POLITECNICO DI TORINO
Repository ISTITUZIONALE

Low-complexity Flow Scheduling for Commodity Switches in Data Center Networks

Original
Low-complexity Flow Scheduling for Commodity Switches in Data Center Networks / Sviridov, G., Bianco, A., Giaccone, P.. - ELETTRONICO. - (2019). (2019 GLOBECOM - IEEE Global Communications Conference Waikoloa, HI, USA Dec. 2019) [10.1109/GLOBECOM38437.2019.9013612].

Availability:
This version is available at: 11583/2743732 since: 2020-05-17T16:11:32Z

Publisher:
IEEE

Published
DOI:10.1109/GLOBECOM38437.2019.9013612

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

13 October 2026

Low-complexity Flow Scheduling for Commodity Switches in Data Center Networks

German Sviridov, Andrea Bianco, Paolo Giaccone
Dipartimento di Elettronica e Telecomunicazioni - Politecnico di Torino - Torino, Italy
e-mail: firstname.lastname@polito.it

***Abstract*—Recently proposed approaches to minimize the Flow Completion Time (FCT) in data centers do not require any a-priory information about the flow size, thus appearing to be both practical and efficient. These solutions are based on a system of multiple priority queues (PQs) at both the servers and the switches and they may require to solve a complex algorithm to optimally split the traffic across the different PQs. However, the actual availability of priority queues at the switches is typically limited, thus restricting the applicability of these approaches.**

In this paper, we propose a novel approach, named NOS2, which requires only 2 PQs at the switches while maintaining multiple PQs at the servers, and leverages a central controller that optimally coordinates the traffic split among the different priority levels. We show by simulation that NOS2 is able to achieve performance close to state-of-art solutions with significantly smaller implementation complexity. Thus, NOS2 is expected to provide a better trade-off between performance and implementation complexity.

I. Introduction

In data center networks (DCNs) different types of flows have different requirements in terms of performance metrics. Mice flows require small flow completion time (FCT) because they typically belong to delay-sensitive applications, such as remote procedure calls (RPCs) or real-time interactive applications. On the contrary, elephant flows are bandwidth sensible and typically require large amount of bandwidth for prolonged time. The coexistence of the two types of flows inside the data center (DC) poses significant challenges for the performance optimization of the respective type of flow.

Standard approaches are based on classical transport protocols and on switches with buffers shared among the flows; as a consequence, all flows are likewise treated. Thus, the FCT of mice flows can be strongly deteriorated by the presence of elephant flows.

Recent proposals like PIAS [1] and Homa [2] try to address this issue by prioritizing mice flows over elephant flows in both the servers and the DC switches. Different types of flows are assigned a different priority, leading to a kind of "spatial division" of flows of different lengths into different priority queues (PQs), with highest priority assigned to mice flows. It was shown that 8 PQs at the servers permit to achieve a reduction in the average FCT up to a factor of 3 with respect to traditional scheduling using a single buffer. At the same time, the required number of PQs at the switches is typically larger than 4. Unfortunately, even if high-end switches are often equipped with 4 to 8 PQs, those queues are not fully available, being often employed to separate different types of traffic [3] such as RDMA [4], DCTCP [5] and traditional TCP-based traffic.

In this paper, we closely approximate the performance of PIAS and Homa with only 2 PQs in the switches, making our approach more practical. We propose *Network Optimized Split 2* (NOS2), which employs a simple flow scheduling at each server based on 8 PQs while maintaining only 2 PQs in the switches. NOS2 leverages the measured DC-wide flow distribution length (denoted as *workload* in the following). We show that such a small number of PQs in switches is enough to achieve performance similar or even better (depending on the scenario) than when employing 8 PQs.

The remainder of the paper is organized as follows. In Sec. II we present the relevant related work. In Sec. III we describe the main issues related to flow prioritization in DCNs. In Sec. IV we describe NOS2, our main contribution, whose performance are investigated in Sec. V. Finally, in Sec. VI we draw our conclusions.

II. Related work

The authors of [1] propose PIAS, which tries to minimize the FCT by splitting flows among different priorities without a precise a-priori knowledge of each flow length. The authors provide a formulation to find an optimal split of traffic among different PQs by exploiting information on the average DC-wide flow length distribution. However, the proposed architecture requires a large number of PQs inside switches, which is typically not easy to obtain. Furthermore the proposed optimal-split algorithm requires to solve a complex optimization problem, thus leading to poor reactivity in the case of sudden changes in the workload. In [3] the authors propose a flow prioritization scheme that exploits only two PQs at network switches. However, differently from NOS2, the architecture introduces a modification of the scheduling algorithm running at the servers and requires expensive modifications of the switch to execute the proposed prioritization mechanism.

In [6] the authors propose a scheduling algorithm, named pFabric, and show that it is able to achieve a near optimal FCT under realistic workloads. pFabric employs a fine-grained flow prioritization scheme that, in the worst case, needs an unbounded number of PQs, limiting its practical applicability. Furthermore, it requires an a-priori knowledge of the length of each individual flow, which is typically unknown and/or difficult to obtain in realistic scenarios.

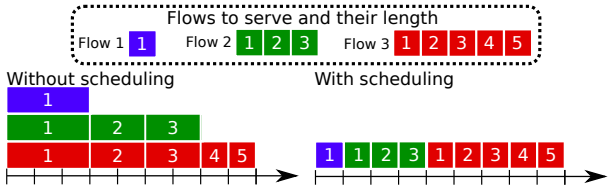


Fig. 1: Example of flow scheduling.

The work in [7] proposes to emulate the presence of PQs inside the network by acting on the congestion window of the transport protocol and giving more aggressiveness when sending the segments of short flows. However, this system provides marginal improvements with respect to a traditional transport protocol, and the performance is still worse than a system based on flow separation in different PQs.

In [2] the authors propose Homa, a credit-based transport protocol in which receivers drive the senders by assigning transmission credits. The credits serve a double purpose: i) to specify which senders are allowed to transmit over the DCN to receivers and ii) to define the network priority to be used during the data transmission. The priority is selected by exploiting information on the flow length distribution so as to prioritize short flows over long ones. However, similarly to [1], this approach requires a large number of PQs inside switches. Furthermore, Homa requires to completely rework the transport protocol inside each server.

III. Reducing FCT with flow prioritization

Flow scheduling consists in choosing a set of flows to serve from an available pool of active flows. Depending on metric employed for such choice, different scheduling algorithms can be implemented. Fig. 1 depicts an example of a pool of 3 flows of different length to be transmitted over a single link. Without any flow scheduler, all three flows will fairly share the available link bandwidth leading to a FCT of 3, 7 and 9 for flow 1, 2 and 3 respectively, achieving an average FCT of 6.33. On the other hand, a flow scheduler which privileges short flows over long one, for the same pool of flows, will lead to a FCT of 1, 4 and 10 with an average FCT of 5. In the latter example the FCT of short flows (flows 1 and 2) is reduced considerably without affecting significantly the performance of the long flows (flow 3).

In realistic workloads the majority of flows inside a data center are mice flows. Indeed, measurements from [8] show that 80% of flows are mice flows composed of less than 10MB. Flow schedulers that privilege short flows over long ones may lead to considerable FCT reductions for mice flows at the expense of the elephants. Nevertheless, [8] showed that mice flows account for less than 1% of the global DCN traffic, thus the performance penalty of long flows is expected to be negligible. In general, the performance of elephant flows depends on their length used distribution, in particular from the shape of the tail of the distribution. It is expected to observe smaller performance degradation for shorter tails and vice-versa.

A. *Flow-size aware scheduling*

In flow size-aware scheduling algorithms, the length of individual flows is assumed to be known in advance and packets belonging to shorter flows are served at higher priority with respect to packets belonging to longer flows. A flow maintains the priority level for its entire lifetime. This scheduling policy, known as Shortest Job First (SJF), has been shown to have a dramatic benefit for the average FCT [9]. A preemptive version of SJF, known as Shortest Remaining Job First (SRJF), assigns priorities to each flow related to the amount of bytes left to their completion. Flows with the least amount of missing bytes to transfer until their completion are served with higher priority with respect to flows requiring more bytes to transfer. This scheduling policy is used by pFabric [6] and it has been proved to be optimal [10] in terms of average FCT.

To implement prioritization of individual packets in real scenarios, strict priority (SP) scheduling is adopted and packets are stored in N separated queues. Let $\mathcal{Q} = \{q_p\}_{p=0}^{N-1}$ be the set of all queues, with p being the priority level (with 0 being the level corresponding to the highest priority). At each time instant the scheduler selects the highest priority queue $q_s \in \mathcal{Q}$ to serve, i.e., $s = \min\{p : q_p \text{ is not empty}\}$.

The main limitation of flow-size aware schedulers is that they require knowledge about the length of individual flows. Furthermore, both SJF and SRJF require a high priority granularity which in turns translate to a large N for the SP scheduler.

B. *Flow-size agnostic scheduling*

In realistic scenarios the length of individual flows is typically unknown or difficult to be obtained without undergoing into deep modifications of the application layer. Furthermore, the amount of PQs available in commercial switches typically ranges from $N = 4$ to $N = 8$ PQs, far less than the amount of PQs required for such fine-grained scheduling policies as SJF or SRJF.

In the absence of the knowledge about the length of individual flows, scheduling policies based on the amount of obtained service have been proved to be optimal. Least Attained Service (LAS) scheduler [11] always gives priority to flows with the least amount of attained service (transferred bytes) leading to small FCT for mice flows as they are prioritized over large flows. LAS is known to be optimal when the cumulative distribution function of the flow length $F(x)$ is known in advance and its hazard rate $h(x) = \frac{F'(x)}{1-F(x)}$ is decreasing [12]. Although mitigating the issue related to the absence of information about the length of individual flows, LAS still requires a large number of PQs as pFabric, thus making it impractical in real scenarios.

This limitation can be overcome by employing a discretized version of LAS, based on few queues (i.e., small N). This solution is known as Multilevel Feedback Queue (MLFQ). Similarly to a SP scheduler, MLFQ is composed of N queues (namely levels) served in a strict priority manner. However, differently from a normal SP scheduler, MLFQ permits flows to be demoted to lower priorities using a set of *demotion*

thresholds $\Omega = \{\omega_i\}_{i=1}^{N-1}$, which are based on the amount of service (in terms of transmitted bytes) $b_f(t)$ a given flow f has obtained up to time t. Notably, $b_f(t)$ is a per-flow counter available in many commercial switches for data centers.

Assuming $\omega_0 = 0$ and $\omega_N = +\infty$, at any given time t the priority p of a flow f is set such that $\omega_p \leq b_f(t) < \omega_{p+1}$. When a new flow f arrives at the transmission buffer of a server, the flow is assigned a priority $p = 0$ and its packets are stored in the highest priority queue q_0. When the amount of served bytes exceeds the first threshold, i.e., $b_f(t) > \omega_1$, the flow is demoted and assigned a lower priority $p = 1$. Consequently, all new packets belonging to the demoted flow will be stored in q_1. This process repeats until the flow terminates or eventually reaches the lowest priority queue $N-1$, where it remains up to completion.

The most crucial aspect in a MLFQ scheduler is the definition of a proper set Ω of demotion thresholds. A simple approach, derived from [6], named Equal Split with N levels (ES-N), splits $F(x)$ in N equal percentiles, as follows:

$$\omega_i = F^{-1}\left(\frac{i}{N}\right), \qquad i = 1, \ldots, N-1$$

However, this approach may lead to early flow demotion, with mice flows quickly ending up in lower priorities together with elephant flows. Similarly, it may lead to late elephant demotion by keeping elephant flows mixed with mice flows for too long in high priority queues before demoting them. This ultimately leads to unbalanced utilization of available PQs, and, as a consequence, to FCT deterioration.

To overcome this limitations, PIAS [1] proposes an information agnostic scheduler able to achieve performance similar to those of pFabric while employing $N = 8$ PQs and only the knowledge about the overall *DC-wide* flow length distribution. PIAS addresses the issue related to unbalanced PQs utilization by providing a formulation for an Optimal Threshold Assignment (OPT) so as to provide best PQs utilization and to minimize the average FCT inside the MLFQ scheduler. The formulation represents the MLFQ scheduler as a tandem of N M/M/1 queues, each queue corresponding to a priority level of the MLFQ scheduler. The service time of each queue is made dependent on the measured average DC-wide flow length distribution. The objective function of the OPT problem is constructed to minimize the average queueing delay inside the tandem of queues, thus minimizing also the average FCT.

Fig. 2 shows two alternative architectures, one based on PIAS and the other on ES-N to compute Ω. Both architectures employ a MultiLevel Feedback Queue (MLFQ) but the thresholds in each server to serve data flows coming from upper layers is computed either with OPT or with ES. A central threshold controller collects statistics on the flows generated at the servers, derives the corresponding workload and executes the OPT (or ES) algorithm. At the same time, servers monitor and notify the threshold controller their flow statistics, and receive the updates on Ω.

The server, before transmitting a packet to the network, tags the packet with a priority equal to the MLFQ level within which it was enqueued. The priorities assigned by each server must be consistent throughout the DCN, meaning that switches must have a number of PQs equal to the number of levels of the MLFQ employed at servers. Upon packet reception, switches are left with the sole role of performing SP scheduling based on the priority tag. To do so, standard mechanisms based on IEEE 802.1p [13] can be exploited, being already available in most of commercial DC switches.

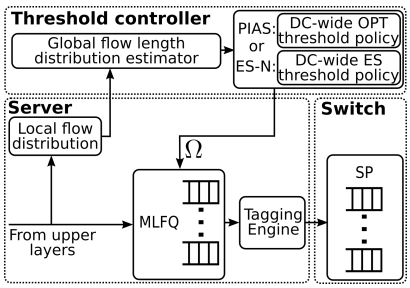


Fig. 2: High-level architecture of PIAS and of ES-N, with many PQs at the servers and at the switches.

C. Applicability in a realistic scenario

The OPT threshold assignment employed in PIAS is known to be NP hard [14] and, to the best of our knowledge, no proven approximation exists for $N > 2$. Thus, the solution can only be found by means of heuristics, which do not provide any guarantees on the optimality of the obtained solution and possibly require a large amount of time to converge.

PIAS presents a major restraint in the case of sudden change in the workload or in the case of sub-optimal solutions, which may lead to threshold-workload mismatch, which was shown to deteriorate PIAS performance by up to 25% [1].

For a wide range of traffic types, most of the demotion thresholds in Ω are located after the 90th percentile of the flow length distribution. Since the majority of realistic workloads are heavy tailed and their empirical estimation is hard to achieve, the estimation errors reduce the efficiency of MLFQ. Indeed, if the tail of the workload has been underestimated, flows may be demoted too early, leading to most flows ending up in the low PQs too early. On the contrary, if the tail has been overestimated, flows may never be demoted to lower priorities, thus reducing the potential performance gain of MLFQ. Furthermore, in the case of different servers with different workloads, PIAS may lead to sub-optimal Ω assignment. Due to the fact that PIAS employs the average DC-wide flow length distribution, servers will use demotion thresholds which are naturally mismatched with respect to their actual fine-grained local workloads.

Despite the previously cited drawbacks, the main disadvantage of PIAS is the large number of required PQs. Indeed, although the majority of commercial switches offer up to 8 PQs, as previously discussed, most of them are typically unavailable, because they are utilized to perform isolation of different types of traffic.

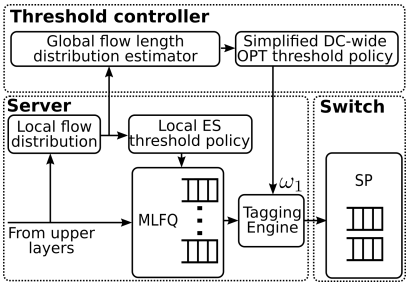


Fig. 3: High-level architecture of NOS2, with multiple PQs at the servers and just 2 PQs in all the switches.

IV. NOS2 ARCHITECTURE

To overcome the shortcomings of previously proposed approaches, we propose the NOS2 architecture. Our key idea is to decouple the MLFQ system with multiple queues at the servers from the queueing occurring at the switches, which adopt only 2 PQs, independently of the number of queues at the servers. The high priority queue at the switches is devoted to mice flows and the low priority one to elephant flows.

Fig. 3 depicts the NOS2 architecture. Differently from PIAS and ES-N, NOS2 leaves to servers the complete freedom in managing their demotion thresholds on the basis of the locally observed workload. However, this does not introduce any expensive threshold computation since NOS2 involves simple ES thresholds for MLFQ scheduler at the servers. At the same time, to compensate for eventual performance losses due to non-optimized thresholds at servers, demotion thresholds inside switches are computed by the threshold controller based on the DC-wide flow length distribution, as in PIAS.

A. Threshold controller

The central controller gathers flow statistics from the servers, computes the DC-global flow length distribution and computes Ω for the switches based on OPT algorithm. Since the switches adopt only 2 queues and just one threshold is needed (i.e., $\Omega = \{\omega_1\}$) the complexity of OPT algorithm is considerably lower with respect to the case of multiple thresholds. Furthermore, in [15] it is shown that, under realistic conditions, there exists a closed-form approximate expression to compute the optimal value of ω_1 with good accuracy. Intuitively, the controller must identify just the elephant and mice flows globally in the DC and instruct the switches to enqueue them in low priority and high priority queues, respectively. Now, as shown in Fig. 3, a single optimal threshold ω_1 is sent to all the servers.

B. Server scheduling and tagging

At each server NOS2 employs MLFQ with simple ES-N threshold policy based on the local estimation of the workload, as shown in Fig. 3. ES-N introduces greater adaptability to variable traffic patterns. Indeed, in the case of a sudden change in workload, a new set Ω can be obtained as soon as the new flow length distribution is obtained, without the latency due to running complex OPT algorithm for multiple thresholds.

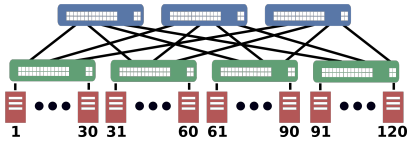


Fig. 4: Data center topology used for the simulations.

Higher workload adaptability comes at a cost of a performance penalty. However, in [1] it was shown that when using ES-8, i.e., with 8 PQs, there is a mere 10% penalty in FCT with respect to using thresholds obtained by the OPT algorithm. This behavior is confirmed later in Sec. V by our simulation results which show an even smaller performance gap between ES-8 and OPT thresholds.

As described before, the role of discriminating flows between mice and elephant flows is left to the threshold controller. Given ω_1, servers perform a simple packet tagging with the use of a Tagging Engine depicted in Fig. 3. Packets are tagged with either a high or low priority tag depending on the amount of already transmitted bytes for the corresponding flow.

C. Switch scheduling

As shown in Fig. 3, the switch enqueues the incoming traffic into one of the two PQs based on the priority level specified by the tag present in the packet and set by the servers. Then, the 2 queues are served with a strict priority scheduling policy. The switches do not require to maintain any per-flow state to implement NOS2, because the optimal discrimination between elephant and mice flows is delegated to the servers and to the central threshold controller.

In summary, the NOS2 approach integrates a fine-grained scheduling with multiple priority levels at each server with a coarse-grained scheduling at the switches that simply keep mice and elephant flows separated inside the DCN.

V. PERFORMANCE EVALUATION

In this section we analyze the performance of the proposed approach by simulations.

A. Simulation methodology

We perform the analysis using the discrete-event simulator NS3 [16], which provides detailed simulation models for the Internet protocols stacks from MAC layer up to the application layer. We consider a standard Ethernet-based leaf-and-spine topology for the DCN, as shown in Fig. 4. The chosen topology connects 120 servers and comprises 3 spine switches, 4 leaf switches and 30 servers per leaf switch. All servers are connected to the leaf switches via a 1 Gbps link, while leaf switches are connected to spine switches via a 10 Gbps link. Thanks to this link dimensioning, the DCN provides full bisection bandwidth.

The queueing mechanisms described in Sec. IV are implemented at the output network interfaces of servers and

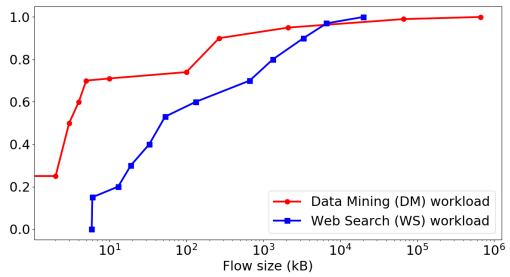


Fig. 5: Flow length distributions employed in simulation.

switches. We employ DCTCP as the end-to-end transport protocol with marking thresholds and gain parameters set according to the guidelines provided in [5], while load balancing across the DCN is performed using ECMP.

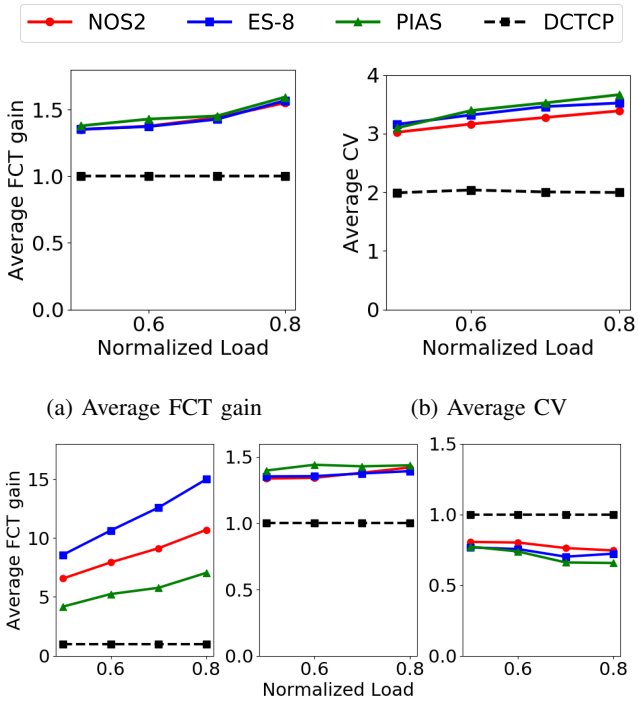
Traffic flows arrive according to a Poisson process and the flow lengths are randomly generated according to a given distribution. We classify flows into mice flows (<100kB), medium flows and elephant flows (>10MB). The destination of each flow is uniformly chosen at random among the other 119 servers. The data center load λ is defined as the average amount of traffic destined to the servers, normalized with respect to the bisection bandwidth. We vary λ between 0.5 and 0.8 to simulate different loads inside the DC. We do not show the results for $\lambda < 0.4$ since the performance of all the considered schemes are almost identical.

We consider for our experiments two realistic, heavy-tailed flow length distributions, depicted in Fig. 5: Data Mining (DM) [8] and Web Search (WS) [5]. The former is composed predominantly of mice flows (80%) and elephants flows (20%). The latter instead shows a smoother transition from mice to elephants. We consider a workload scenario in which all servers generate traffic according to one of the two flow length distributions (DM or WS).

We evaluate the average value and the coefficient of variation (CV) of the Flow Completion Time (FCT), for all the flows, and, independently for each class of flows (mice, medium and elephant). The FCT is measured from the generation at the source server of the first packet belonging to a flow until the reception of the last packet at the destination server. Evaluating the FCT is important to infer the performance of typical DC delay-sensitive applications, which directly affects the Quality of Experience perceived by the end users accessing cloud-based applications.

We compare NOS2 with respect to the following alternative solutions:

- *ES-8*: the architecture in Fig. 2 with MLFQ, $N = 8$ and ES-8 threshold computation.
- *PIAS*: the architecture in Fig. 2 with a MLFQ, $N = 8$ and OPT thresholds computation. The optimal threshold values are taken from the publicly available PIAS source code.
- *DCTCP*: as a reference case, we run bare DCTCP at servers with a single queue at servers/switches, thus disabling completely any flow prioritization scheme.



(a) Average FCT gain (b) Average CV

(c) Average FCT gain for different flow types

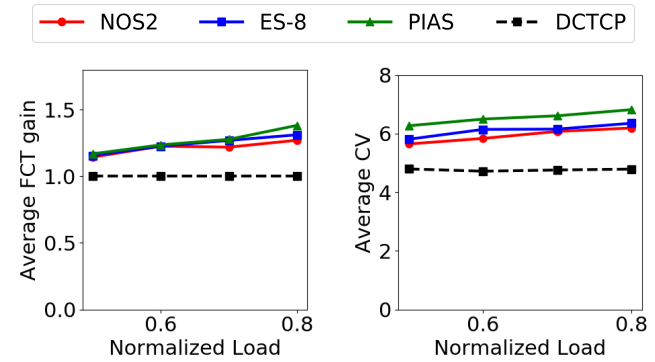
Fig. 6: Web Search workload

We report the *FCT gain*, defined as the FCT achieved by DCTCP divided by the FCT of the considered scheme. Thus, a FCT gain $G > 1$ means that the FCT is reduced by a factor G with respect to DCTCP.

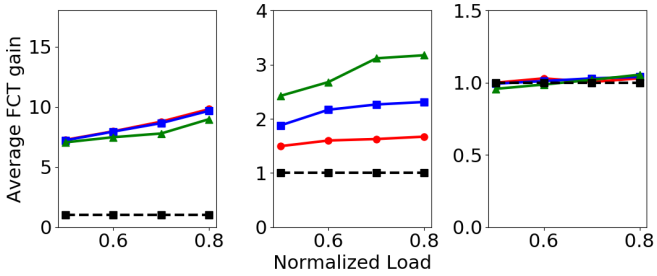
B. Performance analysis

Fig. 6a and Fig. 7a show the aggregate average FCT for the three systems under WS and DM workloads respectively. In both cases, the three systems behave very similarly in terms of average FCT. We now focus on the performance analysis of mice, medium and elephant flows. Under the WS workload, Fig. 6c shows that, for mice flows, the best FCT is obtained by ES-8, the worst by PIAS, with NOS2 showing an intermediate behavior. The opposite holds for the elephant flows, while for medium flows the FCT is almost unaffected by the adopted scheme. On the contrary, Fig. 7c shows that, for the DM workload, only the performance experienced by medium flows are affected by the adopted scheme, and, in this case, the best scheme is PIAS while the worst is NOS2, which still keeps the FCT gain larger than one. It is worth noting that in DM workload the amount of medium flows accounts only for 4% of the flows, which explains why the performance gap between the three architectures in Fig. 7a is so mild.

The explanation behind the relative performance degradation for NOS2 for medium-sized flows is that 2 PQs cannot offer enough resolution to differentiate among all flow types. Since the majority of flows in DM workload are either mice or elephant, the threshold setting which optimizes the average



(a) Average FCT gain (b) Average CV



(c) Average FCT gain for different flow types

Fig. 7: Datamining workload

FCT tries to optimize those two types of flows at the price of sacrificing the performance of medium flows. On the contrary, the WS workload presents a higher concentration of medium flows, which leads to the threshold being optimized for those flows. Similar reasoning applies when it comes to explaining the performance of elephant flows. For WS workload, the performance for elephant flows is penalized considerably compared to DCTCP. Although WS and DM have a similar fraction of elephant flows, WS distribution presents a considerably shorter tail, thus smaller average size of elephant flows. In view of the discussion made in Sec. III, this leads to a bigger penalty when mice flows are prioritized over them. On the other hand, in the case of DM workload the tail of the distribution is considerably longer, thus this effect is less visible.

For what concerns the coefficient of variation of the FCT, reported in Fig. 6b and Fig. 7b, the three systems perform similarly, with NOS2 performing slightly better than ES-8 and PIAS for both WS and DM workloads. There is still a considerable gap between the three architectures and DCTCP. This is not surprising, since DCTCP aims at treating each flow fairly while NOS2, ES-8 and PIAS privilege short flows at the expense of long flows, thus increasing the variance of the FCT.

VI. Conclusions

We introduce a low complexity flow scheduling mechanism, named NOS2. NOS2 exploits a fine-grained MLFQ flow scheduling at each host, with multiple queues, while adopts a simple strict priority scheduler at each switch with only 2 queues. NOS2 leverages the local estimation of the flow length distribution to set the demotion thresholds of the MLFQ scheduler at each server. To improve the reactiveness and reduce the computation complexity, the server demotion thresholds are computed through the Equal Split algorithm. On the contrary, a single demotion threshold for the switches is adopted and thus the traffic is tagged as either high priority or low priority when transmitted at the server. This threshold is globally computed by the central threshold controller, which runs a simple but optimal algorithm.

Our simulation results show that NOS2 achieves a near identical FCT to PIAS and an ES-8 based architecture with 8 PQs in both switches and hosts. Thus, given the lower number of queues and the simpler and more accurate computation of the thresholds, NOS2 is shown to achieve the best tradeoff between performance and complexity.

References

[1] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Information-agnostic flow scheduling for commodity data centers," in *NSDI*, 2015, pp. 455–468.

[2] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *ACM SIGCOMM*, 2018, pp. 221–235.

[3] Y. Lu, G. Chen, L. Luo, K. Tan, Y. Xiong, X. Wang, and E. Chen, "One more queue is enough: Minimizing flow completion time with explicit priority notification," in *IEEE INFOCOM*, 2017.

[4] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "RDMA over commodity Ethernet at scale," in *ACM SIGCOMM*, 2016.

[5] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," *ACM SIGCOMM CCR*, vol. 41, no. 4, pp. 63–74, 2011.

[6] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pFabric: Minimal near-optimal datacenter transport," in *ACM SIGCOMM CCR*, vol. 43, no. 4, 2013, pp. 435–446.

[7] 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 *IEEE INFOCOM*, 2013, pp. 2157–2165.

[8] 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 *ACM SIGCOMM CCR*, 2009.

[9] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of scheduling*. Courier Corporation, 2003.

[10] L. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, 1968.

[11] M. Nuyens and A. Wierman, "The foreground–background queue: a survey," *Performance evaluation*, vol. 65, no. 3-4, pp. 286–307, 2008.

[12] S. Aalto and U. Ayesta, "Recent sojourn time results for multi-level processor-sharing scheduling disciplines," *Statistica Neerlandica*, vol. 62, no. 3, pp. 266–282, 2008.

[13] "IEEE 802.1p standard." [Online]. Available: https://www.ieee802.org/

[14] H.-W. Jiao and S.-Y. Liu, "A practicable branch and bound algorithm for sum of linear ratios problem," *European Journal of Operational Research*, vol. 243, no. 3, pp. 723–730, 2015.

[15] K. Avrachenkov, P. Brown, and N. Osipova, "Optimal choice of threshold in two level processor sharing," *Annals of Operations Research*, 2009.

[16] G. F. Riley and T. R. Henderson, "The NS-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.