
   movl    $__NR_rt_sigprocmask, %eax
   syscall
   popq    %rdi                    /* Reload %rdi, adjust stack.  */
   cfi_adjust_cfa_offset(-8)
   cmpq    $-4095, %rax            /* Check %rax for error.  */
   jae     SYSCALL_ERROR_LABEL     /* Jump to error handler if error.  */
```

```
int sigprocmask(int how, sigset_t *set, sigset_t *oldset)
{
        int error;

        spin_lock_irq(&current->sighand->siglock);
        if (oldset)
                *oldset = current->blocked;

        error = 0;
        switch (how) {
        case SIG_BLOCK:
                sigorsets(&current->blocked, &current->blocked, set);
                break;
        case SIG_UNBLOCK:
                signandsets(&current->blocked, &current->blocked, set);
                break;
        case SIG_SETMASK:
                current->blocked = *set;
                break;
        default:
                error = -EINVAL;
        }
        recalc_sigpending();
        spin_unlock_irq(&current->sighand->siglock);

        return error;
}
```

- setjmp/longjmp
  - 没有系统调用和锁，性能高
- 构造jmpbuf复杂
  - 参考”Portable Multithreading The Signal Stack Trick For User-Space Thread Creation”
- Trick
  - 利用ucontext进入协程，setjmp获得jmpbuf

Thanks