# A Cost-efficient Scheme with Decoupling Host-side Flow Scheduling from Switches in DCNs

Weibin Xie, Fang Wang, Yu Hua, Dan Feng, Yuchong Hu, Chu Li

*Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System (School of Computer Science and Technology, Huazhong University of Science and Technology), Ministry of Education of China*
*Corresponding author: wangfang@hust.edu.cn*

***Abstract*—Cloud datacenters use limited network resources to host complex and diverse applications that requires us to treat upper applications as a black box and provide low latency for latency-sensitive short flows. Many good performance schemes need to preliminarily obtain flow information (flow size, deadline or traffic distribution) or modify hardware or software, which leads to difficult use and inefficient in practice.**

**To solve the dilemma, we present SPQ, an information-agnostic and readily-deployable flow scheduling scheme to provide near-optimal flow completion times (FCT) for latency-sensitive short flows and deal with long-tailed distribution effectively. SPQ decouples host-side flow scheduling from switch-side flow scheduling to achieve broad practical scenes with approximating the Least Attained Service (LAS) scheduling discipline. In this way, SPQ can achieve near-optimal scheduling order at end-hosts. Meanwhile, we utilize two novel feedback adjustment mechanisms to alleviate the effect of long flows on short flows.**

**Our simulations show that SPQ achieves outstanding performance and broad practical scenes. Compared with DCTCP, L2DCT and PIAS, SPQ achieves obvious advantages under three workloads, and for example, it reduces the average FCT of short flows by up to 56%, 55% and 8% respectively, and reduces the 99th percentile FCT by up to 67%, 69% and 24% respectively. And the average FCT of short flows for SPQ only have a 0-1.1% gap to the ideal information-aware scheme under data mining workload.**

## I. Introduction

More and more applications migrate to cloud datacenters such as web search, Online Data-Intensive (OLDI), social networking, recommendation systems, and advertisement systems, etc [12]. These applications generate a mass of latency-sensitive short flows mixed with long background flows, with indeterminate and variable flow sizes and deadlines [2], [16]. They need low latency for these short flows in order to satisfy user demands. Even a very small delay can make a large impact on application performance, then degrading user experience and the revenue [5], [12]. Moreover, many applications don't know how many bytes of network flow will produce [2], [12]. So we have to treat the upper applications as a black box. Meanwhile, all of flows must compete for limited network resources, which commonly causes packet losses and timeout retransmissions. With packet losses and timeout retransmissions, many flows' FCT will increase to two orders of magnitude, forming a long-tailed distribution [19]. It will increase latency-sensitive short flows' FCT.

However, many recent DCN flow scheduling schemes go to two extremes: The information-aware schemes [4], [10], [11], [14], [15], [18] one-sided pursue good performance, but ignore the practicality. These schemes need preliminarily to obtain flow information or modify hardware or software. They usually can obtain better performance than other schemes. However, it is difficult to implement them in practice. In these schemes, the in-network priority scheduling schemes (such as pFabric [4] and PIAS [5]) generally achieve more excellent performance. On the other hand, although the information-agnostic schemes [2], [3], [7], [12], [17] achieve broad practical scenes, they only obtain limited performance.

In view of the above situations, there are some challenges to overcome to this dilemma:

1) **A real information-agnostic solution:** The solution must treat upper applications as a black box, and obtain very wide practical scenarios. The solution doesn't need preliminarily any information including traffic distribution, and the change of workloads should not increase the FCT of latency-sensitive short flows.
2) **Near-optimal FCT for short flows:** Compared with the ideal information-aware scheme (pFabric), the solution must achieve analogous FCT for latency-sensitive short flows.
3) **Dealing with long-tailed distribution effectively:** The solution must make sure the excellent performance for the 99% of the latency-sensitive short flows [18], [19].
4) **Friendly-deployment:** The solution doesn't need to modify the hardware and datacenter applications, and should be compatible with the existing datacenter transmission protocols.

In this paper, we present SPQ, an information-agnostic and readily-deployable flow scheduling scheme to provide near-optimal FCT for latency-sensitive short flows and deal with long-tailed distribution effectively. Unlike those in-network priority scheduling schemes, such as pFabric and PIAS which use the same flow scheduling method at host-side and switch-side, SPQ decouples the host-side flow scheduling from switch-side flow scheduling to approximate the LAS scheduling discipline, and utilizes L2DCT [12] protocol and Explicit Congestion Notification (ECN) [8] to provide rate control and loss recovery. Meanwhile, SPQ provides two novel feedback adjustment mechanisms to alleviate the effect of long flows on short flows. It utilizes feedbacks of ECN and timeout retransmission to infer the fabric conditions and the effect of

978-1-5386-2704-4/17/$31.00 ©2017 IEEE

long flows on short flows, using to decide how much effort must be used to adjust the rate of long flows.

Compared with pFabric and PIAS that generally divide all flows into 8 different priorities, we provide more fine-grained priorities to better distinguish different bytes of flows at end-hosts. And the priorities are based on the bytes that each flow has already sent to the fabric, so that we can achieve the approximately theoretically optimal scheduling order at end-hosts. At switches, we utilize shallow buffer and First-In-First-Out (FIFO) scheduling policy. In this way, we can make full use of the host-side flexibility of dealing with flows to achieve the approximately theoretically optimal scheduling order, and avoid these restrictions of in-network flow scheduling schemes at switches. So SPQ can achieve outstanding performance and broad practical scenes without any restrictions.

The key contributions of this paper are as follows:

- We analyze the limitations of in-network priority scheduling schemes and their very narrow practical scenes, and we explain the root cause of these limitations. We also illustrate what kind of flow scheduling scheme is really needed in datacenters.
- We design SPQ, which achieves outstanding performance and broad practical scenes with two key innovations: decoupling host-side flow scheduling from switch-side flow scheduling and two novel feedback adjustment mechanisms.
- We evaluate SPQ using the ns2 [13] simulator, comparing with two information-agnostic schemes (DCTCP, L2DCT), a semi-information-agnostic scheme (PIAS), and the ideal information-aware scheme (pFabric).

We evaluate SPQ using three realistic workloads. Our results show that SPQ obtains outstanding performance and broad practical scenes without any restrictions, and achieves the objective of this paper: providing near-optimal FCT for latency-sensitive short flows and dealing with long-tailed distribution effectively. For example, under hybrid workload, the average FCT of short flows for SPQ only have a 0-3.5% gap to the ideal information-aware scheme (pFabric), and this gap is within 0-1.1% under data mining workload. Furthermore, compared with DCTCP, L2DCT and PIAS, SPQ achieves obvious advantages for all flow sizes and loads under three workloads. It reduces the average FCT of short flows by up to 56%, 55% and 8% respectively, and reduces the 99th percentile FCT by up to 67%, 69% and 24% respectively.

## II. Motivation

There're many current situations motivating us to design a real information-agnostic flow scheduling scheme with outstanding performance and broad practical scenes:

- The complex datacenter environment requires us treat upper applications as a black box. There're many applications whose flow size or deadline information is hard to preliminarily obtain, such as HTTP chunked transfer , database query response, stream processing, and etc [5].
- Its very difficult and costly to dynamically transmit the applications flow information to the transport layer. Because current operating systems lack appropriate interface to do this work [5], [6].
- Although PIAS is good attempt to provide a appropriate solution, PIAS doesn't completely solve the dilemma.

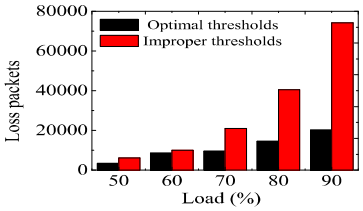



Fig. 2: Under web search workload, the loss packets of PIAS at varying loads with different demotion thresholds.

## III. Design

### A. Design Rationale

There are three well-known limitations about the in-network priority scheduling schemes such as pFabric and PIAS. The first one is that they try to divide diverse and variable flows with a set of static demotion thresholds. It makes them have very narrow practical scenarios and hard to provide steady performance for practical applications, since the datacenter applications are diverse and complex [4], [16]. It's impracticable to dynamically calculate these thresholds to provide a set of optimal thresholds when the datacenter applications are changing. If there's mismatch between demotion thresholds and traffic, the performance will fall. Figure 1 shows the process of PIAS's performance declines when workloads change: PIAS sets the optimal thresholds for the workloads of left figure, when workloads change, it causes the mismatch between thresholds and workloads. This case can seriously weaken the performance of PIAS. The right figure shows that the latency-sensitive short flows fall into the lowest priority queue, and wait behind the long flows. This situation also exits in other in-network priority scheduling schemes.

We also give a very simple simulation example about PIAS to illustrate this problem. In this example, we run a web search workload to count the number of loss packet on NS2 simulator, which has 100000 flows with different sizes. And we use two series of different demotion thresholds: one is the optimal thresholds of web search workload, and another is the optimal threshold of data mining workload. With the improper thresholds, figure 2 shows that the number of loss packets rose sharply. A large number of packet loss will inevitably lead to serious bandwidth loss and timeout retransmissions. Meanwhile, both bandwidth loss and timeout retransmissions can cause severe performance degradation.

The second one is that the number of priority queues available in existing commodity switches are limited, usually 4-10 queues [11], [18]. The in-network priority scheduling schemes generally mark all flows with 8 priorities at end-hosts, and use these different priorities to decide which packet to schedule or drop [4], [11]. This method tries to use a simple program to distinguish between short flows and long flows. However, this method can't strictly distinguish different bytes

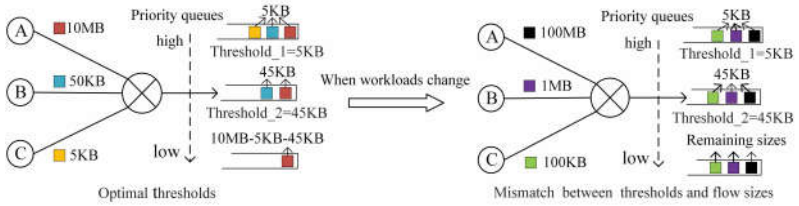



Fig. 1: Simple principle illustrating that the performance decline of in-network prioritization (PIAS) causing by mismatch between thresholds and workloads. To simplify the description, we only use three priority queues.

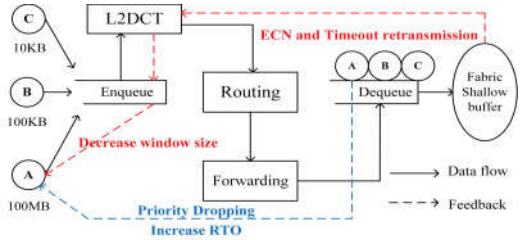



Fig. 3: The schematic diagram of SPQ

of flows, unless the number of priority queues available in existing commodity switches are infinite.

The last well-known limitation is that the in-network prioritization means that switches make local decisions, which can easily lead to upstream bandwidth loss, especially in multi-link scenarios [4], [11].

The root cause of these well-known limitations is that they want to provide the same flow scheduling scheme at end-hosts with switches. However, the flexibility of dealing with flows has huge difference between end-hosts and switches. We can implement a lot of fine operations to shape flows at end-hosts without any limitations, but we can't do this at existing commodity switches. In fact, the in-network priority scheduling schemes sacrifice the flexibility and performance of end-hosts, in exchange for the priority scheduling at switches. This method limits their performance and practical scenarios in return. However, if we decouple host-side flow scheduling from switches, we can avoid these restrictions to obtain outstanding performance and wide practical scenarios.

### *B. The Detailed Design of SPQ*

*1) Packet Scheduling:* SPQ decouples the host-side flow scheduling from switch-side flow scheduling to approximate the LAS scheduling discipline, and utilizes L2DCT and ECN to provide rate control and loss recovery (Figure 3). At end-hosts, SPQ flags all of flows with different priorities based on the amount of bytes that each flow has sent to the fabric. In this way, SPQ can provide fine-grained priorities to decide which packet to schedule or drop. Due to the high flexibility of dealing with flows at end-hosts and the efficient scheduling discipline, SPQ can achieve the approximately theoretically optimal scheduling order at end-hosts. There're two cases leading to the dropout of packets that belong to long flows: in the first case, when the queue of NIC is fully occupied, SPQ will drop the packet with the lowest priority. This case is normally flow priority scheduling at the queue of NIC, regardless of the state of fabric congestion. In the second case, fabric congestion worsens and long flows seriously restrict the rate of short flows, which leads to the packets that belong to short flows can't be retransmitted straightway. SPQ drops the packets with the lowest priority until the fabric congestion is mitigated, regardless of whether the queue of NIC is fully occupied.

At switches, SPQ uses shallow buffer and First-In-First-Out (FIFO) policy to avoid these well-known limitations of in-network priority schemes. In this way, SPQ can achieve wide practical scenes without any limitations. SPQ utilizes shallow buffer to accelerate the rotation of packets, alleviating the effect of latency-insensitive long flows on latency-sensitive short flows [2], [11]. At the same time, SPQ makes use of the feedbacks of ECN and timeout retransmission to adjust rate of long flows. SPQ can use these feedbacks to infer the degree of network congestion , and decide how much effort must be used to adjust the rate of long flows at end-hosts. We provide two feedback adjustment mechanisms to adjust the rate of long flows: approximately theoretically optimal priority scheduling at end-hosts (the blue dotted line in figure 3), and rate control based on the feedbacks of ECN and timeout retransmission (the red dotted line in figure 3).

In short, SPQ always ensures that the less bytes of each flow has sent into fabric, the higher priority it has at the end-hosts. If long flows affect the rate of latency-sensitive short flows in fabric, we make use of two feedback adjustment mechanisms to adjust the rate of long flows.

*2) Rate Control:* We have two feedback mechanisms to provide rate control: fine-grained priority scheduling at end-hosts and rate control based on the feedbacks of ECN and timeout retransmission. Before explaining the principle, we first discuss when congestion and packet loss take place. The first case: there're too many long flows occupying the queues of switches. The second case: the link is fully utilized [11]. We must utilize different mechanisms to cope with the two different cases, because they imply different fabric conditions. We use feedbacks of ECN and timeout retransmission to infer the fabric conditions.

When an ACK packet which is set ECN-Echo flag is received, and the packets which don't receive the ECN mark don't need to be retransmitted. It suggests the fabric isn't fully utilized, but there're too many long flows competing bandwidth with short flows. So we must reduce the rate of long flows. SPQ utilizes L2DCT [12] protocol to adjust their rate. In this case, these long flows' window size are set as:

$$cwnd = cwnd * (1 - b/2)$$

When no packets are marked with ECN, these packets window size are set as:

$$cwnd = cwnd + k$$

Where b is the backoff penalty, k is the increment in congestion window per RTT [12]. How to calculate the value of b and k is beyond the scope of this paper, we can refer to the references [12] to find the method.

The another case is that the link is almost fully utilized, leading to packets that belong to short flows can't be transmitted straightway. If an ACK packet which is set ECN-Echo flag is received, and the packets which don't receive the ECN mark need to be retransmitted. It suggest that the fabric is almost fully occupied, and the long flows strictly restrict the rate of short flows. We should adopt more strict method to restrict the rate of long flows. On the basis of L2DCT transfer protocol (These long flows' window size are still set as: $cwnd = cwnd * (1 - b/2)$), we use the another feedback mechanism to adjust the rate of long flows: fine-grained priority dropping at end-hosts. In this case, we drop the lowest priority packets which belong to long flows at the queue of NIC, until the degree of congestion relieves to the first case. When the degree of congestion fall, we only use L2DCT transfer protocol to adjust the rate of long flows. Otherwise, in the whole process, we use the L2DCT protocol to make the short flows keep the normal rate.

In short, we make use of feedbacks of ECN and timeout retransmission to infer the fabric conditions and the effect of long flows on short flows, using to decide how much effort must be used to adjust the rate of long flows.

*3) Loss Recovery:* SPQ uses different timeout retransmission values based on the packet priorities and fabric condition. At the beginning, all flows have the same value of RTO . When an ACK packet which is set ECN-Echo flag is received, and there're packets needed to be retransmitted, we increase the long flows' value of RTO.

## IV. Implementation

We implement SPQ on large scale ns-2 simulations.

### A. End-host Implementation

We implement L2DCT transport protocol and the approximately theoretically optimal scheduling mechanism on NS2. Contrasted the Linux kernel modules, the packet priority scheduling mechanism is similar to a Linux kernel module built on top of L2DCT protocol (see the figure 3). We encode all packets' priorities in the IP header based on the bytes that each flow has sent into fabric.

TABLE I: Main simulation parameters setting

| Scheme | Parameter |
|---|---|
| DCTCP<br>L2DCT<br>PIAS | qsize=240$pkts$<br>ECN_markingthresh=65<br>min_rto=2$ms$ |
| pFabric | qsize=120$pkts$<br>min_rto=250$\mu$s |
| SPQ | qsize=240$pkts$<br>ECN_markingthresh=65<br>min_rto(short flows)=2$ms$<br>min_rto(long flows)=200$ms$ |

### B. Switch Implementation

**ECN**: there're two most important parameters: The ECN marking threshold K and the weight (g) given to new samples. the value of K has a great effect on the rate of flows. There're too many long flows compete link with short flows in fabric when given a high value. When given a low value, the rate of short flows will be restrained, and long flows can't obtain high throughout. So the appropriate value must provide low latency for short flows and high throughout for long flows [2], [12]. The value of g directly affects the congestion window size [2]. Referring to DCTCP, we set g=1/16, and K=65 packets for 10G link [2].

**Shallow buffer**: The appropriate buffer size must greatly alleviate the effect of long flows on short flows without resulting in too many timeout retransmissions. In our simulations, we set buffer size of each port to 360KB.

## V. Evaluation

### A. Simulation Illustration

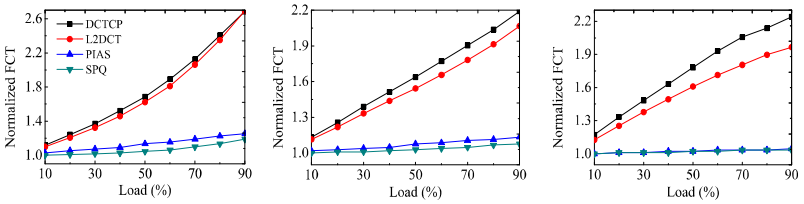
**Fabric topology setting**: we use a leaf-spine topology [1], [4], [5], [9] to evaluate these schemes. It is a widely used datacenter topology [1], [4], [9]. And the topology including 4 spine (core) switches, 9 leaf (top-of-rack) switches, and 144 hosts. Each of leaf switch uses 16 10 Gbps downlinks to connect the 16 hosts respectively, and 4 40 Gbps uplinks to connect the 4 core switches respectively.

**Traffic workloads**: We choose three traffic workloads: a web search workload [19], a data mining workload [9], and a hybrid workload [5] that consists of the former two workloads. In our simulations, all flows are generated on a Poisson process, and the source and destination of each flow are selected randomly [4], [5]. The three traffic workloads have the long-tailed characteristics mixing of latency-sensitive short flows and latency-insensitive long flows. According to their flow size distributions [4], we find: about the web search workload, the size of all flows under 1MB are over 70%, but these flows whose flow size over 1MB contribute over 95% of all bytes. It's more extremely skewed about the data mining workload: the flow size under 250KB are approach 90%, and the 95% of bytes come from 3.6% flows whose flow size are over 35MB [4]. So that the data mining workload is very easy to handle [4]. Therefore, among most of our simulations and analysis, we focus on web search and hybrid workload.

**Compared schemes**: We compare SPQ with four schemes: two information-agnostic schemes including DCTCP and L2DCT, a semi-information-agnostic scheme: PIAS, and the ideal information-aware and state-of-the-art scheme: pFabric. We refer to the best setting of the four schemes to choose their main parameters with a little modification. The specific value of these parameters we can find in the table 1.

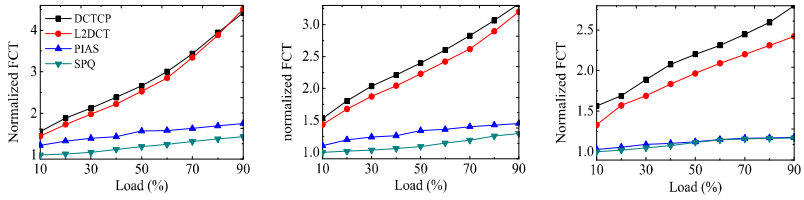
### B. Simulations Performance

*1) Comparison with information-agnostic and semi-information-agnostic schemes:* We use normalized FCT to illustrate the advantage of SPQ, and we regard the FCT of SPQ as the standard to measure other schemes. Here,




(a) Web search workload (b) Hybrid workload (c) Datamining workload

Fig. 4: (0,100KB] Avg: The normalized average FCT of short flows for three workloads at various loads.




(a) Web search workload (b) Hybrid workload (c) Datamining workload

Fig. 5: (0,100KB] 99th prctile: The normalized 99th percentile FCT of short flows for three workloads at various loads.

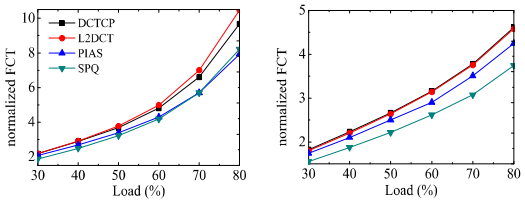
we compare SPQ with two information-agnostic schemes (DCTCP and L2DCT) and a semi-information-agnostic scheme (PIAS), the results suggest that SPQ achieves outstanding performance under three workloads. We mainly consider these aspects: the average and 99th percentile FCT of latency-sensitive short flows, and the average FCT of all flows.

**Latency-sensitive short flows:** Figures 4 and 5 illustrate that SPQ obtains the best performance for latency-sensitive short flows under the three workloads. On the other hand, figure 5 shows that SPQ deals with long-tailed distribution effectively for latency-sensitive short flows, SPQ achieves outstanding performance for the 99th percentile FCT of latency-sensitive short flows.

Compared with DCTCP, L2DCT, under the web search workload, SPQ reduces the average FCT of latency-sensitive short flows by up to 56% and 55% respectively, and reduces the 99th percentile FCT by up to 67% and 69% respectively. Under the hybrid workload, SPQ improves the average FCT of latency-sensitive short flows by up to 51% and 48% respectively, and improves the 99th percentile FCT by up to 61% and 59% respectively.

Compared with the semi-information-agnostic scheme PIAS, under the web search workload, SPQ improves the average FCT of short flows by up to 8%, and by up to 24% at the 99th percentile FCT. Under the hybrid workload, SPQ improves the average FCT of short flows by up to 5%, and by up to 19% at the 99th percentile FCT.

**Overall performance:** SPQ still achieves very excellent performance for the average FCT of all flows. Figure 6 shows that SPQ significantly outperforms DCTCP, L2DCT and PIAS under hybrid workload: improving the average FCT of all flows by up to 19%, 18% and 12% respectively. The results are analogous under the data mining workload. However, under web search workload, SPQ has the nearly similar performance with PIAS at high loads. The reason is that there're too many short flows (more than half) which restrict the rate of long




(a) Web search workload (b) Hybrid workload

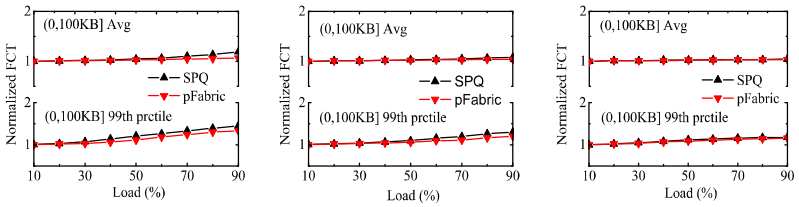
Fig. 6: Overall average: The normalized average FCT of all flows for web search workload and hybrid workload.

flows. At high loads, packet loss is more likely to occur in short flows. When it takes place, SPQ decreases the rate of long flows to ensure the low latency for short flows. It will increase the average FCT of long flows, so the average FCT of all flows with SPQ will increase.

*2) Comparison with the ideal information-aware scheme:* We regard the FCT of pFabric as the standard to evaluate SPQ. Figure 7 indicates that SPQ achieves analogous performance with pFabric. The performance gap between SPQ and pFabric is very small, which comes from high loads: at low loads, the average and 99th percentile FCT of short flows with SPQ are completely same with pFabric. For example, under the hybrid workload, the gap of average FCT is within 0-3.5%, and the gap of 99th percentile FCT is within 0-7.7%. Under data mining workload, the gap of average FCT is within 0-1.1%, and the gap of 99th percentile FCT is within 0-1.8%. However, SPQ achieves very wide practical scenes without any restrictions. It's worthwhile to sacrifice very little performance in exchange for very broad practical scenarios without any restrictions. It's also the purpose of this paper that achieves outstanding performance and broad practical scenarios.

## VI. Related Work

Existing flow scheduling schemes can be divided into three categories: information-agnostic scheduling schemes (e.g., [2], [3], [7], [12], [17]), information-aware scheduling




(a) Web search workload (b) Hybrid workload (c) Datamining workload

Fig. 7: (0, 100KB]: Normalized FCT of short flows for three workloads at various loads.

schemes(e.g., [4], [10], [11], [14], [15], [18]), and the semi-information-agnostic scheduling scheme (e.g., [5]).

**Information-agnostic scheduling schemes:** These schemes achieve very wide practical scenes at the expense of inferior performance, and they are readily-deployable flow scheduling schemes. They don't need to obtain any prior knowledge and modify hardware and software. For example, DCTCP [2] improves the FCT of short flows by shallow buffer and ECN [8]. On the basis of DCTCP, L2DCT [12], D2TCP [17],MCP [7] and HULL [3] add new mechanisms to improve the FCT of short flows or satisfy the deadline of deadline-constrained traffic respectively. However, these schemes don't support preemption and lack explicit distinguishing between latency-sensitive short flows and long flows in the fabric, leading to the inferior performance [11].

**Information-aware scheduling schemes:** These schemes achieve good performance. However, they need preliminarily to obtain flow information or modify hardware or software, which leads to difficult use and inefficient in practice. These schemes generally use prior knowledge to distinguish different flows. On the basis of prior knowledge of flow sizes or deadlines, pFabric [4] imitates Shortest Remaining Processing Time (SRPT) algorithm, PDQ [10] and D3 [18] use switch arbitration and explicit rate control to implement flow scheduling policy [5], and PASE [11] synthesizes the strengths of many solutions to provide outstanding performance [11]. LSTF [14] simulates Least Slack Time First (LSTF) scheduling algorithm.

**Semi-information-aware scheduling schemes:** The scheme (PIAS [5]) attempts to achieve suitable balance between outstanding performance and wide practical scenes. PIAS does not need to gain the size of all individual flows in advance and modify hardware. However, it not only needs to obtain the traffic distribution in advance, but also requires the datacenter applications unvarying, which makes PIAS just a semi-information-aware scheduling scheme with very narrow practical scenes.

## VII. Conclusion

We proposed SPQ, an information-agnostic and readily-deployable flow scheduling scheme to provide near-optimal FCT for latency-sensitive short flows and deal with long-tailed distribution effectively. SPQ achieves outstanding performance and wide practical scenes without any prior knowledge and modifying hardware and software. The simulation results show that SPQ achieves all of our design goals.

## VIII. Acknowledgment

This work is supported in part by the National High Technology Research and Development Program (863 Program) of China under Grant No.2013AA013203, No.2015AA016701, No.2015AA015301; National Basic Research 973 Program of China under Grant 2011CB302301; Key Laboratory of Information Storage System, Ministry of Education, China. This work is also supported by NSFC 61502190.

## References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proc. ACM SIGCOMM 2008*.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center tcp (dctcp). In *Proc. ACM SIGCOMM 2010*.

[3] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda. Less is more: Trading a little bandwidth for ultra-low latency in the data center. In *Proc. USENIX NSDI 2012*.

[4] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. pfabric: Minimal near-optimal datacenter transport. In *Proc. ACM SIGCOMM 2013*.

[5] Wei Bai, Li Chen, Kai Chen, Dongsu Han, Chen Tian, and Hao Wang. Information-agnostic flow scheduling for commodity data centers. In *Proc. USENIX NSDI 2015*.

[6] L. Chen, K. Chen, W. Bai, and M. Alizadeh. Scheduling mix-flows in commodity datacenters with karuna. In *Proc. ACM SIGCOMM 2016*.

[7] L. Chen, S. Hu, K. Chen, H. Wu, and D. H. K. Tsang. Towards minimal-delay deadline-driven data center tcp. In *Proc. ACM HotNets-XII 2013*.

[8] K. Ramakrishnan S. Floyd and D. Black. Rfc 3168: the addition of explicit congestion notification (ecn) to ip, 2001.

[9] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz., P. Patel, and S. Sengupta. Vl2: A scalable and flexible data center network. In *Proc. ACM SIGCOMM 2009*.

[10] C. Hong, M. Caesar, and P. B. Godfrey. Finishing flows quickly with preemptive scheduling. In *Proc. ACM SIGCOMM 2012*.

[11] A. Munir, G. Baig, S. M. Irteza, I. A. Qazi, A. Liu, and F. Dogar. Friends, not foes: Synthesizing existing transport strategies for data center networks. In *Proc. ACM SIGCOMM 2014*.

[12] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan. Minimizing flow completion times in data centers. In *Proc. IEEE INFOCOM 2013*.

[13] The Network Simulator NS-2. http://www.isi.edu/nsnam/ns/.

[14] M. Radhika, A. Rachit, R. Sylvia, and S. Scott. Universal packet scheduling. In *Proc. USENIX NSDI 2016*.

[15] A. Sivaraman, S. Subramanian, A. Agrawal, S. Chole, S. Chuang, T. Edsall, M. Alizadeh, S. Katti, N. McKeown, and H. Balakrishnan. Towards programmable packet scheduling. In *Proc. ACM HotNets-XIV 2015*.

[16] D. Maltz T. Benson, A. Akella. Network traffic characteristics of data centers in the wild. In *Proc. ACM IMC 2010*.

[17] B. Vamanan, J. Hasan, and T. N. Vijaykumar. Deadline-aware datacenter tcp (d2tcp). In *Proc. ACM SIGCOMM 2012*.

[18] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron. Better never than late: Meeting deadlines in datacenter networks. In *Proc. ACM SIGCOMM 2011*.

[19] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz. Detail: Reducing the flow completion time tail in datacenter networks. In *Proc. ACM SIGCOMM 2012*.