Titan: Fair Packet Scheduling for Commodity Multiqueue NICs

Brent Stephens, Arjun Singhvi, Aditya Akella, and Mike Swift

July 13th, 2017
Ethernet line-rates are increasing!
Servers need:

To drive increasing line-rates

Low CPU utilization networking
Underlying mechanisms:

Segmentation Offload

Multiqueue NICs
TCP Segmentation Offload (TSO)

- Many operations performed by the OS are per-packet, not per-byte
- TSO allows the OS to send large segments to the NIC
- TSO NIC hardware generates packets from segments

Using large segments (64KB) instead of packets can reduce CPU load
Fairness Problems

TSO and multiqueue cause pervasive unfairness
Fairness is important

• Fairness is needed so competing applications can **share the network**

• Fairness is needed for **predictability**
  • Unfairness leads to unpredictable completion times across runs
  • Perfect fairness → perfect predictability

• Fairness can **improve application performance**
  • Ex: Weighted Coflow Scheduling
    • [Chowdhury SIGCOMM11, Chowdhury SIGCOMM14]
Titan Goals:

- Drive increasing line-rates 
- Low CPU utilization
- Per-flow fairness
- Work on commodity NICs
Multiqueue Fairness in Linux:

- Flow arrivals to each transmit queue are **dynamic**
- The OS **statically** uses a per-flow hash to assign flows to queues
- The NIC scheduler **statically** uses deficit round-robin (DRR) to provide per-queue fairness
- In the datacenter, the OS **statically** chooses a TSO size
Titan Design:

As flows dynamically arrive and complete, in Titan:

The OS dynamically:
- Assigns weights to flows
- Tracks the flow occupancy of queues
- Picks queues for flows
- Updates the NIC with queue weights

The NIC dynamically:
- Applies queue weights from the OS
Causes of Unfairness:

Multiqueue unfairness

TSO unfairness
Problem: Hash collisions
Problem: Hash collisions

Solution: Dynamic Queue Assignment (DQA)

• OS assigns a weight to each flow
• DQA picks the queue with the lowest occupancy when a flow starts
• Queue occupancies are updated:
  • Any time a flow starts enqueuing data
  • Any time a flow has no enqueued bytes (at most each TX interrupt)
Problem: Hash collisions

Solution: Dynamic Queue Assignment (DQA)
Problem: Asymmetric Oversubscription

F1 and F2 receive half throughput
Problem: Asymmetric Oversubscription

Solution: Dynamic Queue Weight Assignment (DQWA)

- OS assigns weights to flows
- OS updates the NIC scheduler with queue occupancies as flows start and stop (at most each TX interrupt)
- NIC updates DRR weights

This is implementable on existing commodity NICs because it only needs to update DRR weights!
Problem: Asymmetric Oversubscription

Solution: Dynamic Queue Weight Assignment (DQWA)

DQA and DQWA provide long-term fairness

This is implementable on existing commodity NICs because it only needs to update DRR weights!
Problem: TSO Unfairness

- Short-term unfairness can cause bursts of congestion in the network
- Short-term unfairness can increase latency

Wire Packet Scheduler

Short-term unfairness
Problem: TSO Unfairness

Solution: Dynamic Segmentation Offload Sizing (DSOS)

• DSOS dynamically changes the segment size during oversubscription
  • Same implementation as GSO
• CPU vs fairness tradeoff
  • Segmenting after the TCP/IP stack reduces CPU costs
Implementation

- DQA, DQWA, and DSOS are implemented in Linux 4.4.6
- Support for `ndo_set_tx_weight` is implemented in the Intel `ixgbe` driver for the Intel 82599 10Gbps NIC
- Titan is open source!

https://github.com/bestephe/titan
Evaluation

• Microbenchmarks
  • 2 servers, 1 switch
  • 8 queue NICs
  • Vary number of flows (level of oversubscription)

• Incremental fairness benefits of DQA, DQWA, and DSOS
  • DQA and DQWA: expected to improve long-term fairness
  • DSOS: expected to improve short-term fairness
Evaluation – Fairness Metric

Metrics:
- Normalized fairness metric (NFM) inspired by Shreedhar and Varghese:
  - NFM = 0 is fair
  - NFM > 1 is very unfair

\[
\text{NFM} = \frac{\text{Bytes}(\text{MaxFlow}) - \text{Bytes}(\text{MinFlow})}{\text{Bytes}(\text{FairShair})}
\]
Microbenchmarks – 1s Timescale

• Linux is unfair at all subscription levels
• DQA often significantly improves fairness
  • At 48 flows, flow churn prevents DQA from evenly spreading flows
• DQWA improves fairness when DQA cannot evenly spread flows across queues
• DSOS does not have a significant impact on long-term fairness
Microbenchmarks – 1ms Timescale

- At short timescales and under oversubscription, DQA and DQWA do not significantly improve fairness
  - TSO is the primary cause of unfairness

- DSOS (16KB) often reduces unfairness by >2x
Cluster Experiments

CDF of completion times in a 1GB all-to-all shuffle (24 servers)

- Ideal CDF would be a vertical line
- Titan makes performance more predictable
- Titan improves tail performance (>90\textsuperscript{th} percentile)

Titan improves fairness without changing the network core!
Additional Evaluation

Additional performance metrics:
- Throughput: line-rate
- Latency: no significant change
- CPU Utilization:
  - DQA and DQWA: increase < 10%
  - DSOS is better than statically decreasing the TSO size
    - DSOS motivates creating a better TSO implementation (zero-copy)

Linux network configuration trade-off study
- See paper
Summary

• Multi queue NICs can lead to significant flow-level unfairness
  • Titan significantly improves fairness by allowing the OS to *dynamically* interact with the NIC packet scheduler
  • Titan is implementable on commodity NICs!

https://github.com/bestephe/titan