

# 了解网络

核心系统数据库组 余锋

<http://yufeng.info>

@淘宝褚霸

2012-07-29

```
$ sudo hwconfig
```

```
Chipset:          Intel 82801JIB A0 (ICH10)
```

```
Network:          eth0 (bnx2): Broadcom BCM5709 Gigabit,  
                  4c:b1:6c:8f:4a:bc, 1Gb/s <full-duplex>
```

```
Network:          eth1 (bnx2): Broadcom BCM5709 Gigabit,  
                  4c:b1:6c:8f:4a:bc, no carrier
```

```
OS:               RHEL Server 6.2, Linux 2.6.32-  
                  131.21.1.tb477.e16.x86_64 x86_64, 64-bit
```

**TCP/IP Offload Engine(TOE)** for increased bi-directional throughput and performance

Integrated iSCSI Host Bus Adapter(HBA) functionality

**Receive Side Scaling (RSS)**

TCP Segmentation

802.1q VLAN Tagging

Link Aggregation and Load Balancing

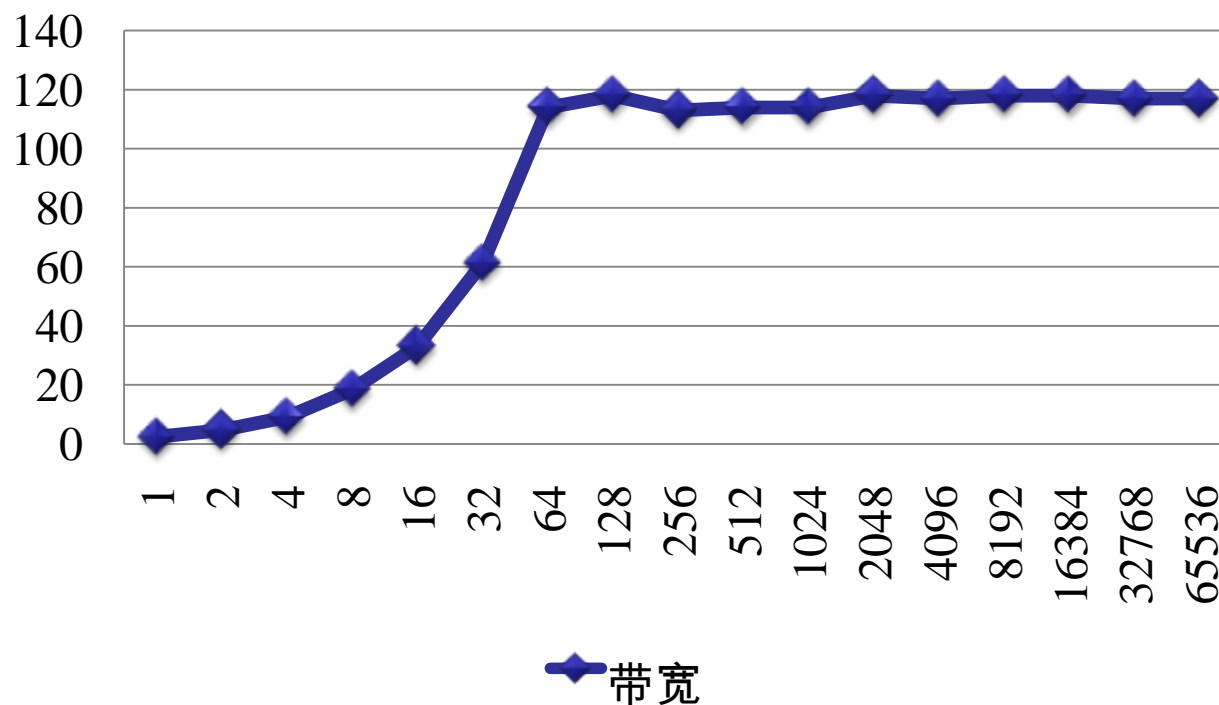
Jumbo Frames

iSCSI HBA

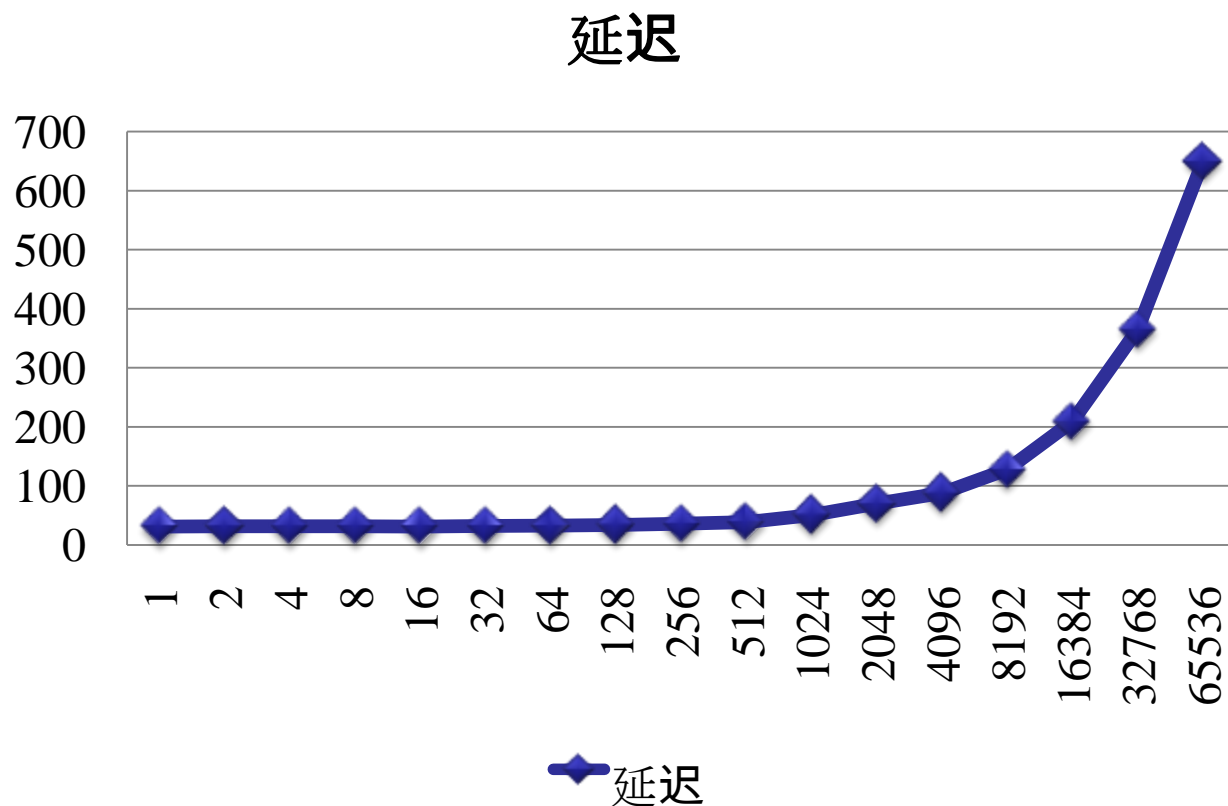
IPV6 Checksum

# 千兆网卡带宽

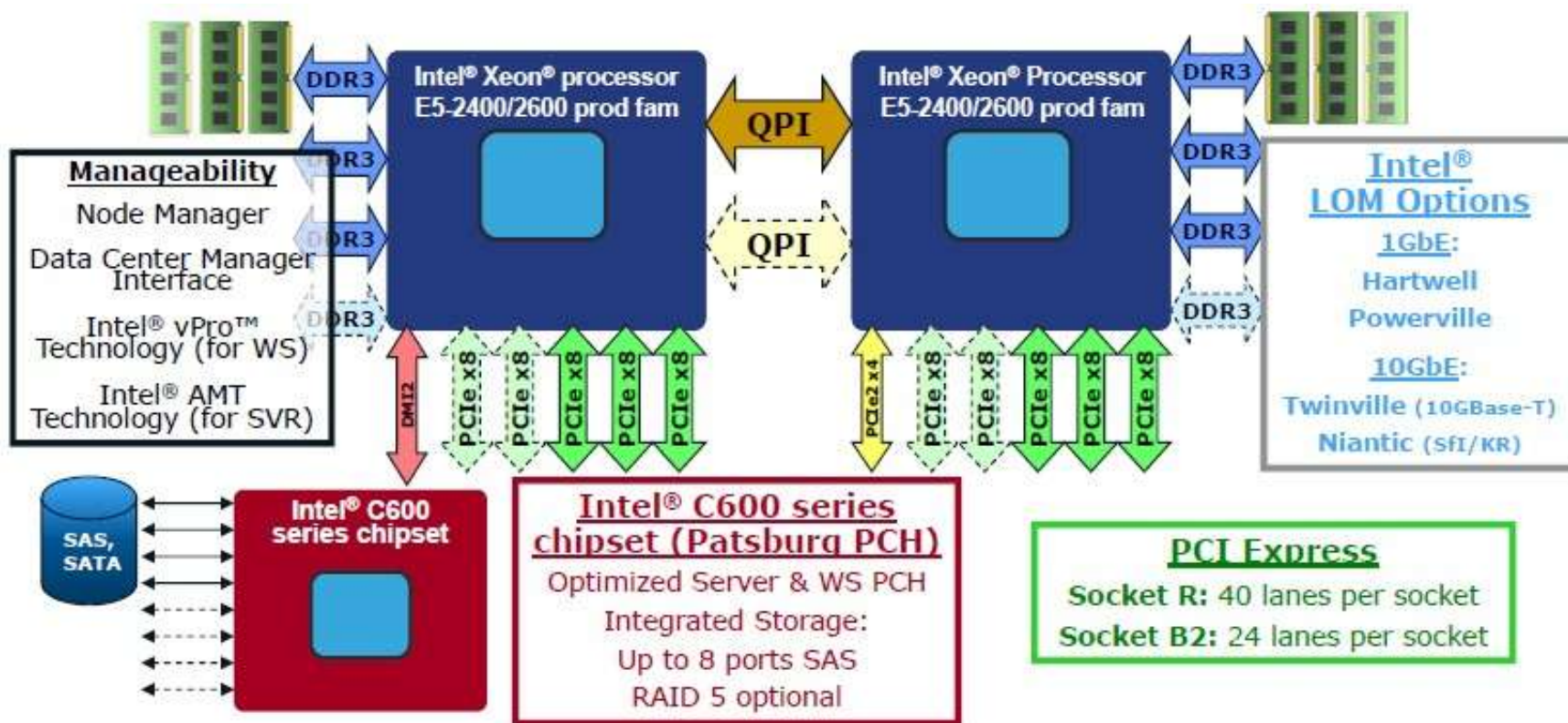
带宽



# 千兆网卡延迟



# 网卡新趋势(1)



# 网卡新趋势(2)

英特尔® 至强® 处理器 5500



英特尔 至强 处理器 5500

- 集成内存控制器
- 英特尔® QuickPath 接口(英特尔® QPI)

英特尔® 至强® 处理器 E5-2600



英特尔 至强 处理器 E5-2600

- 英特尔® 集成I/O
- 每个CPU插槽支持多达40条PCIe lane
- 英特尔® Data Direct I/O 技术

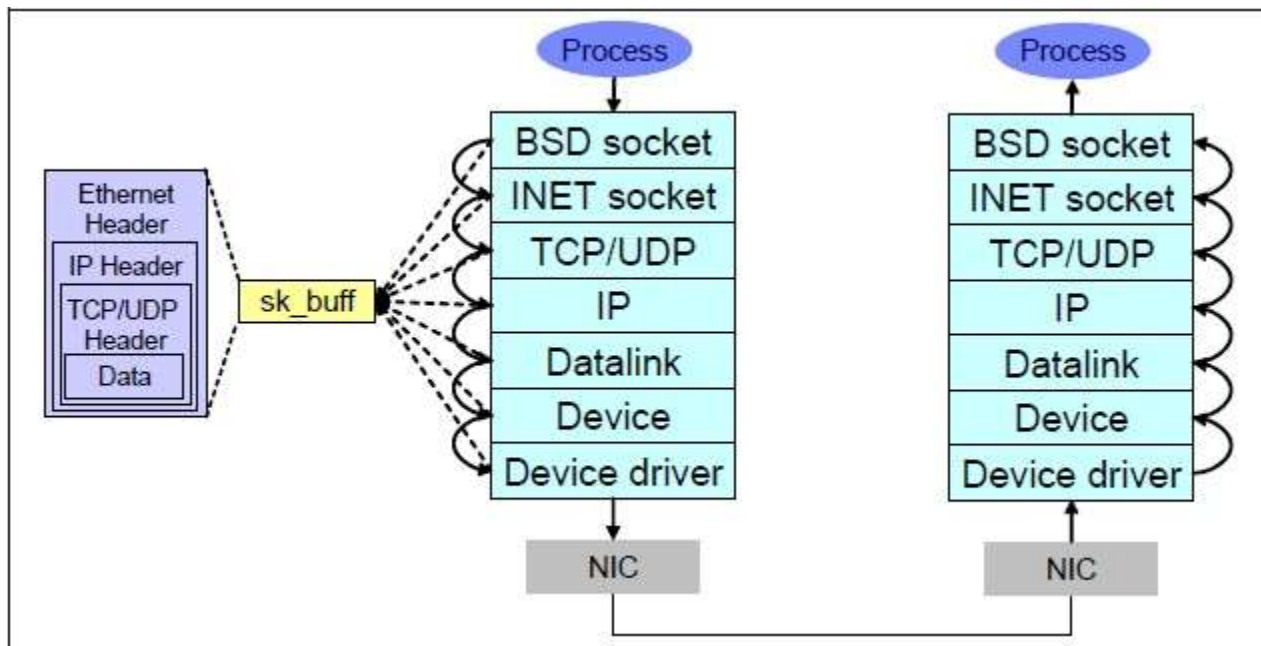
# 性能必知数字

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns



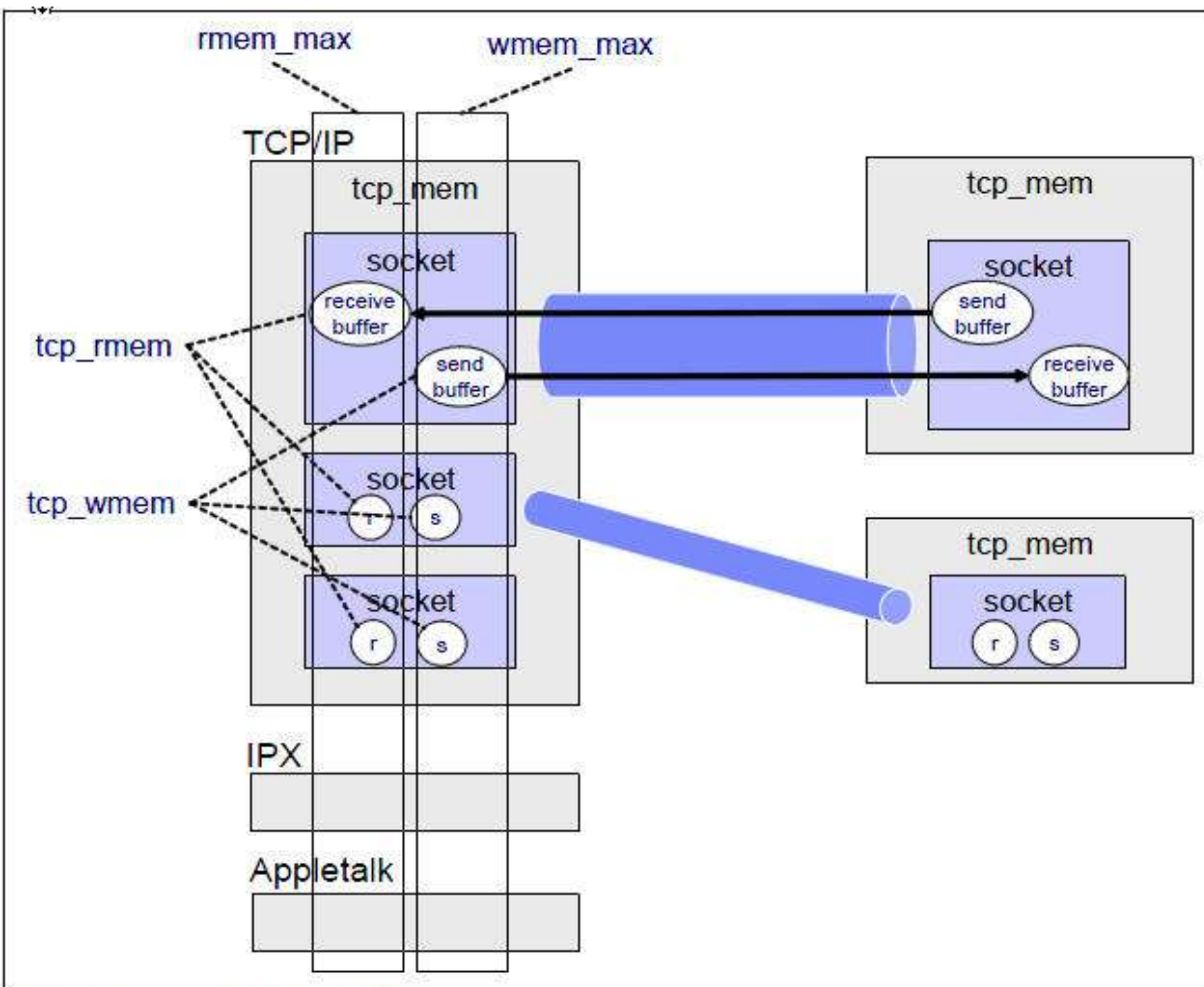


# Linux网络协议栈





# 微调协议栈



```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_keepalive_time = 1200
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.ip_local_port_range = 1024 65000
net.ipv4.tcp_max_syn_backlog = 8192
net.ipv4.tcp_max_tw_buckets = 5000
```

原则：dmesg可以观察到协议栈在抱怨什么，它抱怨什么我们解决什么！

TCP协议栈内存 不可交换物理内存

# 网卡bonding

```
[chuba@rds064075.sqa.cm4 systemtap-1.6]$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.6.0 (September 26, 2009)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: eth0 (primary_reselect always)
Currently Active Slave: eth0
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth0
MII Status: up
Link Failure Count: 0
Permanent HW addr: 4c:b1:6c:8f:4a:bc
Slave queue ID: 0

Slave Interface: eth1
MII Status: down
Link Failure Count: 0
Permanent HW addr: 4c:b1:6c:8f:4a:bd
Slave queue ID: 0
```

```
[root@my174 beam]# dstat
-----total-cpu-usage----- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send|  in   out| int  csw
  5   1  92   3   0   0|1417k 4532k|    0    0| 24k   33k|1559   25k
  0   0 100   0   0   0|    0    0| 420B 1092B|    0    0| 1002   106
  0   0 100   0   0   0|    0    0| 140B   364B|    0    0| 1003   122
  0   0 100   0   0   0|    0    0| 140B   364B|    0    0| 1003   108
```

硬中断:

```
yufeng@yufeng-laptop:~$ cat /proc/interrupts
           CPU0           CPU1
0:         3757785       3774177   IO-APIC-edge   timer
1:           4053         4013   IO-APIC-edge   i8042
8:              0           1   IO-APIC-edge   rtc0
9:          2198        2141   IO-APIC-fasteoi   acpi
```

- irqbalance 智能的均衡硬件中断。
- 手动 `[root@linux /]# echo ff > /proc/irq/19/smp_affinity`

软中断:

```
yufeng@yufeng-laptop:~$ cat /proc/softirqs
           CPU0           CPU1
HI:         0           0
TIMER:      2821708     13270908
NET_TX:         1           7
NET_RX:      60385     57542
BLOCK:      89454     87914
```

# RPS/RFS 解决softirq平衡

RPS is not automatically switched on, you have to configure it.

```
echo ffff >/sys/class/net/eth0/queues/rx-0/rps_cpus
```

Same for RFS if you prefer to use RFS

```
echo 16384 >/sys/class/net/eth0/queues/rx-0/rps_flow_cn
```

显著提高软中断的均衡性，大大提高性能。

e1000e on 8 core Intel

No RFS or RPS	104K tps at 30% CPU
No RFS (best RPS config):	290K tps at 63% CPU
RFS	303K tps at 61% CPU

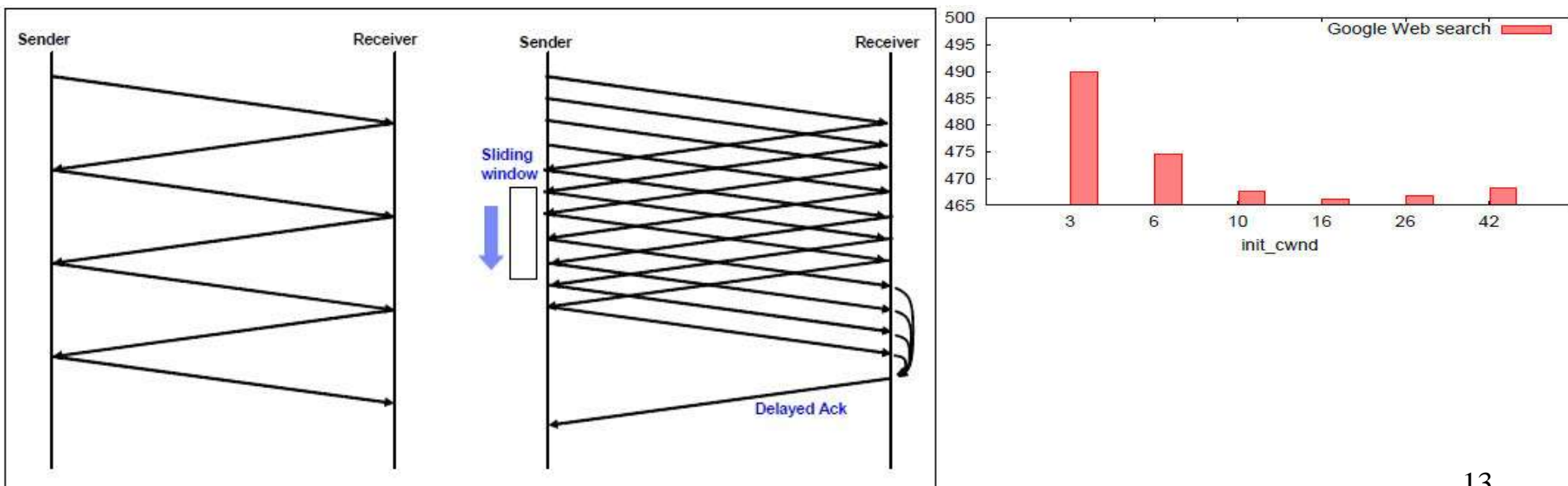
RPC test tps	CPU%	50/90/99% usec	latency	Latency	StdDev
No RFS/RPS	103K	48%	757/900/3185		4472.35
RPS only:	174K	73%	415/993/2468		491.66
RFS	223K	73%	379/651/1382		315.61



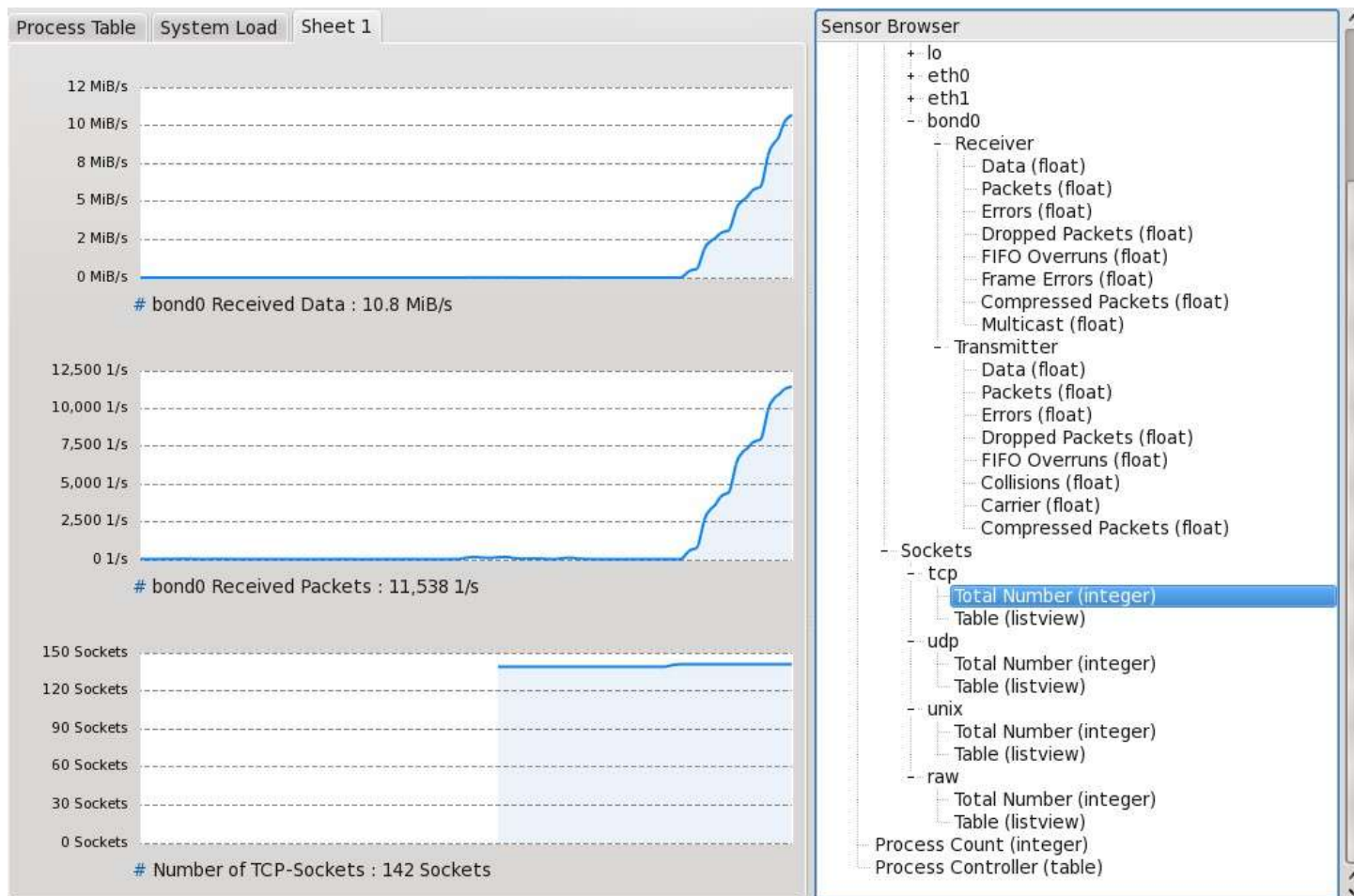
通过提高初始拥塞窗口的大小(3), 大大减少短连接的响应时间.

make sure your Linux kernel is 2.6.30 or higher.

ip route change [default via a.b.c.d dev ethX ... ] initcwnd 10



# ksysguard观察网络行为



The image shows the Wireshark network traffic analysis tool. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, and Help. Below the menu is a toolbar with various icons for file operations, packet capture, and analysis. A filter bar is present with a dropdown menu and buttons for Expression..., Clear, and Apply.

The main packet list table shows the following data:

No. -	Time	Source	Destination	Protocol	Info
8640	0.247739	10.0.0.4	10.0.0.5	TCP	33924 > 19302 [ACK] Seq=3 Ack=66 Win=5792 Len=0 TSV=77776188 TSER=70619327
8641	0.247762	10.0.0.5	10.0.0.4	TCP	19303 > 33924 [SYN] Seq=0 Len=0 MSS=1460 TSV=70619327 TSER=0 WS=7
8642	0.247768	10.0.0.4	10.0.0.5	TCP	33924 > 19303 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=77776188 TSER=70619327
8643	0.247855	10.0.0.5	10.0.0.4	TCP	19303 > 33924 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=70619327 TSER=77776188
8644	0.247863	10.0.0.5	10.0.0.4	TCP	[TCP segment of a reassembled PDU]
8645	0.247867	10.0.0.4	10.0.0.5	TCP	33924 > 19303 [ACK] Seq=1 Ack=65 Win=5792 Len=0 TSV=77776188 TSER=70619327
8646	0.247884	10.0.0.4	10.0.0.5	TCP	33924 > 19303 [PSH, ACK] Seq=1 Ack=65 Win=5792 [TCP CHECKSUM INCORRECT] Len=1
8647	0.247889	10.0.0.4	10.0.0.5	TCP	33924 > 19303 [FIN, ACK] Seq=2 Ack=65 Win=5792 Len=0 TSV=77776188 TSER=70619327

The detailed view of the selected packet (8644) shows the following information:

- Frame 8644 (130 bytes on wire, 130 bytes captured)
- Ethernet II, Src: Ibm\_3f:19:b5 (00:11:25:3f:19:b5), Dst: Ibm\_3f:19:b3 (00:11:25:3f:19:b3)
- Internet Protocol, Src: 10.0.0.5 (10.0.0.5), Dst: 10.0.0.4 (10.0.0.4)
- Transmission Control Protocol, Src Port: 19303 (19303), Dst Port: 33924 (33924), Seq: 1, Ack: 1, Len: 64
  - Source port: 19303 (19303)
  - Destination port: 33924 (33924)
  - Sequence number: 1 (relative sequence number)
  - [Next sequence number: 65 (relative sequence number)]
  - Acknowledgement number: 1 (relative ack number)
  - Header length: 32 bytes
  - Flags: 0x0018 (PSH, ACK)
  - Window size: 5888 (scaled)
  - Checksum: 0xdb1c [correct]
  - Options: (12 bytes)
  - TCP segment data (64 bytes)

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```

0020  00 04 4b 67 84 84 12 0d f0 79 30 54 b3 42 80 18  ..Kg.... .y0T.B..
0030  00 2e db 1c 00 00 01 01 08 0a 04 35 90 bf 04 a2  . . . . . . . . . .
0040  c5 3c 6e 65 74 70 65 72 66 00 6e 65 74 70 65 72  .<.netper f.netper
0050  66 00 6e 65 74 70 65 72 66 00 6e 65 74 70 65 72  f.netper f.netper
0060  66 00 6e 65 74 70 65 72 66 00 6e 65 74 70 65 72  f.netper f.netper
  
```

Window size (tcp.window\_siz P: 48303 D: 48303 M: 0 Drops: 0



```

IPTraf
TCP window table (sorted host ports) ----- Packets ----- Bytes Flag Interface
10.232.64.72:5903 > 2252 5624205 --A- bond02
10.13.88.163:58651 > 3417 159600 -PA- bond04
10.232.64.73:2888 > 62 3844 --A- bond0
10.232.64.72:33995 > 31 2232 -PA- bond0
10.232.64.75:33287 > 4 216 --A- bond0
10.232.64.72:4098 > 4 216 -PA- bond0
10.232.64.75:24007 > 6 312 --A- bond0
10.232.64.72:1021 > 6 312 --A- bond0
10.232.64.74:24007 > 6 312 --A- bond0
10.232.64.72:1023 > 6 312 --A- bond0
10.232.64.72:24007 > 6 312 --A- bond0
10.232.64.73:1022 > 6 312 --A- bond0
10.232.64.72:1022 > 6 312 --A- bond0
10.232.64.73:24007 > 6 312 --A- bond0
10.232.64.71:43418 > 4 216 --A- bond0
10.232.64.72:4096 > 4 216 -PA- bond0
10.232.64.73:986 > 3 156 --A- bond0
10.232.64.72:24010 > 3 156 --A- bond0
10.232.64.73:4190 > 4 216 --A- bond0
10.232.64.72:4098 > 4 216 -PA- bond0
TCP connections -----
ICMP dest unrch (port) (576 bytes) from 10.232.64.77 to 10.232.64.72 on bond
ICMP dest unrch (port) (576 bytes) from 10.232.64.77 to 10.232.64.72 on bond
UDP (121 bytes) from 10.232.64.72:53417 to 172.24.102.227:514 on bond0
UDP (1369 bytes) from 10.232.64.72:60779 to 10.232.64.77:25826 on bond0
UDP (1340 bytes) from 10.232.64.72:60779 to 10.232.64.77:25826 on bond0
ICMP dest unrch (port) (576 bytes) from 10.232.64.77 to 10.232.64.72 on bond
ICMP dest unrch (port) (576 bytes) from 10.232.64.77 to 10.232.64.72 on bond
UDP (1340 bytes) from 10.232.64.72:60779 to 10.232.64.77:25826 on bond0
UDP (1345 bytes) from 10.232.64.72:60779 to 10.232.64.77:25826 on bond0
ICMP dest unrch (port) (576 bytes) from 10.232.64.77 to 10.232.64.72 on bond
ICMP dest unrch (port) (576 bytes) from 10.232.64.77 to 10.232.64.72 on bond
TCP connections -----
Packets captured (all interfaces): 9077 | TCP flow rate: 6 1981.00 kbits/s
Up/Dn/PgUp/PgDn-scroll H-more TCP info W-chg actv win S-sort TCP X-exit

```

# socktop

----- PROCESSES -----								
PID	UID	#SEND	#RECV	SEND_KB	RECV_KB	PROT	FAMILY	COMMAND
16468	50920	23	22	0	2	IP	LOCAL	gnome-settings-
6658	50920	23	22	0	2	IP	LOCAL	gnome-settings-
9545	50920	23	22	0	2	IP	LOCAL	gnome-settings-
17365	50920	23	22	0	2	IP	LOCAL	gnome-settings-
7144	50920	23	22	0	2	IP	LOCAL	gnome-settings-
2196	50920	23	22	0	2	IP	LOCAL	gnome-settings-
16431	50920	15	16	2	0	IP	LOCAL	Xvnc
6610	50920	15	16	2	0	IP	LOCAL	Xvnc
9416	50920	15	16	2	0	IP	LOCAL	Xvnc
17306	50920	15	16	2	0	IP	LOCAL	Xvnc
=====								

# 网络系统调用代价

```
sudo ./syscalltimes -n qperf -t -u chuba -p `pgrep qperf`
```

System Call	Count	Total ns	Avg ns	Min ns	Max ns
wait4	2	4024520544	2012260272	2009330244	2015190300
read	46095	3485115497	75607	1776	5174715
write	28987	274827544	9481	5401	210723
sendto	2	21129	10564	9158	11971
select	14	5584573	398898	2022	4885391
setsockopt	2	5293	2646	2483	2810
listen	2	24955	12477	8776	16179
fcntl	2	5896	2948	2748	3148
getsockname	6	10325	1720	1140	2688
socket	8	3433358	429169	4825	1959357
recvmsg	4	13792	3448	1342	5976
bind	4	18815	4703	2954	5381
connect	2	26498	13249	8661	17837
ioctl	2	6380	3190	2875	3505
close	10	243409	24340	2087	158796
accept	3	16716181	5572060	353843	10778264
fork	2	178637	89318	76755	101882
rt_sigreturn	2	6677	3338	3033	3644
lseek	4	10511	2627	1437	3363
setitimer	6	9227	1537	1112	2132
times	4	9032	2258	1782	2584

# 协议栈缺内存引发问题

```
$ sudo stap sk_stream_wait_memory.stp  
1218230114875167: python(17631) blocked on full send buffer  
1218230114876196: python(17631) recovered from full send buffer  
1218230114876271: python(17631) blocked on full send buffer  
1218230114876479: python(17631) recovered from full send buffer
```

```
$ netstat -s|grep drop
```

```
281340 outgoing packets dropped
```

```
77 packets dropped from out-of-order  
queue because of socket buffer overrun
```

```
7 ICMP packets dropped because they were  
out-of-window
```



```
$ sudo dropwatch -l kas  
Initalizing kallsymsa db  
dropwatch> start  
Enabling monitoring...  
Kernel monitoring activated.  
Issue Ctrl-C to stop monitoring  
1 drops at netlink_unicast+251  
15 drops at unix_stream_recvmsg+32a  
3 drops at unix_stream_connect+1dc
```

```
[chuba@rds064075.sqa.cm4 ~]$ sudo ethtool eth0
Settings for eth0:
    Supported ports: [ TP ]
    Supported link modes:   10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full

    Supports auto-negotiation: Yes
    Advertised link modes:  10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full

    Advertised pause frame use: No
    Advertised auto-negotiation: Yes
    Speed: 1000Mb/s
    Duplex: Full
    Port: Twisted Pair
    PHYAD: 1
    Transceiver: internal
    Auto-negotiation: on
    MDI-X: Unknown
    Supports Wake-on: g
    Wake-on: d
    Link detected: yes
```



谢谢大家！