OVERLOAD CONTROL
FOR MS-SCALE RPCS WITH BREAKWATER

Inho Cho, Ahmed Saeed, Joshua Fried, Seo Jin Park, Mohammad Alizadeh,
and Adam Belay, MIT CSAIL

in *Proc. of USENIX,* 2020

NSLab Seminar

Ga Eun Seo

Networking and Systems Lab.

20211133@sungshin.ac.kr

January 26, 2023

Contents

- Introduction
- Motivation
- Design
- Implementation and Testbed Setup
- Evaluation
- Conclusion

Introduction

- The most important goals for data center network (DCN) transport designs is to minimize the flow completion times (FCT)
  - Many of today's cloud applications have very demanding latency requirements
  - A small delay can directly affect application performance and degrade user experience
- To minimize FCT, most recent proposals assume prior knowledge of accurate per-flow information
  - In this paper, point out that this information is difficult to obtain and use in many applications
  - Existing transport layer solutions with this assumption are difficult to implement.

Introduction (Cont'd)

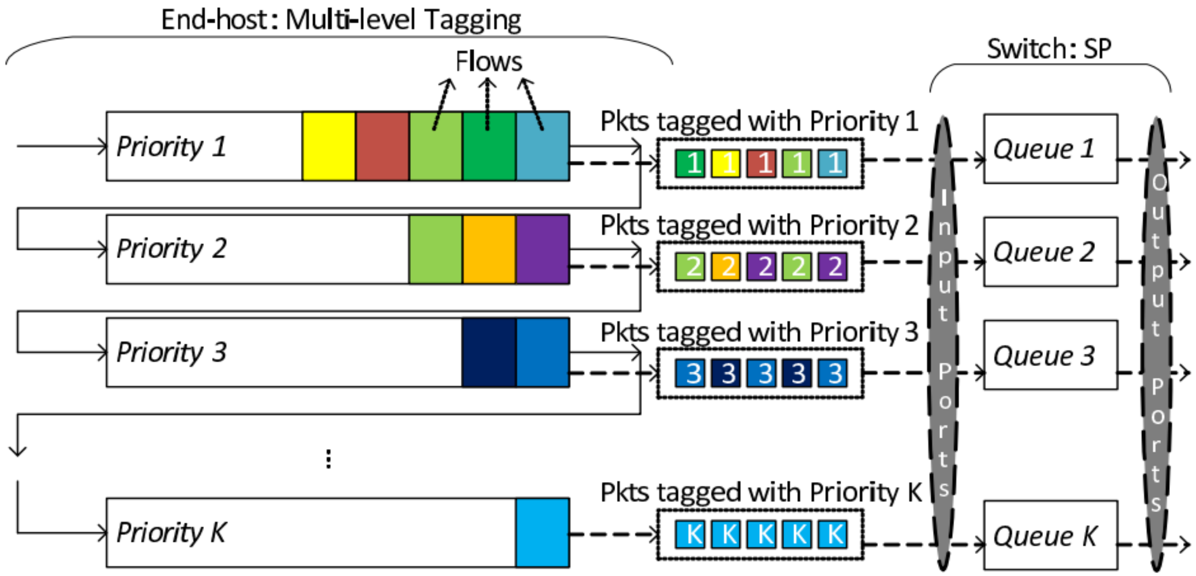
- Design goal for the best scheme to minimize FCT with existing commodity switches without prior knowledge of flow size information
  - Information-agnostic
    - Design should not assume prior knowledge of flow size information available to the application
  - FCT minization
    - The solution should be able to implement optimal flow scheduling regardless of information
    - Minimize the average and tail FCT of short flows, which are sensitive to latency, while not adversely affecting the FCT of long flows
  - Readily-deployable
    - Soultion should work with DCN's existing commodity switches and be backward compatible with legacy TCP/IP stacks
- In this paper, we create PIAS, an information-agnostic (information independent) flow scheduling that minimizes FCT in DCN
  - PIAS implements MLFQ (Multiple Level Feedback Queue) by utilizing multiple priority queues that can be used in existing commodity switches.

Motivation (Cont'd)

- HTTP chunked transfer
  - Chunked transfer has been supported since HTTP 1.1, where dynamically generated content is transferred during the generation process
  - A flow generally consists of multiple chunks, and the total flow size is not available at the start of the transmission
- Database query response
  - SQL servers send partial query results as they are created, instead of buffering the result until the end of the query execution process
  - The flow size again is not available at the start of a flow
- Stream processing
  - During the data processing, data tuples completed in one node are continuously delivered to the next node in the stream processing chain
    - The amount of data to be processed is unknown until the stream finishes
- Practical limitations
  - Certain cases where flow size information can be obtained or inferred, but practical implementation issues are still prohibited.
    - need to patch all modules in every application that generate network traffic
    - current operating systems lack appropriate interface for delivering the flow size information to the transport layer

Design

- PIAS utilizes multiple priority queues available on commodity switches and implements MLFQ
  - The PIAS flow is dynamically demoted from a high-priority queue to a low-priority queue according to the bytes transferred
    - PIAS allows short flows to complete in the first few priority queues, so it generally prioritizes longer flows, effectively mimicking SJF without knowing the flow size
  - By implementing the MLFQ, PIAS has two advantages
    - Prioritize shorter flows over larger flows
    - Allow large flows to be demoted to the same low priority queue to fairly share the link



PIAS Overview

Design(Cont'd)

◆ Design Challenge

- How to determine the demotion threshold for each queue in the MLFQ to minimize FCT
  - PIAS calculates thresholds based on the traffic information of the entire DCN and distributes the same threshold settings to all end hosts
  - By solving the FCT minimization problem,
    Derive a series of analytical solutions for the optimal demotion threshold
- DCN traffic is time and space dependent,
  How can PIAS operate efficiently and reliably in such a dynamic environment
  - Using ECN
  - For PIAS to work on highly dynamic DCNs, mitigate the effects of inconsistency
- How to ensure PIAS compatibility with legacy TCP/IP stacks in production DCNs
  - PIAS effectively eliminates TCP timeouts by minimizing the response time of each long flow

Design(Cont'd)

- Switch Design
  - Priority Scheduling
    - Packets are excluded from the queue strictly according to priority when fabric ports are idle
  - ECN marking
    - An arriving packet is marked as CE (Congestion Experienced)
      if the immediate buffer occupancy is greater than the marking threshold

Design(Cont'd)

- Perform MLFQ-based flow scheduling
  - Packets with different priority tages are sorted into different priority queues
  - When the link is idle, head packets from the highest non-empty priority queue are sent
- Select priority scheduling
  - Priority queuing specification provides better innetwork prioritization specification and potentially achieves lower FCT than WFQ
  - WFQ degrades TCP performance by causing packet ordering problems
- Using ECN is to mitigate the effect of mismatch between demotion threshold and traffic distribution

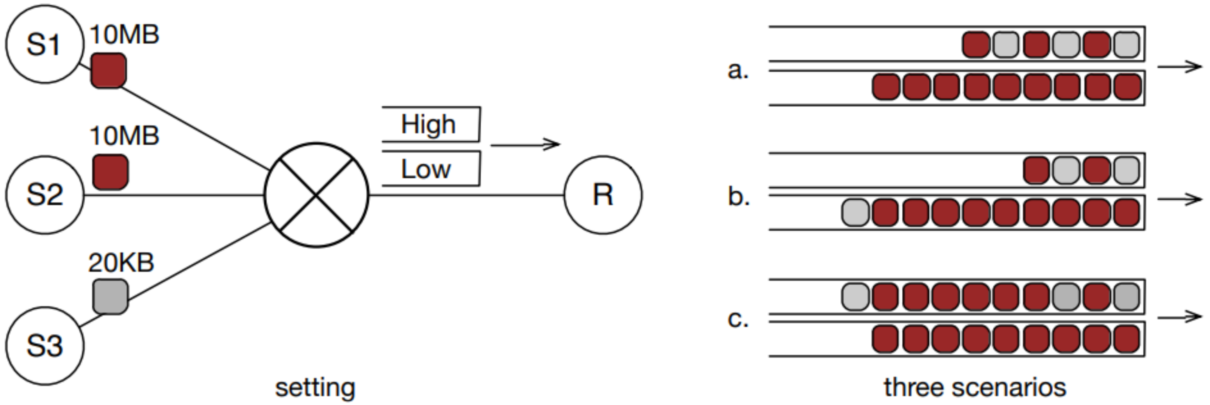
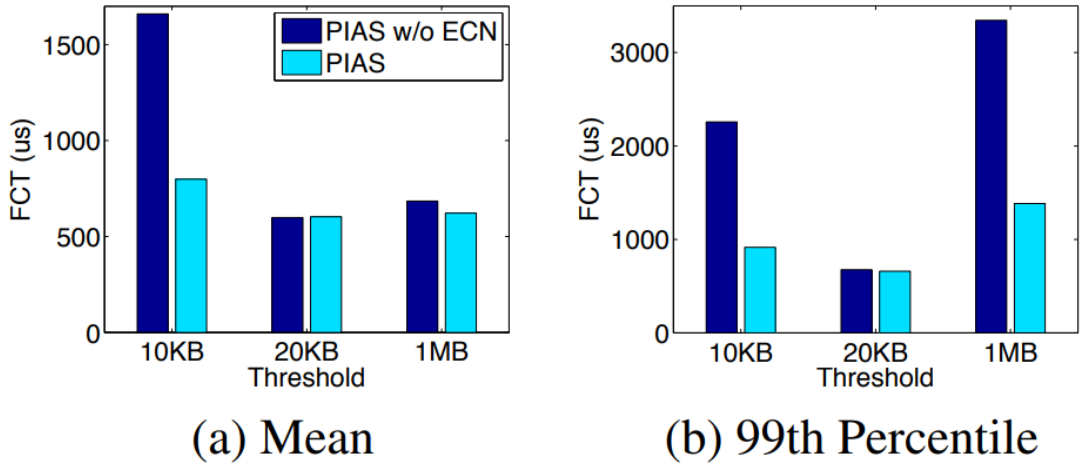


Illustration example



Completion time of 20KB short flows

Design(Cont'd)

- Rate Control
  - To deal with flow depletion
    - Many concurrent short flows starve long flows, causing TCP timeouts
  - TCP timeouts are mainly caused by starvation, not packet drops
  - PIAS effectively eliminates TCP timeouts by minimizing the response time of each long flow
    - Note that PIAS can alleviate starvation between long streams
      - Because two long flows in the same low-priority queue share the link fairly in a FIFO fashion
  - PIAS does not require packet reordering

Design(Cont'd)

- The core idea of PIAS is to emulate SJF
  - optimal for minimizing the average FCT on a single link
  - Make a local decision-based solution
    - This approach may incur some performance loss to the fabric
    - Local decision-based solutions maintain very good performance in most scenarios, only experiencing performance loss at very high loads (e.g. 90% or more)
    - ns-2 simulation with production DCN traffic shows that PIAS works well in practice

- Demotion threshold updating
  - PIAS uses ECN to effectively handle discrepancies between demotion thresholds and traffic distribution
  - No need to frequently change demotion threshold which can be overhead

Implementation and Testbed Setup

◆ Packet Tagging

- The packet tagging module maintains per-flow state and marks packets as prioritized at the end host
- Implemented as a kernel module in Linux
- The TCP/IP stack and Linux TC and consists of three components: a NETFILTER hook, a hash-based flow table, and a packet modifier
  - The NETFILTER hook intercepts all outgoing packets using the LOCAL OUT hook and sends them to the flow table
  - Each flow in the flow table is identified by a 5-tuple (src/dst IP, src/dst port and protocol)
  - The packet modifier sets the priority of the packet by modifying the DSCP field of the IP header to the corresponding value

- Offloading techniques such as Large Segmentation Offloading (LSO) reduce the accuracy of packet tagging
  - A final implementation solution must reside in the NIC hardware to permanently avoid this interference

Implementation and Testbed Setup (Cont'd)

- Switch Configuration
  - Apply strict priority queuing and classify packets at the switch
    - Use ECN marking based on instant queue length with a single marking threshold
  - latency
    - Besides switch queuing delays in the network, the sender NIC is the first point of contention in the fabric.
  - Hardware and software queuing on end hosts causes large queuing delays, severely degrading application performance
    - POST ROUTING performs ECN marking and priority queuing on hosts and switches
  - ECN marking for each port
    - Inability to provide ideal isolation between queues, such as per-queue ECN marking
    - Provides much better burst tolerance and supports a larger number of queues in shallow buffer switches
    - If many packets of low priority flows are queued at the switch, it can push higher priority flows back

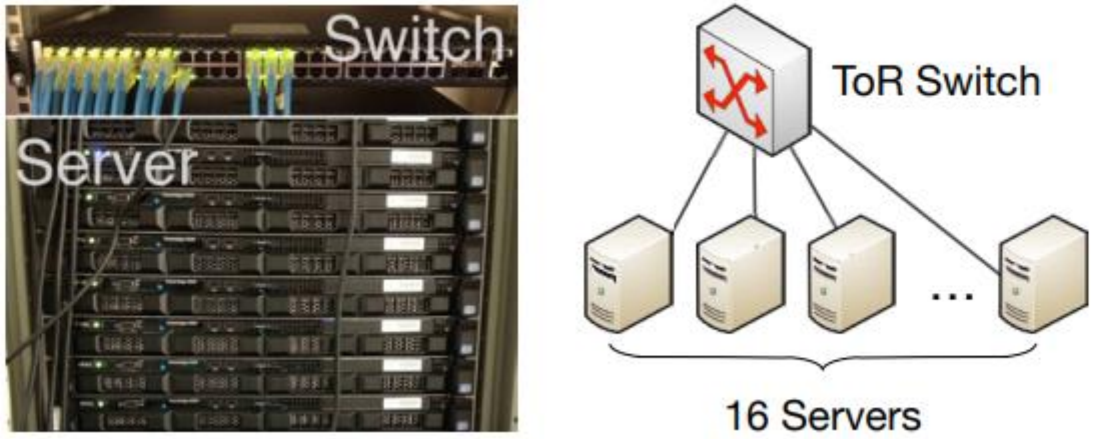
Implementation and Testbed Setup (Cont'd)

- Rate Control
  - Using open source DCTCP patch for Linux 2.6.38.3
  - Observe undesirable interactions between open-source DCTCP implementations and switches
  - If the instant queue length is greater than the ECN mark threshold, the switch drops non-ECT packets from the ECN-enabled queue.
    - Severely degrades TCP performance
    - Set ECT on all TCP packets in the packet modifier

Implementation and Testbed Setup (Cont'd)

◆ Testbed Setup

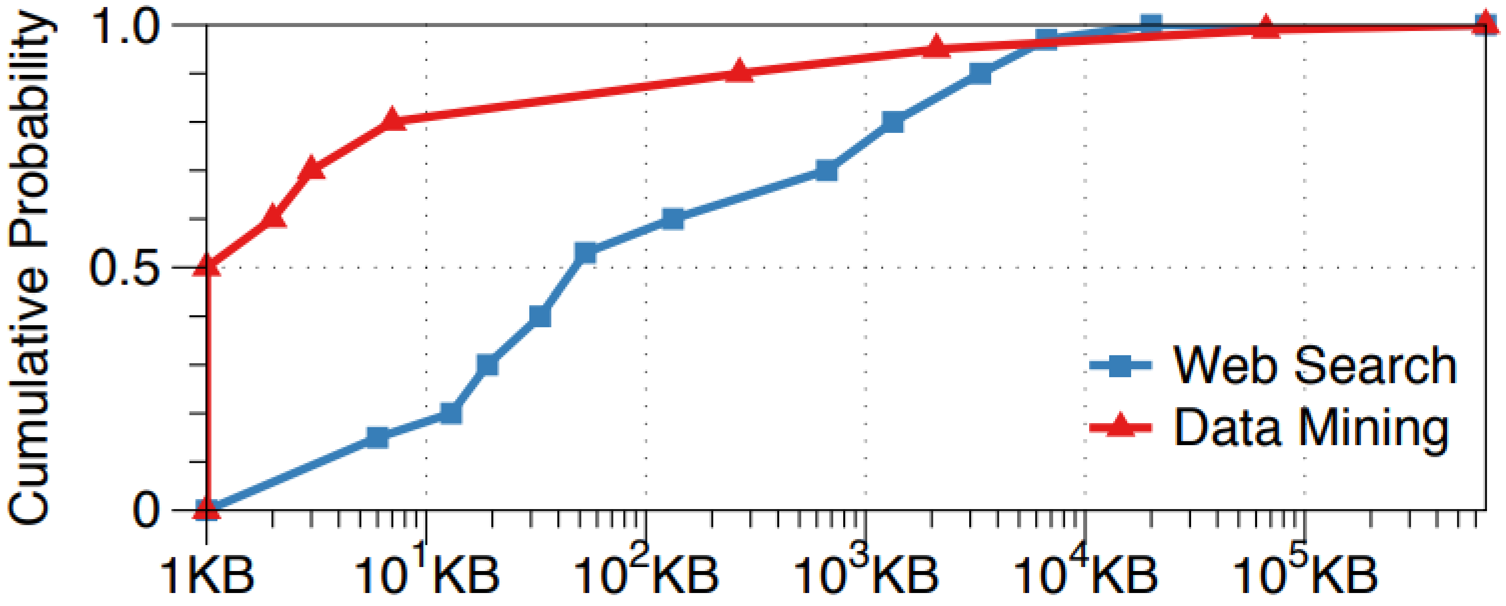
- Build a small testbed consisting of 16 servers connected to a Pronto 3295 48-port Gigabit Ethernet switch with 4 MB shared memory as shown
  - Each server is a Dell PowerEdge R320 with 4-core Intel E5-1410 2.8GHz CPU, 8G memory, 500GB hard disk and Broadcom BCM5719 NetXtreme Gigabit Ethernet NIC
- Switches support ECN and strict priority queuing with up to 8 service queue classes
- Running Debian 6.0-64 bit with Linux 2.6.38.3 kernel
- Also built a smaller 10G testbed to measure end-host queuing latency on high-speed networks.



Testbed Topology

Evaluation

- ◆ Setting
  - PIAS uses 8 priority queues by default, enabling ECN marking per port
  - Set ECN display threshold to 30 KB as recommended by DCTCP
  - Use two realistic workloads
    - web search workload
    - Data mining workloads in production data centers
  - Evaluate PIAS using Memcached's Application Benchmark



Traffic distributions used for evaluation

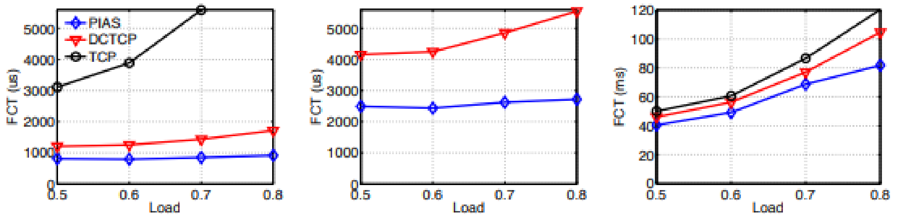
Evaluation(Cont'd)

- Results with realistic workloads
  - Generate dynamic traffic based on actual workload
  - Developing a client/server model for measuring FCT at the application layer
    - A client application running on one computer periodically makes a request to another computer to get data
    - The server application running on 15 different machines responds with the requested data
  - Evaluate the performance of PIAS, DCTCP and TCP while varying the network load from 0.5 to 0.8
  - If the DCN has a moderate average traffic load (e.g. 30% [16])
    - Unlikely to have a long-term load above 80% in practice.

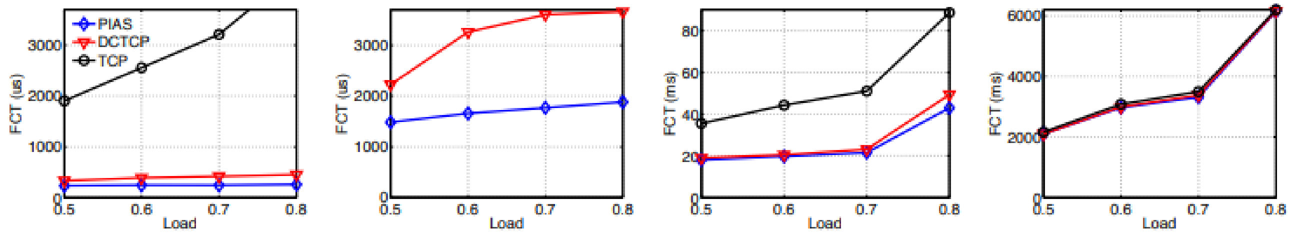
Evaluation(Cont'd)

◆ Observations

- PIAS achieved the best performance in both the mean and 99th percentile FCT of small flows.
  - Compared to DCTCP, PIAS reduces the average FCT of small flows by ⇠37-47% for web browsing workloads and ⇠30-45% for data mining workloads.
  - The improvement of PIAS over DCTCP in short-flow 99th percentile FCT is even greater: ⇠40-51% for web browsing workloads and ⇠33-48% for data mining workloads.
- PIAS provides the best performance even in medium flow
  - Up to 22% lower mid-flow average FCT than DCTCP for web browsing workloads
- PIAS does not severely penalize large flows
  - Because PIAS prioritizes short flows over long flows, and ⇠60% of all bytes in the web browsing workload come from flows less than 10 MB
  - Since data center workloads are dominated by small to medium flows, performance gap does not affect the overall average FCT



(a) (0,100KB]: Avg (b) (0,100KB]: 99th Percentile (c) (100KB,10MB]: Avg (d) (10MB,∞): Avg



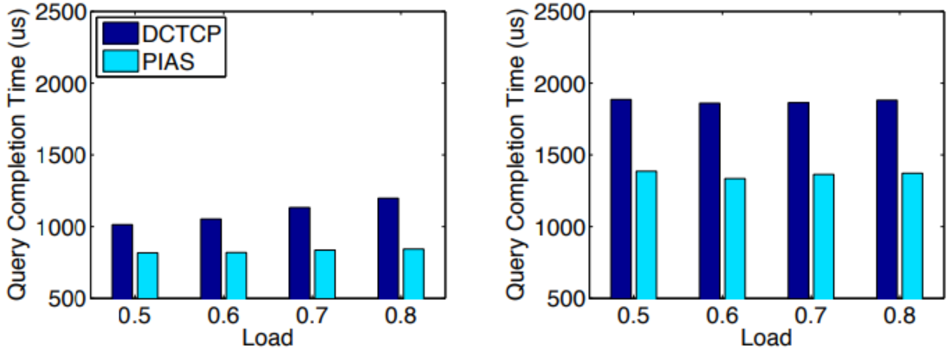
Web search workload and Data mining workload

Evaluation(Cont'd)

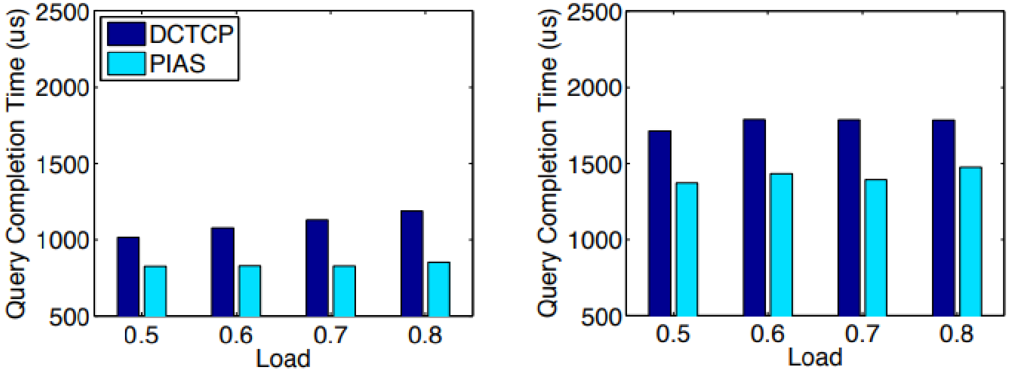
- Results with the Memcached application
  - Deploy Memcached cluster with 16 machines
    - To evaluate how PIAS improves the performance of latency-sensitive applications
    - One machine is used as a client and the other 15 machines are used as servers to emulate partition/aggregate soft real-time services
    - Client sends a GET query to all 15 servers, each responding with a 1KB value
    - The query completes only when the client receives all responses from the server
    - Measure query completion time as an application performance metric
  - Basic query completion time is about 650us on our testbed
  - Based on the distribution of the web browsing workload,
    generate background traffic with a mixture of mouse and elephant flows

Evaluation(Cont'd)

- Because the switch enables both dynamic buffer management and ECN, none of the queries experience TCP timeouts
  - As the background traffic load increases, DCTCP's average query completion time also increases (1016–1189 us at 40 qps and 1014–1198 us at 20 qps)
- PIAS maintains relatively stable performance
  - At 0.8 load, PIAS achieves ⇠28-30% lower average query completion time than DCTCP
  - PIAS reduced 99th percentile query completion time by ⇠20-27%
- In summary, PIAS effectively improves the performance of Memcached applications by reducing the latency of short flows



(a) Mean (b) 99th Percentile

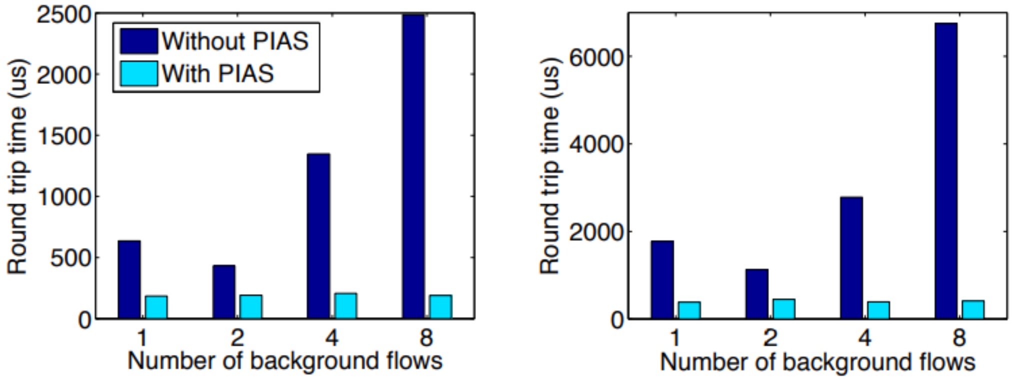


(a) Mean (b) 99th Percentile

Query completion time at 20 qps and Query completion time at 40 qps

Evaluation(Cont'd)

- End host queuing delay
  - Mainly focused on network switching nodes
  - Without PIAS, ping packets experience latency delay of up to 6748us with 8 background flows.
  - Measure RTT using PIAS and compare with results without PIAS
  - The PIAS scheduling module does not affect network utilization, and the large-scale flow still maintains a goodput of more than 9 Gbps during the experiment
    - Using LSOs to reduce CPU overhead makes fine-grained transmit control difficult to achieve, and some delays still exist in the NIC's transmit queue
      - can be improved by offloading scheduling to NIC hardware



(a) Mean (b) 99th Percentile

RTT with background flows

Conclusion

- Through PIAS, leverage existing commodity switches in DCNs to minimize the average FCT for flows
  - especially the smaller ones, without assuming any prior knowledge of flow sizes

- Implemented a PIAS prototype using all commodity hardware and evaluated PIAS through a series of small-scale testbed experiments as well as large-scale packet-level ns-2 simulations
- Both our implementation and evaluation results show that PIAS is a viable solution that achieves all design goals