Improving Server Efficiency by Optimizing TCP Pure ACK Receive Processing

Michael Chan and David Cheriton
mcfchan@stanford.edu, cheriton@stanford.edu

Motivations
- Servers today spend significant portion of cycles on TCP/IP stack
- Impact of future network technologies such as 40GE and 100GE
  - Servers handle more concurrent clients; bigger network processing burden on CPU
- Problems with current network stacks: Expensive packet buffer management, many layers of processing
- Many servers are bulk TCP data senders, receiving mostly pure TCP ACKs, e.g. Web servers, file repositories
- Pure ACK RX processing is simple; can be optimized to:
  1. Consume fewer cycles per ACK
  2. Improve ACK throughput

Optimizing pure ACK RX processing
- Detect pure TCP ACK by device and driver
- Parse TCP header fields
  - Seq #, ACK #, receiver window size, timestamps
- Deliver TCP header fields directly to TCP layer
- Reuse the same packet buffer for next receive operation

Implementation in Linux Kernel
- Removed all layers between driver and TCP ACK processing
  - LRO, network taps, IP, Netfilter, other TCP code
- Removed many memory management operations
  - No need to deallocate and reallocate packet buffers
- 40-line ACK header parser, 400-line TCP Pure ACK fast path

Evaluation
Two machines: Xeon Nehalem 2.66GHz, Intel 82574L Gigabit Lan-on-Motherboard, Neterion X3120 10GE NIC

Micro-benchmark
Flood a single core with 100K pure ACKs; bare minimum ACK processing time (ACKs do not cause TX of new segments):

Original: 1269ns ± 66ns
Optimized: 457ns ± 7ns

Bulk Data Transfer
- Send bulk data via 10GE NIC to 20 clients concurrently
- # CPU cycles saved
- Per-ACK processing time when TX is involved

![Graphs and Diagrams]

40% reduction in per-ACK processing time
14% fewer CPU cycles required for data transfer