Linux TCP/IP Performance on Long Fat-pipe Network toward Internet2 Land Speed Record

Junji Tamatsukuri, Takeshi Yoshino, Yutaka Sugawara, Mary Inaba, Kei Hiraki
Data Reservoir Project / University of Tokyo
Theme of This presentation

• We present many practical result of the highest performance single stream TCP
  – Linux TCP stacks have essentially high performance rather than other OS.
  – But many problem to get 10Gbps class performace.
  • Many reasons of these problems are unknown now.
Our Project

- “Data Reservoir” is a Data Grid System for Scientific Data

- Our Goal
  - High performance data site replication between long distance places.
  - We need high TCP stream performance to realize “Data Reservoir”.

Our Internet2 IPv4 Land Speed Record History

Distance bandwidth product
Pbit m / s

Year
2000 2001 2002 2003 2004 2005 2006 2007

- **2000**: 1,000
- **2004/11/9**: Data Reservoir project
- **2004/12/24**: 216 Pbit m / s
- **2005/11/10**: 240 Pbit m / s
- **2006/2/20**: 264 Pbit m / s

10 Gbps * 30,000km

2004/11/9
Data Reservoir project
WIDE project
149 Pbit m / s
Our Internet2 IPv6 Land Speed Record History

Distance bandwidth product
Pbit m / s

10 Gbps * 30,000km

2005/10/29
Data Reservoir project
WIDE project
167 Pbit m / s

2005/11/13
Data Reservoir project
WIDE project
208 Pbit m / s
Our TCP/IP result on LFN

- Our project has the most higher experience TCP/IP communication on LFN
- We have 4 points of our tuning approach
  1. Precise logging tools for LFN high speed communication
  2. Real LFN over 30,000km and Pseudo LFN environment in our labo.
  3. Many result of TCP/IPv4, v6 on both LFN
  4. TCP tuning method for LFN
1, hardware logging tool TAPEE

• Packet logging tool with precise timestamp.
  – To analyze TCP stream
  – To view physical layer behavior

• Hardware/Software Solution
  – Packet processing
  – Data capturing/ Data analyzing
1. Function of TAPEE

- Preprocessing by hardware
  - Copy packets by light TAP
  - Remove payload to decrease data size
  - Adding precise timestamps by 100ns
  - Packing Several frames to decrease packet rate.
Our Experimental Environment

• TCP communication between Linux Servers (Sender → Receiver)
  – Application: iperf-2.0.2

• Servers
  – Opteron / Xeon

• Network
  – Real / Pseudo Network
2, Real Long Fat-pipe Network

- LSR needs 30,000km Network (Our network is 33,000km)
- Sum of distance among Routing Point
- Oversea Circuit consists of OC-192/SDH
- 10GbE WAN-PHY (9.26Gbps)
2, Our Real LFN Diagram

Tokyo
APAN/JGN2

Hitachi GS4000

Foundry NI40G

Cisco ONS15454

Fujitsu XG800

sender
receiver

Atlantic Ocean
Hitachi GS4000

LAN PHY (10Gbps)

WAN PHY (9.286Gbps)

Hitachi GS4000

Pacific Ocean
JGN2

Send

Japanese

Europe

Seattle
Pacific Northwest Gigapop

Send

Chicago
StarLight

Hitachi GS4000

Force10 E1200

Nortel HDXc

Surfnet

Amsterdam
SARA

Hitachi GS4000

Nortel HDXc

CA*net

L3 Switch

L2 Switch

L1 Switch
2, Pseudo LFN Environment

• Insert long latency among servers artificially
• Hardware
  – TGNLE (Our project develop)
    • Upto 1600ms RTT
  – Anue H series Network Emulator
    • Upto 800ms RTT
• Test environment before Real LFN experiment.
3, Linux Server Specification
Architectural Difference

• PCI-X performance
  – PCI-X 1.0 : Opteron (8.5Gbps)
  – PCI-X 2.0 : Xeon MP (over 10Gbps)

• CPU performance
  – Memory Latency/Bandwidth
    • Opteron With Memory controller
    • Xeon without Memory controller

• Interrupts to CPUs
3, Hardware 1: Opteron

- Processor: Dual Opteron 250 (2.6GHz)
- MotherBoard: Rioworks HDAMA
- Memory: 2GB (Overclock DDR CL2)
- I/O Performance limitation: PCI-X 1.0 8.6Gbps (133MHz x 64bit)
3, Hardware 2 : Xeon MP

- Processor: Quad Xeon MP 3.66GHz (IBM x260)
- Memory: 32GB (DDR2 x 4bank)
- No I/O Performance limitation : PCI-X 2.0 (266MHz x 64bit)
## TCP/IP performance matrix

<table>
<thead>
<tr>
<th></th>
<th>BIC TCP</th>
<th>IPv4</th>
<th>IPv6</th>
<th>IPv4</th>
<th>IPv6</th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8G limit</td>
<td>Chelsio T110</td>
<td>opteron</td>
<td>7.2G (90%)</td>
<td>5.9G (75%)</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Chelsio N210</td>
<td>opteron</td>
<td>x</td>
<td>x</td>
<td>7.0G (85%)</td>
<td>7.0G (85%)</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Xeon</td>
<td>x</td>
<td>x</td>
<td>5.75G (75%)</td>
<td>5.75G (75%)</td>
<td>5.43G (65%)</td>
<td>1Gbps (12%)</td>
</tr>
<tr>
<td>10G limit</td>
<td>Chelsio T310</td>
<td>Xeon</td>
<td>9.0G (90%)</td>
<td>5.43G (54%)</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Linux 2.6.12 IPv4 Xeon Performance

- The highest performance stream
Linux 2.6.16 IPv6 Opteron Performance

Stream opteron9c-v6:0 -> opteron6c-v6:0 No.1

Throughput (1 ms moving ave.)
Throughput (1000 ms moving ave.)

Throughput [Gbps]

Time [sec]
Software TCP performance on Linux 2.6.16 later

- **Window Buffer Size**
  - Theoretical Value = RTT * Bandwidth
- **NAPI**
  - Effective for high interrupts from network arrival.
  - We use static optimized interrupt interval.
- **TSO**
  - Effective for reducing packet checksum calculation
- **TCP Scaling**
  - Delayed Ack effective for High performance.
  - But longer scaling time is needed.
# TCP/IP Performance Matrix

<table>
<thead>
<tr>
<th>8G limit</th>
<th>Chelsio T110</th>
<th>opteron</th>
<th>7.2G (90%)</th>
<th>5.9G (75%)</th>
<th>x</th>
<th>x</th>
<th>x</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chelsio N210</td>
<td>opteron</td>
<td>x</td>
<td>x</td>
<td>7.0G (85%)</td>
<td>7.0G (85%)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Xeon</td>
<td>x</td>
<td>x</td>
<td>5.75G (75%)</td>
<td>5.75G (75%)</td>
<td>5.43G (65%)</td>
<td>1Gbps (12%)</td>
<td></td>
</tr>
<tr>
<td>10G limit</td>
<td>Chelsio T310</td>
<td>Xeon</td>
<td>9.0G (90%)</td>
<td>5.43G (54%)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

- IPv4
- IPv6
Linux 2.6.12 IPv6 Xeon Performance

- IPv6 result on same host
Linux 2.6.17 IPv6 Xeon Performance

- Current IPv6 performance.
- This result have packet dropping in peak.
# TCP/IP performance matrix

<table>
<thead>
<tr>
<th></th>
<th>8G limit</th>
<th>10G limit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chelsio T110</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>opteron</td>
<td>IPv4: 7.2G (90%) IPv6: 5.9G (75%)</td>
<td>x IPv4: 7.0G (85%) IPv6: 7.0G (85%) x IPv4: 5.75G (75%) IPv6: 5.75G (75%)</td>
</tr>
<tr>
<td><strong>Chelsio N210</strong></td>
<td>opteron: x IPv4: 7.0G (85%) IPv6: 7.0G (85%)</td>
<td>x IPv4: 5.75G (75%) IPv6: 5.75G (75%)</td>
</tr>
<tr>
<td>Xeon</td>
<td>x IPv4: 9.0G (90%) IPv6: 5.43G (54%)</td>
<td>x IPv4: 5.43G (65%) IPv6: 1G (12%) x</td>
</tr>
<tr>
<td><strong>Chelsio T310</strong></td>
<td>Xeon: x IPv4: 9.0G (90%) IPv6: 5.43G (54%)</td>
<td>x IPv4: 5.43G (65%) IPv6: 1G (12%) x</td>
</tr>
</tbody>
</table>

The table compares TCP/IP performance across different versions (2.6.12, 2.6.17, 2.6.18-rc5) and platforms (IPv4, IPv6). The performance metrics include throughput in Gbps and percentage utilization. The table highlights the performance differences between IPv4 and IPv6, as well as the impact of different versions on performance.
Linux 2.6.18-rc5 IPv4 TSO on

- Only 5.8Gbps on Xeon system
- Relative stable performance
Linux 2.6.18-rc5 IPv4 TSO off

- Only 5.6Gbps on Xeon system
- Stable performance
Linux 2.6.18-rc5 IPv6 GSO off

- Unstable performance
- Packet loss happened in the kernel
- TCP stack send duplicate ack for retransmission, but network doesn’t drop any packets.
Linux 2.6.18-rc5 IPv4 TSO on RTT=10ms (1s average)

- RTT=10ms peak result is the same of RTT=500ms
Linux 2.6.18-rc5 IPv6 GSO off
RTT=10ms (1s average)

Stream x260-1c-v6-g:45222 -> x260-2c-v6-g:5001 No.1

Throughput (1000 ms moving ave.)

- Average performance is 3.8Gbps.
- This is almost 60% result of IPv4.
Linux 2.6.18-rc5 IPv6 GSO off
RTT=10ms (1ms and Stream Info)

Stream x260-1c-v6-g:45222 -> x260-2c-v6-g:5001 No.1

Throughput [Gbps]

Time [sec]
Ack Framing problem

- SONET has frame
- Some network instruments small packet packing into same frame
- Ack packets has no interval or frame interval
Sending ACK packets

- Ack Sending
Pseudo LFN behavior

- Same packet interval is in Receiver side.
Real LFN behavior

• Almost packet interval push into 0 μs by framing
Real LFN vs Pseudo LFN

• Both LFN shows the same performance macroscopically
  – 1s average performance is same.
• Real LFN shows the modified packet arrival interval.
  – SONET framing packing Ack packets.
• Receiver side receives short packets burst on Real LFN.
  – Real LFN needs higher packet receiving performance.
Toward the new LSR on IPv6

• We hope GSO stability on IPv6
  – The current performance bottle neck is a CPU performance of checksum Calculation.
• Stable performance on PCI-X 2.0 or PCI-Express x 16
  – There is a performance shield on 6 Gbps
Summary

• Our LSR high performance TCP communication
  – We measured detailed network stream packets and showed many result
  – Feedback tuning for high performance
• TCP communication on LFN is difficult, but we can utilize till the same performance no relation with its latency.
acknowledge

• Thanks for advice and support
  – Prof. Akira Kato University of Tokyo, ITC
  – WIDE Project
  – JGNII, IEEAF
  – Pacific Northwest Gigapop
  – AlaxalA Networks

• Thanks for providing Oversea Network
  – JGNII, SURFnet, IEEAF, CANARIE/CA*net
Linux 2.6.16 IPv6 Opteron Performance

Current TCP stack shows stable window scaling on both IPv4 and IPv6
• Large Window buffer occurs packet loss on peak performance.
Adverized Window is grown faster than window size.
Slow window scaling is effect of delayed ack.
Almost 100kbps on same network
We met same condition on 2.6.12 IPv4 with TSO
Linux 2.6.18-rc5 IPv6 GSO off
RTT=10ms (1s and Stream Info)
3. Network Interface Card

- PCI-X 1.0
  - Chelsio N210

- PCI-X 2.0
  - Chelsio T310
### Linux 2.6.12 IPv4 Xeon Performance

<table>
<thead>
<tr>
<th>usage(%)</th>
<th>function</th>
<th>IPv4 T310 receiver side</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.1211</td>
<td>timer_interrupt</td>
<td></td>
</tr>
<tr>
<td>10.5991</td>
<td>mwait_idle</td>
<td></td>
</tr>
<tr>
<td>6.1435</td>
<td>find_busiest_group</td>
<td></td>
</tr>
<tr>
<td>5.7787</td>
<td>apic_timer_interrupt</td>
<td></td>
</tr>
<tr>
<td>4.3406</td>
<td>account_system_time</td>
<td></td>
</tr>
<tr>
<td>3.8784</td>
<td>scheduler_tick</td>
<td></td>
</tr>
<tr>
<td>3.4558</td>
<td>run_timer_softirq</td>
<td></td>
</tr>
<tr>
<td>3.2597</td>
<td>t3_intr</td>
<td></td>
</tr>
<tr>
<td>2.7998</td>
<td>schedule</td>
<td></td>
</tr>
<tr>
<td>2.463</td>
<td>_do_IRQ</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>usage(%)</th>
<th>function</th>
<th>IPv4 T310 sender side</th>
</tr>
</thead>
<tbody>
<tr>
<td>39.1652</td>
<td>copy_user_generic</td>
<td></td>
</tr>
<tr>
<td>7.1538</td>
<td>tcp_sendmsg</td>
<td></td>
</tr>
<tr>
<td>3.7135</td>
<td>tcp_ack</td>
<td></td>
</tr>
<tr>
<td>3.592</td>
<td>t3_eth_xmit</td>
<td></td>
</tr>
<tr>
<td>3.3121</td>
<td>put_page</td>
<td></td>
</tr>
<tr>
<td>2.7089</td>
<td>t3_intr</td>
<td></td>
</tr>
<tr>
<td>2.0278</td>
<td>timer_interrupt</td>
<td></td>
</tr>
<tr>
<td>1.9771</td>
<td>free_tx_desc</td>
<td></td>
</tr>
<tr>
<td>1.8016</td>
<td>skb_release_data</td>
<td></td>
</tr>
<tr>
<td>1.6117</td>
<td>kfree</td>
<td></td>
</tr>
</tbody>
</table>

- In IPv4, TSO or TOE is available. This result use TSO on sender side.
- Memory copy spend most of time, both side. From the effect of TSO, packet processing load is relatively small.
Current Performance

• We measured newest kernel 2.6.18-rc5 performance on same pseudo environment.
  – Limitation: Chelsio T310 could not execute on latest kernel for driver structure change.
  – Chelsio N210 (limited by PCI-X performance, 8.5Gbps)
RTT=10ms Performance

• Same test executed on small latency network.
  – Packet losses decrease the performance smaller than large latency network.
  – same packet loss phenomena shown in short interval
  – But relative higher performance than LFN.
Our result

• TCP Stream Behavior
  – Linux 2.6.12, 2.6.17, 2.6.18-rc5

• Behavior difference between Real LFN and Pseudo LFN

• Current Kernel performance
### Linux 2.6.12 IPv6 Xeon Performance

<table>
<thead>
<tr>
<th>usage(%)</th>
<th>function</th>
<th>usage(%)</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.6659</td>
<td>csum_partial_copy_generic</td>
<td>48.2684</td>
<td>csum_partial_copy_generic</td>
</tr>
<tr>
<td>22.9821</td>
<td>copy_user_generic_c</td>
<td>4.0249</td>
<td>timer_interrupt</td>
</tr>
<tr>
<td>12.8658</td>
<td>csum_partial</td>
<td>3.0945</td>
<td>tcp_sendmsg</td>
</tr>
<tr>
<td>3.9911</td>
<td>timer_interrupt</td>
<td>2.7058</td>
<td>cache_alloc_refill</td>
</tr>
<tr>
<td>2.1931</td>
<td>kfree</td>
<td>2.3096</td>
<td>memcpy</td>
</tr>
<tr>
<td>2.1852</td>
<td>process_responses</td>
<td>1.7153</td>
<td>free_block</td>
</tr>
<tr>
<td>1.795</td>
<td>tcp_v6_rcv</td>
<td>1.6977</td>
<td>put_page</td>
</tr>
<tr>
<td>1.7642</td>
<td>fib6_lookup</td>
<td>1.5748</td>
<td>_rmqueue</td>
</tr>
<tr>
<td>1.1321</td>
<td>eth_type_trans</td>
<td>1.4846</td>
<td>do_gettimeoffset_pm</td>
</tr>
<tr>
<td>1.1299</td>
<td>memcpy</td>
<td>1.3065</td>
<td>_mod_page_state</td>
</tr>
<tr>
<td>1.0183</td>
<td>free_block</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IPv6 N210 receiver side**  **IPv6 N210 sender side**

- In IPv6 have no hardware function, packet production use most of CPU power.
- CPU load is very high especially in sender side.
- Memory copy load is also high. This is same behavior on IPv4.