Link Aggregation for High Speed Single TCP Stream Transfer

Data Reservoir Project
The University of Tokyo

Kei Hiraki

National Institute of Information and Communications Technology

Our Goal

- TCP/IP single stream communication over 10 Gbps
  - Important for many scalable applications

- Finding solution for very high-speed TCP communications
  - Finding bottlenecks in end-to-end solution
  - Application to DISK to DISK data transfer
  - Web systems for over 10G bandwidth

Network Interface for >10 Gbps

- 40 GE NIC (Mellanox)
  - Very expensive
  - B/W bottleneck at PCI-express (8 lanes) ~32Gbps
- Bundle multiple 10Gbps interfaces for single stream TCP/IP

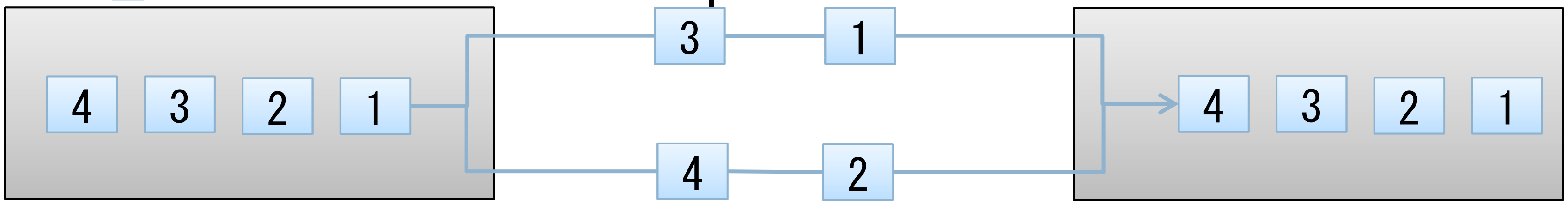
Link Aggregation

- Aggregate multiple NIC for single connection
  - Load balancing and fault tolerance
  - Similar to RAID storage
- Already widely used --- IEEE 802.1AX-2008

- Single-stream TCP/IP is difficult on IEEE802.1AX
- Unit of load balancing
  - 1) IP address / MAC Address
  - 2) Based on loads at each NIC

Link Aggregation on Linux

- bonding Virtual device in Linux
  - bonding : Link Aggregation on Linux
- Mode1 (round-robin Mode)
  - Equally distribute packet by packet
  - Sender:Transmit equally divided packet to two links
  - Receiver:Receive packet from two links then



Link Aggregation on Linux

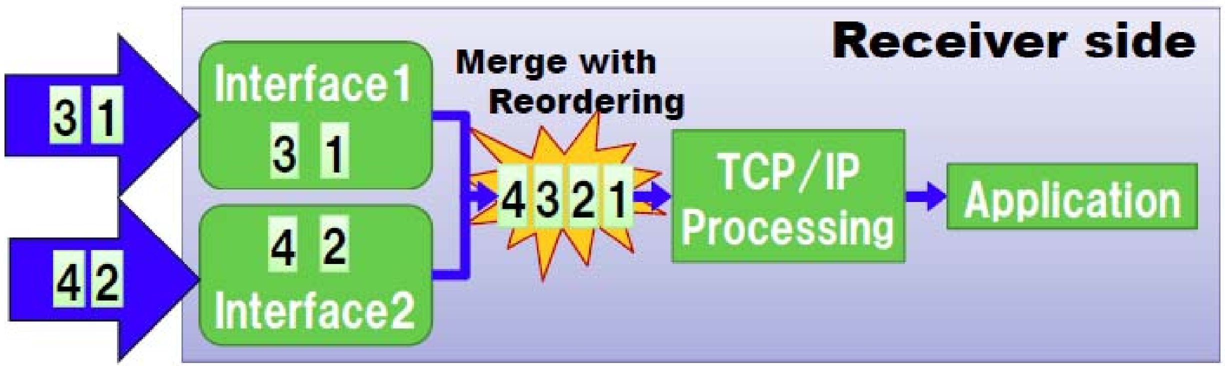
Overview

Performance Problem :Link Aggregation

- **Main Causes of performance decrease**
  - ✓ Concentration of loads on specific core
  - ✓ Shuffle of TCP packet sequence
  - ✓ Heaviness of TCP processing

Our solution

- ✓ Reorder packets before TCP processing



- ✓ Distribute CPU and interrupt affinities
- ✓ Modify TCP processing to improve parallelism

Implementation Detail

- **Reordering packets**
  - Reordering packets on stream merging
    - Add simple reordering mechanism before TCP processing
    - Avoid heavy load of reordering on TCP processing

Implementation Detail

- **Improve parallelism**

- Modify TCP processing
  - Parallelization of TCP receiving process
  - Separate application / TCP loads into different cores
- Distribute CPU and interrupt affinities
  - Distribute interrupt affinities of NICs
  - Fix processor affinity mask of application

Evaluation

- Compare Linux bonding and our optimized bonding
  - Round-robin mode

- Systems used

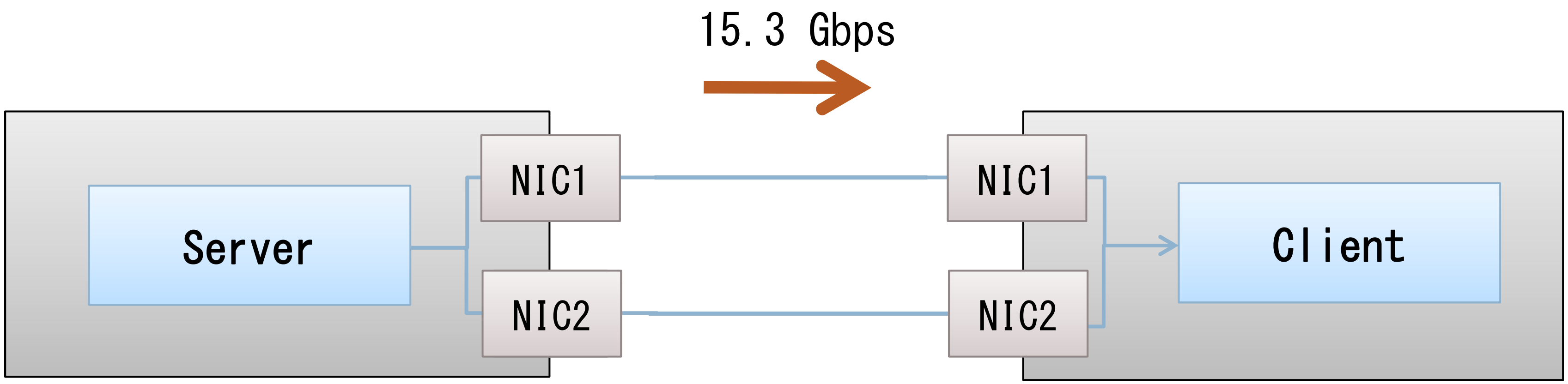
CPU	Intel Core i7 940
Motherboard	ASUS Rampage II GENE
Chipset	Intel X58 chipset
Memory	6 GB DDR3 SDRAM
NIC	Chelsio S310E-CR NIC *2
OS	CentOS 5.5 (kernel: linux-2.6.34.7)

Normal Linux bonding device

- Theoretical Peak Performance:19.82Gbps

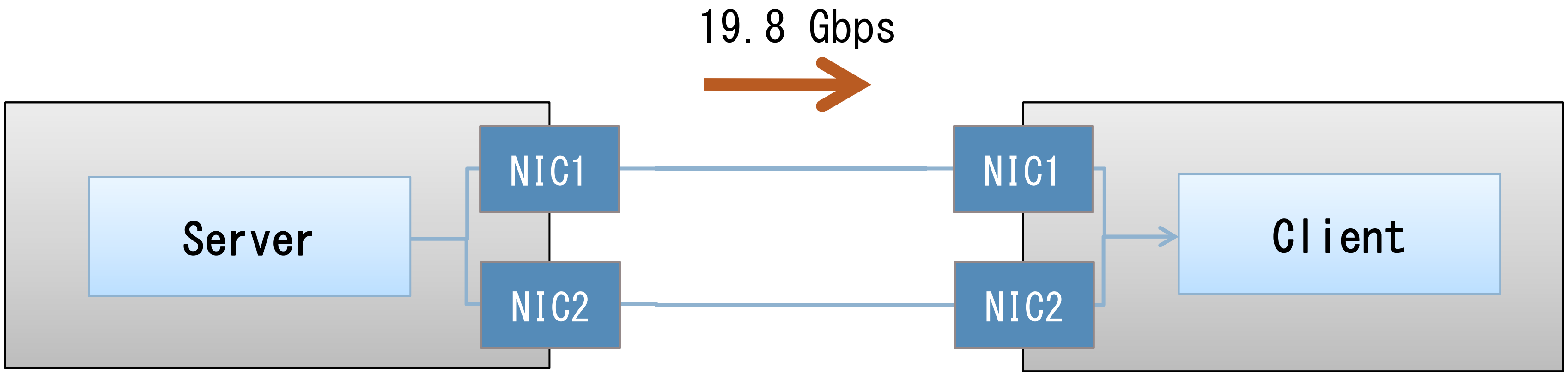
  ≒ 2 × 10Gbps × (9190−40) / (9190+26+16)

- Measured speed:15.3Gbps(77% of the peak)



Our optimized bonding device

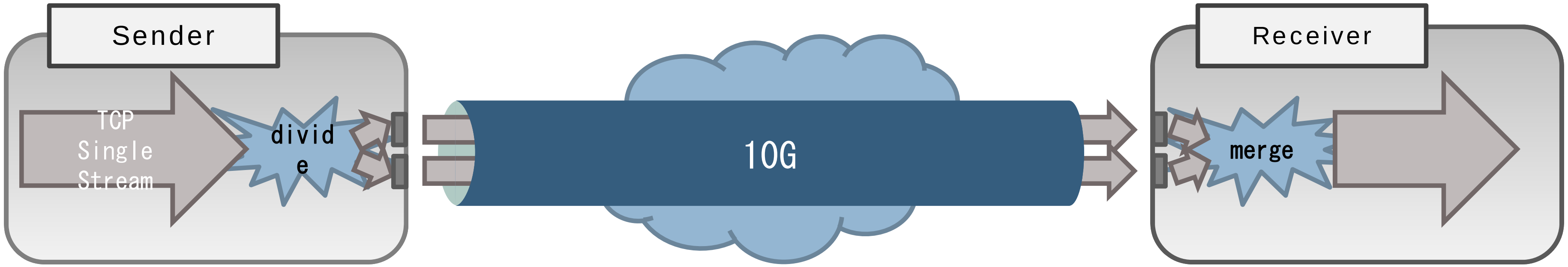
- Measured speed: 19.8Gbps(100% of Theoretical Peak)

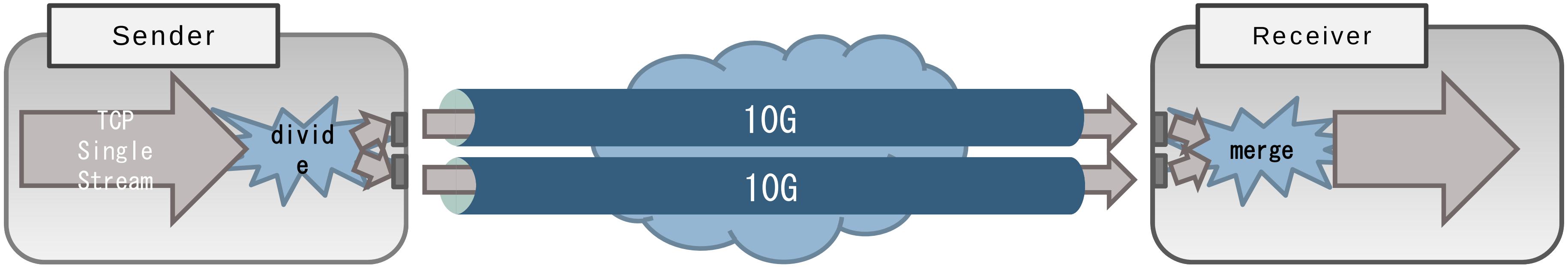


Experiments at SC10

- Performance measurements on actual LFN
  - Many complicated situation by actual LFN
  - Jitters, packet order change
  - Current performance (7~8 Gbps /w 1~1.5 Gbps back ground traffic in NLR FrameNet:10 Gbps)

- We are improving performance here


Sender
TCP
Single
Stream
divid
e
10G
Receiver
merge


Sender
TCP
Single
Stream
divid
e
10G
10G
Receiver
merge

Summary

- SC10 experiments are useful to find problems
- Optimized bonding is essential to get 40 Gbps single-stream TCP
- We thank JGN2plus and NICT for cooperation of SC10 experiments