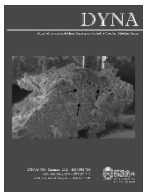
DYNA
ISSN: 0012-7353
Universidad Nacional de Colombia

Gutiérrez, Sergio Armando; Barcellos, Marinho; Branch, John Willian
Dynamic adjustment of a MLFQ flow scheduler to improve cloud applications performance
DYNA, vol. 85, no. 206, 2018, July-September, pp. 16-23
Universidad Nacional de Colombia

DOI: https://doi.org/10.15446/dyna.v85n206.71626

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS



DYNA
http://dyna.medellin.unal.edu.co/

Dynamic adjustment of a MLFQ flow scheduler to improve cloud applications performance

Sergio Armando Gutiérrez [ac], Marinho Barcellos [b] & John Willian Branch [a]

[a],*Facultad de Minas, Universidad Nacional de Colombia, Medellín, Colombia, saguti@unal.edu.co, jwbranch@unal.edu.co*
[b] *Instituto de Infiormática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil, marinho@inf.ufrgs.br*
[c] *Facultad de Ingenierías, Universidad de Medellín, Medellín, Colombia, sagutierrez@udem.edu.co*

Received: April 10th, de 2018. Received in revised form: June 7th, 2018. Accepted: June 12th, 2018

Abstract
State-of-the-art solutions for flow scheduling propose the use of Multi Level Feedback Queue (MLFQ) as a mechanism to avoid the requirement of prior information (i.e. agnosticism) regarding flow sizes. This is an important aspect to achieve the performance goals of high responsiveness and high throughput that is expected in Cloud Applications (e.g. search engines, social networks, and e-commerce sites). These goals are tightly associated with the prioritization of short flows (a few KB in size), the majority for these applications rather than long flows (several MB in size). However, these applications usually cannot provide information in advance about the size of the flows. In this paper, we analyze the feasibility of providing dynamic adjustment for a MLFQ-based scheduling system in such a way that it adapts itself to the time and space variations exhibited by Data Center Network (DCN) traffic without requiring prior information about workload properties.

Keywords: Flow scheduling; data center networks; MLFQ; agnostic flow scheduling

Ajuste dinámico de un programador de flujos MLFQ para mejorar el desempeño de aplicaciones en la nube

Resumen
Las soluciones presentes actualmente en el área de conmutación de flujos proponen el uso del concepto de Colas Multinivel con Realimentación (MLFQ por su sigla en inglés) como mecanismo para evitar el requerimiento de información previa (mecanismo agnóstico) con respecto al tamaño de los flujos de datos. Este es un aspecto importante para el logro de las metas de desempeño de alta capacidad de respuesta y alto rendimiento, esperadas en las aplicaciones en la nube (Por ejemplo, motores de búsqueda, redes sociales y sitios de comercio electrónico). Estas metas están estrechamente asociadas a la priorización de los flujos cortos (con tamaños de unos pocos KB), mayoritarios en estas aplicaciones, sobre los flujos largos (con tamaños de varios MB). Sin embargo, estas aplicaciones usualmente no son capaces de proporcionar de antemano la información acerca del tamaño de los flujos. En este artículo, analizamos la viabilidad de proporcionar ajuste dinámico a un esquema de conmutación basado en MLFQ, de tal manera que éste sea capaz de adaptarse a las variaciones espacio temporales que se observan en el tráfico presente en las redes de centro de datos, sin que se requiera información previa sobre las propiedades de las cargas de trabajo.

Palabras clave: Conmutación de flujos; redes de centro de datos; MLFQ; conmutación agnóstica de flujos

1. Introduction

Cloud applications running on Data Center Networks (DCN) have very strict performance requirements which, when unsatisfied, might affect the revenue obtained by the applications' owners [1,2,5-7]. The traffic associated with these applications consists of a mix of short (those transporting a few kilobytes) and long flows (those transporting several megabytes or gigabytes). An important performance goal for these applications is the minimization of the flow completion time (FCT) of short flows without inducing starvation in long flows [3,4].

The literature proposes flow scheduling and queue management as strategies to achieve this goal of minimizing

How to cite: Gutiérrez, S.A., Barcellos, M. and Branch, J.W., Dynamic adjustment of a MLFQ flow scheduler to improve cloud applicationsperformance. DYNA, 85(206), pp. 16-23, September, 2018.

the FCT of short flows. However, this involves three challenges: First, in many cases, it is not possible to have a-priori information about the flow sizes to plan the scheduling [5,6]. Hence, it is impossible to know whether a flow will be short or long when it starts. Second, even if this information is available, taking advantage of it might imply prohibitive modifications in different elements of the infrastructure (switches and end hosts) and/or the applications themselves [4]. Third, the scheduling mechanisms need to adapt themselves to the time and space variations that the traffic might exhibit [9,19].

Solutions for flow scheduling reported in the literature [4-6,9,13,15] fail to address some or all of the previously mentioned challenges. In particular, although some approaches are agnostic regarding the specific size of flows for its scheduling, they still assume that it is possible to know properties in advance such as the flow size distribution for the workloads present on the network [6,9].

In this paper, we analyze the situation, especially in the context of solutions based on a Multi-Level Feedback Queue (MLFQ) scheduler when the current network traffic does not fit the scheduling configuration. By using simulation, we observe how, despite the fact that this approach introduces agnosticism in the flow scheduling, it is still dependent on previous knowledge of some general features for the workload present on the network to achieve the goal of minimizing FCT for the short flows. To address this dependency, we propose the implementation of a monitoring mechanism system focused on detecting traffic properties that can be used to dynamically adjust the scheduling configuration in such a way that it can handle the space and time variations that the traffic associated to cloud applications usually exhibits [1,7,14,19].

This paper is organized as follows: Section 2 analyzes the demotion threshold problem that might be present in MLFQ-based flow scheduling systems. Section 3 describes our monitoring approach and discusses some assumptions that it leverages. Section 4 presents the results of a preliminary evaluation of the effectiveness of our proposal to minimize the FCT of short flows on a Data Center workload. Section 5 presents the related literature, and Section 6 concludes the paper.

2. Threshold mismatch in agnostic flow scheduling mechanisms

2.1. MLFQ

MLFQ is a concept that has come from Operating Systems [10]. It was conceived for multiuser timeshare systems and it is based on employing a multilevel queue to manage processes, that is, the instances of the programs executed on the system. The goal of MLFQ is to approximate the Shortest Job First (SJF) scheduling heuristic, which theoretically minimizes both the average and tail process completion time. MLFQ aims to prioritize short processes (which are mainly associated with interactive operations and I/O intensive operations) over long processes (usually associated to batch operations), without knowing in advance the exact duration of the former.

Fig. 1 presents the scheme of a MLFQ scheduler in an operating system. It consists of a set of FIFO queues and a scheduling policy. The number of queues goes from two up to a given number according to the implementation (e.g. the Solaris Operating System uses sixty). Each queue has an associated value indicating the maximum quantum for a process to be queued at that given queue (i.e. a given priority level). Higher priority queues have smaller quantum values associated and these values increase between successive queues.

Initially, all the processes enter the highest priority queue. A process is demoted and queued on the next priority queue whenever it exceeds the specified quantum to stay at that queue. Hence, this approach leads to the prioritization of short over long processes. The former tend to complete within the first queues whereas the later tend to sink towards the lowest priority queue.

The scheduling policy in MLFQ is very simple. In order to determine the next process to dequeue, the set of queues is checked following a top-down strategy. Initially, the highest priority queue is checked. If this queue is non-empty, then the first process in the queue is scheduled and runs until blocking. This process is then moved to the end of the same queue (implementing FIFO) or to the end of next queue (demoted to the next lower priority). However, if the highest priority queue is empty, then the next queue (lower priority) is checked just as with the highest one. The scheduling is repeated for the next process, always starting with the highest priority queue.

Recently, the literature on flow scheduling proposed to adapt the notion of MLFQ to schedule data flows. Flow scheduling approaches based on MLFQ aim at approximating scheduling heuristics such as Least Attained Service (LAS) or Shortest Job First (SJF) in order to prioritize short flows [16]. This approximation is performed by dynamically assigning priorities to flows without actually knowing their size in advance. Flows are initially processed at the top priority queue of a MLFQ, and they are progressively demoted according to a given criteria such as deadline meeting, byte count, or associated congestion notifications [3, 5, 6, 9].

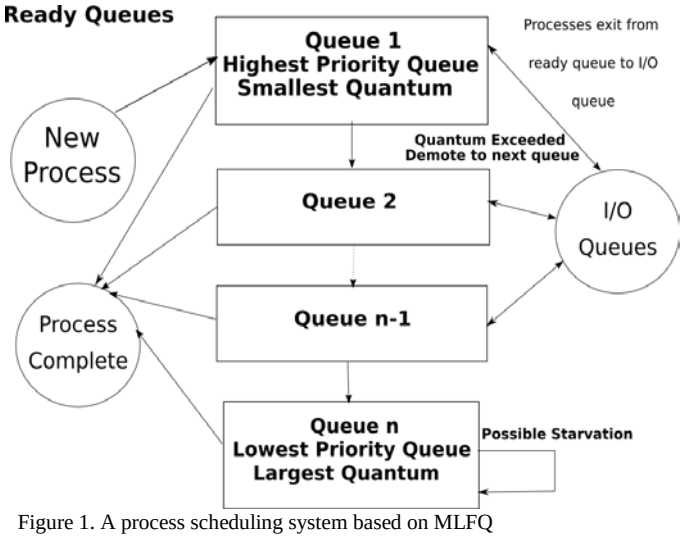


Figure 1. A process scheduling system based on MLFQ
Source: The authors

2.2. Threshold mismatch in MLFQ

Defining the demotion thresholds for an MLFQ scheduler is a challenging task. It involves determining the right values that indicate when the scheduled entity should be moved from a given priority level to another lower level.

This definition is associated with a specific performance goal. In the case of flow scheduling for cloud applications, these thresholds should aim at minimizing the FCT of short flows since it influences the responsiveness as perceived by the end user. However, although approaches based on MLFQ are conceptually agnostic (i.e. they do not require a-priori information about the duration of a process or the size of a flow for the scheduling), they still require some prior information to derive the demotion thresholds. This information is associated to specific properties of the traffic workloads. If the demotion thresholds do not fit these properties, then the threshold mismatch problem arises. Specifically, in the context of flow scheduling, the threshold mismatch problem might hamper the goal of minimizing the FCT, and this might hurt especially short flows.

To illustrate this point, assume a simplified MLFQ with two priority queues and, therefore, a single demotion threshold. Suppose that, for the sake of simplicity, there are only two flow sizes: 10KB and 10MB. Assume that for a given workload, 90% of its flows are 10KB and the remaining 10% are 10MB. In this situation, it would be optimal to set the demotion threshold to 10KB since this would keep the short flows in the high priority queue until their completion. Thus, short flows are prioritized over the long flows, which will ultimately be queued into the low priority queue. Assume that later, there is a shift in the workload. In the new workload, the size of small flows is 20KB instead of 10KB. With the demotion threshold set to 10KB, some of the packets of the short flows will be demoted to the low priority queue. Hence, the latency for these short flows will be larger than it should be. Fig. 2 illustrates this situation.

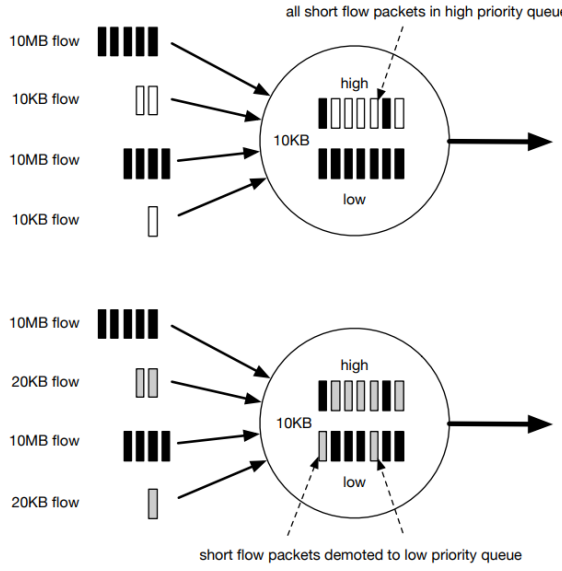


Figure 2. The threshold mismatch problem
Source: The authors

3. Dynamic adjustment of MLFQ scheduler

In this section, we propose an approach to adapt the configuration of a MLFQ scheduler, aiming to achieve workload-agnostic operation. That is, an approach that avoids the requirement of prior information about workload properties that are present in the network.

3.1. Overview

As we previously discussed, state-of-the-art solutions addressing agnostic flow scheduling require information about the distribution of the flow sizes for the workloads. This information is used to derive the demotion thresholds for the MLFQ scheduler. This means that, whenever the workload shifts, there might be a mismatch between the current scheduling configuration and the current workload. This mismatch might adversely affect the FCT for the short flows.

We propose observing the traffic entering into the switch ports. This observation, performed during a given time slot (monitoring window) enables the acquisition of information to infer the traffic behavior associated to the workloads present in the network. Periodically (say, every 100ms), the demotion thresholds are adjusted according to information acquired during the current monitoring window (say 500ms). The rationale behind this approach is to *use the past to try to predict the future*

3.2. Assumptions

Our concept of adaption of MLFQ scheduling leverages the notion of Programmable Switches. Hence, we assume that the MLFQ is implemented with the switch queues associated to each port, and the demotion thresholds are defined and maintained within the switch. The operations of traffic monitoring and threshold adjustment are executed also within the switch (i.e. separated execution threads). This is essentially different to the approach proposed by related work, which considers a packet tagging mechanism based on MLFQ, with strict priority scheduling performed at switches [6, 9].

Although the MLFQ scheduling is implemented at switches, some degree of cooperation is required from the end hosts. Particularly, we assume that end hosts can add a header to packets to inform a) the amount of bytes that a flow has currently sent and b) the final size of the flow. The addition of packet headers has already been used in related work [6,13,15].

Switches receive and process this information sent from end hosts and maintain a sorted list for each port, which contains the size of each flow that is completed during the monitoring window. Then, periodically, the switch calculates a set of predefined percentiles (say 10^{th}, 20^{th}, etc) values from that sorted list. These percentiles provide a clue to adjust the demotion thresholds of the MLFQ scheduler of the particular port. After the adjustment, the list of flow sizes is pruned to make the monitoring window slide forward.

Our observation is that low percentiles can define upper bounds for short flows, which should be associated to higher

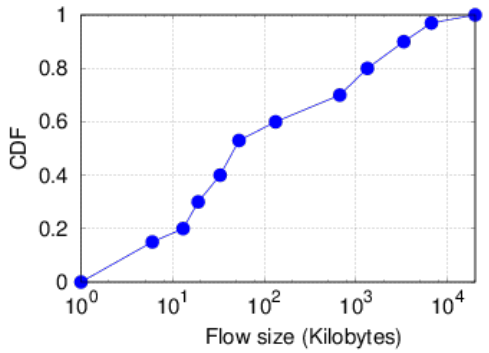


Figure 3. Typical Web Search workload in a production data center
Source: The Authors with data from [19]

priority levels, and higher percentiles can define upper bounds for long flows, which should be associated to lower priority levels. For example, consider the workload Web Search described in Fig. 3. Assume a MLFQ system with two queues (therefore, one demotion threshold). For that workload, the plotted CDF indicates that 80% of the flows are shorter than 10KB. Our intuition is that by calculating low percentiles (say 10th or 20th) in the list of completed flow sizes, we might have a good idea of setting a demotion threshold that approximates the upper bound for the short flows in the workload and then prioritize them by separating from the long flows.

We claim that one advantage of implementing our threshold adjustment mechanism within the switch is avoiding misuse by a malicious host. For example, with packet marking at end hosts, it might happen that a malicious host marks its packets with the highest priority value regardless the amount of bytes that flows have transmitted. In contrast, in our approach, this misuse is avoided since the switch schedules packets according to the amount of bytes transmitted by each flow. Also, since the threshold adjustment is local to switches, it adapts to the time and space variations of the traffic. For example, for different switches, the notion of what is a short flow and the sizes of these flows varies. However, since short flows are the majority in typical data center workloads, the calculation of low percentiles at each switch will yield to prioritize short flows at each switch. This also contrasts with the related work, in which the demotion thresholds are configured globally for all the DCN switches.

4. Experiments and results

4.1. Simulation model

We evaluated our proposal of threshold adjustment via NS-2. This is a discrete time, packet-based network simulator, widely used in the literature to perform large scale evaluation of different solutions in the field of computer networks. Our simulations were executed on a server with an Intel(R) Core(TM) i7-4790S CPU with eight cores @ 3.20Ghz and 8GB of RAM. In our experiments, we compare our proposal of threshold adjustment against the closest related work which is PIAS [6], with static threshold configuration.

4.1.1. Assumptions and input factors

In our experiments, we followed the same guidelines as related work. For our transport protocol, we used DCTCP [1] configured with the recommended values for the parameters K and g (65 and 1/16 respectively) [2]. For the control of the Explicit Congestion Notification, we implemented per-port marking. For a detailed analysis of this choice, please refer to Section IV-A number 2 of [6].

For our evaluation, we considered different traffic load levels ranging from 50% to 90%. We define the monitoring window to be 250ms and we perform the threshold adjustment in 200ms intervals.

4.1.2. Topology

A switch S0 with four connected hosts named from N0 to N3 forms our experimental topology. Links connecting hosts and switches are all full duplex with a bandwidth of 10Gbps and propagation delay of 20.2us. These values correspond to typical values present in infrastructure of production data centers [6]. There are two queues and a single demotion threshold that form the MLFQ system for this configuration. Fig. 4 shows the schematic diagram of the experimental topology.

4.1.3. Workload

The workload consisted of 10000 flows, with 80% being short (10 KB) and 20% being long flows (1000 KB). Nodes N1 and N3 sent long flows towards N0 whereas N2 sent short flows towards N0. In these experiments, we defined that the percent that was going to be calculated from the list of completed flow sizes to update the demotion threshold was the 10^{th} percentile.

4.1.4. Convergence of the Demotion Thresholds

In this section, we analyze the convergence of the demotion threshold. For the sake of comparison, we assess three different values of demotion threshold: 1KB, 10KB and 1000KB that are used for static configuration and for initial value of the demotion threshold. These particular values induce three situations: in the first configuration (1KB), short flows are demoted prematurely. That is, when the flow has transmitted 1KB, its packets are queued into the low priority

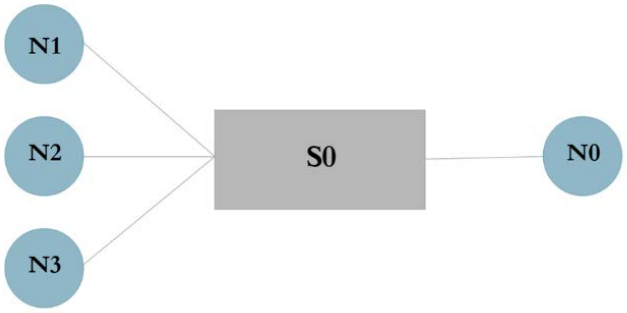
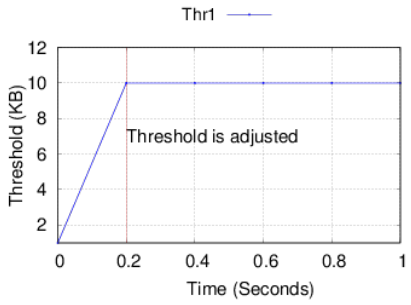


Figure 4. Experimental topology
Source: The authors

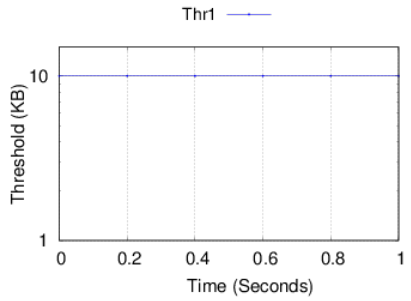
queue. The second configuration (10KB) is optimal as it keeps short flows in the high priority queue until its completion and demotes the long flows towards the low priority queue. The third configuration (1000KB) causes long flows to stay longer at the high priority queue, which hurts the FCT of short flows. The first and third configurations clearly induce the threshold mismatch problem discussed on Section 2.2.

Fig. 5 shows the convergence of the demotion threshold in the explicit threshold mismatch situations. The threshold converges to the value of 10KB, regardless of the initial value. With the threshold configured to the optimal value for this workload, we do not observe changes since the proposed approach detects that the threshold is already set to an adequate value.

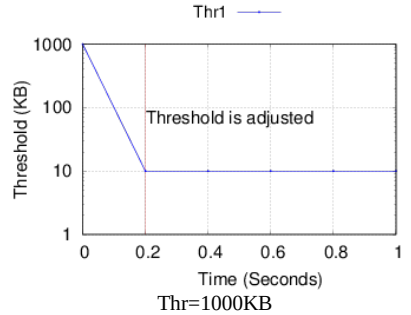
These initial experiments show that the adjustment approach operates adequately in terms of convergence of the demotion threshold. In the next section, we assess the effectiveness of the adjustment mechanism in terms of minimization of the average and tail FCT when compared to the static setting of the demotion threshold.



Thr=1KB



Thr=10KB



Thr=1000KB

Figure 5. Convergence of the demotion threshold in the proposed approach with explicit threshold mismatch
Source: The Authors

4.1.5. Minimization of the FCT

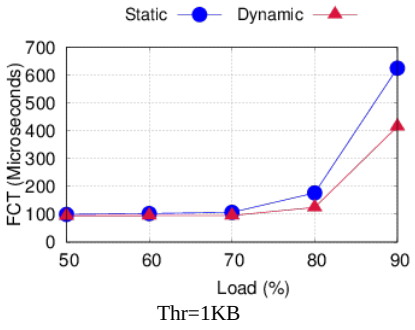
In this section, we present the results after assessing the FCT minimization. In this set of experiments, we compare the average and tail FCT for short and long flows, both in a static threshold configuration and with dynamic adjustment. These experiments consider different levels of link occupation ranging from 50% to 90%.

Short Flows

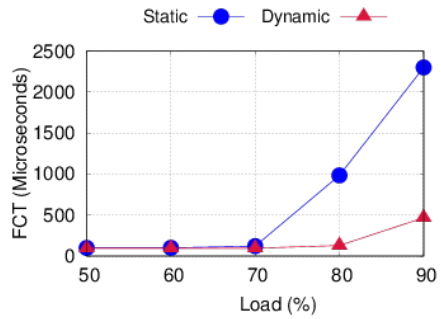
Fig. 6 presents the average FCT of the short flows for the configurations inducing threshold mismatch. It can be observed that the dynamic adjustment improves the metrics in these cases, especially at high traffic loads (higher than 70%). When the demotion threshold is set to 1KB, an average FCT reduction achieved is between 4.9% and 33.5%. When the threshold is set to 1000KB, the reduction is between 3.7% and 86%.

Fig. 7 presents the result for the tail (99th percentile) FCT. In this case, the dynamic adjustment also improves the performance by reducing the FCT in the situation of threshold mismatch, especially at high traffic loads. This improvement goes from 33.9% to 59.5% when the threshold is set to 1KB whereas it is between 51.3% and 93.9% when the threshold is set to 1000KB.

When the threshold is set to the optimal value for this workload (i.e. 10KB), the dynamic adjustment achieves results close to those obtained with the static setting. That is, the dynamic adjustment does not perform further adjustment as it detects that the threshold is already set to an appropriate value for the workload. Due to this, the plot presenting this situation has been omitted.



Thr=1KB



Thr=1000KB

Figure 6. Average FCT of the short flows
Source: The Authors

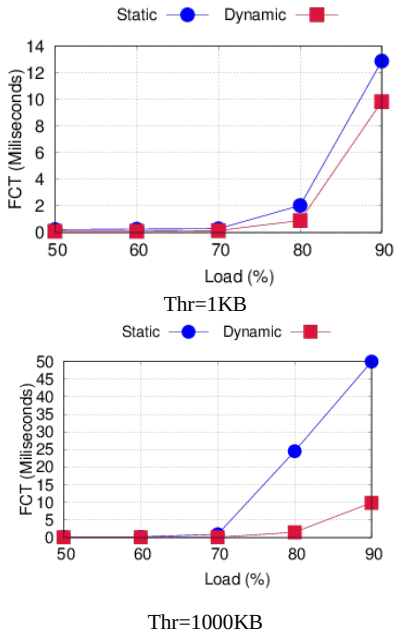


Figure 7. Tail FCT of the short flows
Source: The Authors

Long Flows

Fig. 8 presents the average and tail FCT for the long flows when the demotion threshold is set to 1000KB. The improvement in the metrics is more noticeable at high loads. For the average FCT, the improvement is between 27.1% and 46.7%. For the tail FCT, the improvement is between 24% and 45.8%.

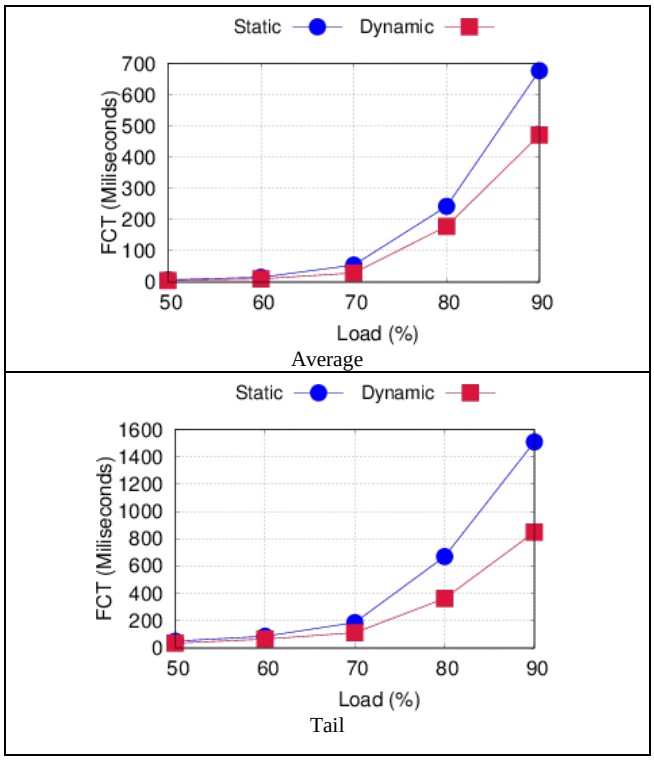


Figure 8. Average and Tail FCT of long flows with threshold set to 1000KB
Source: The authors

With respect to the other cases, the dynamic adjustment does not induce increments for either the average or the tail FCT. As observed in the previous result, the demotion thresholds converges to the size of the short flows. Therefore, it separates the short and the long flows avoiding them becoming mixed. Due to this separation, the long flows are not affected by the prioritization of the short flows.

5. Related work

We present a survey of literature addressing the problem of Information-Agnostic Flow Scheduling in a Data Center Networks context. Information-agnostic is the set of transport mechanisms that try to achieve the performance goals for cloud applications without detailed information for the flow scheduling. For a more extended review of the field's state-of-the-art, the interested reader can refer to [16,18].

HULL [3] and QJUMP [11] are approaches that aim to avoid the queuing delays that might affect especially short flows, but they do not actually require the size of the flow in advance. HULL leverages the concept of Phantom Queues that consists in simulating a queue associated with the egress port in switches. This simulation is implemented through a counter that is updated whenever a packet exits a link at high rate. Congestion signaling (e.g. ECN marking of DCTCP [1]) is associated to the counter instead of the physical switch queue, which can be configured to represent a speed slower than the actual physical link. Thus, some "bandwidth headroom" can be reserved to process high priority traffic. On the other hand, QJUMP focuses on reducing the network interference that throughput-oriented applications might cause by causing high queuing delays. QJUMP claims that applications dominated by short flows exhibit low latency variance and that low throughputs require higher priorities whereas applications with high latency variance and high throughput require lower priorities. In order to reduce this network interference, end hosts perform rate limitation in a non-intrusive way; this enables the applications to specify their required priorities. These approaches are agnostic in the sense that they do not require to know in advance the size of the flows in order to schedule them. However, they do not aim at improving the scheduling but reducing the occupation of switch buffers. That is to say, improving the FCT of short flows is not a primary goal of these works. In addition, in the case of QJUMP, it requires an additional API which implies modifications in the applications in order to use it.

Moreover, PIAS [6] and KARUNA [9] are agnostic flow scheduling approaches that aim to minimize the FCT of short flows by controlling the packet scheduling. They leverage MLFQ to achieve this goal. In PIAS, end host track the amount of bytes sent by each flow. According to this information, end hosts mark packets to match priority levels that are configured at data center switches. This packet tagging is performed following a MLFQ-like approach. That is, packets initially are marked so that they enter into the highest priority queue at switches. When the flow has sent more than a given amount of bytes (i.e. the demotion threshold), packets are marked with the following lower priority level. Hence, the more bytes a flow sends, the lower the priority value used to mark its

packets. This means that demotion thresholds are defined and configured at end hosts whereas switches simply perform priority scheduling. KARUNA [9] can be considered an extension of PIAS, which extends to consider deadline constrained flows. KARUNA divides flows into three categories: deadline-constrained, non deadline-constrained with known sizes, and non deadline-constrained with unknown sizes. Deadline-constrained flows are tagged to enter into the highest priority queue within datacenter switches. Non deadline-constrained flows with known size are tagged to enter into a specific queue. Non-deadline constrained flows with unknown size are sieved through the MLFQ scheduler in a similar way to the approach used in PIAS.

A challenging task associated to MLFQ-based scheduling is the derivation of a set of demotion thresholds that minimizes the average and tail FCT. PIAS and KARUNA calculate these thresholds based on traffic information consisting in the CDF of the flow sizes of the workload that will be present in the network. After the derivation of the thresholds, they are distributed and deployed at end hosts. Then, they use these thresholds to perform the packet tagging procedure [6, 9].

We consider that this approach includes some limitations. Since traffic in DCN presents time and space variations, a set of thresholds that minimizes the FCT of a given workload might not be adequate for a different one [19]. Also, the mentioned state-of-the-art approaches do not update their demotion thresholds dynamically. As previously explained, they need to be derived by considering the properties of the new workload, and they need to be manually deployed at end hosts.

6. Conclusions and future work

In this paper, we have proposed an approach to adjust the demotion thresholds of a MLFQ-based flow scheduling approach. We consider that this approach is an important first step in order to achieve a truly workload-agnostic flow scheduling solution. In this sense, our proposal aims at overcoming some of the limitations present in the state-of-the-art such as the requirement of information about the CDF of the workload that will be present in the network.

We performed a preliminary evaluation of our approach in a small experimental topology as a proof-of-concept of the approach's operation. We verified its adaption capability and confirmed that it provides a minimization of the FCT of short flows at high traffic loads. We also observed that the use of percentiles of the completed flow sizes provide a useful hint for the adjustment of the demotion thresholds.

For future work, we propose to design a smarter mechanism for the definition of the reference percentiles. An important improvement that could be introduced in our proposal is the capacity to determine from the observation of the traffic which would be adequate values to define the percentiles, in order to increase the accuracy of the scheduler.

Finally, an important task to consider for future work would be the integration of this threshold adjustment in a real software switch. Although recent literature presents the concept of programmable switches [8, 21], these devices still have open questions regarding elements such as the structure of their queuing systems. Thus, we consider that well-known software switches [17] with a more robust internal architecture design can provide a more mature starting point in order to develop this threshold adjustment.

Acknowledgments

This work was supported by: the Universidad Nacional de Colombia (UNAL) through the "Outstanding Postgraduate Student" scholarship from 2012 to 2016; Colciencias, through the "567 - National Doctorate Studies" scholarship from 2013 to 2017; the Universidad de Medellín (UDEM); and the Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil.

References

[1] Alizadeh, M., Greenberg, A., Maltz, D.A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S. and Sridharan, M., Data Center TCP (DCTCP). Proceedings of the ACM SIGCOMM 2010 Conference, New York, NY, USA, 2010, pp. 63-74. DOI: 10.1145/1851275.1851192

[2] Alizadeh, M., Javanmard, A. and Prabhakar, B., Analysis of DCTCP: stability, convergence, and fairness. Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, New York, NY, USA, 2011, pp. 73-84. DOI: 10.1145/1993744.1993753

[3] Alizadeh, M., Kabbani, A., Edsall, T., Prabhakar, B., Vahdat, A. and Yasuda, M., Less is more: trading a little bandwidth for ultra-low latency in the data center. Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, Berkeley, CA, USA, 2012, pp.19-19.

[4] Alizadeh, M., Yang, S., Sharif, M., Katti, S., McKeown, N., Prabhakar, B. and Shenker, S., pFabric: minimal near-optimal datacenter transport. Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, New York, NY, USA, 2013, pp. 435-446. DOI: 10.1145/2486001.2486031

[5] Bai, W., Chen, L., Chen, K., Han, D., Tian, C. and Wang, H., Information-agnostic flow scheduling for commodity data centers. Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation, Berkeley, CA, USA, 2015, pp. 455-468.

[6] Bai, W., Chen, L., Chen, K., Han, D., Tian, C. and Wang, H., PIAS: practical information-agnostic flow scheduling for commodity data centers. IEEE/ACM Transactions on Networking. 25(4), pp. 1954-1967, 2017. DOI: 10.1109/TNET.2017.2669216

[7] Benson, T., Akella, A. and Maltz, D.A., Network traffic characteristics of data centers in the wild. Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, New York, NY, USA, 2010, pp. 267-280. DOI: 10.1145/1879141.1879175

[8] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G. and Walker, D., P4: programming protocol-independent packet processors. SIGCOMM Comput. Commun. Rev. 44(3), pp. 87-95, 2014. DOI: 10.1145/2656877.2656890

[9] Chen, L., Chen, K., Bai, W. and Alizadeh, M., Scheduling mix-flows in commodity datacenters with Karuna. Proceedings of the 2016 ACM SIGCOMM Conference, New York, NY, USA, 2016, pp. 174-187. DOI: 10.1145/2934872.2934888

[10] Corbato, F.J., Marjorie Merwin-Daggett and Daley, R.C., An experimental time-sharing system. Classic Operating Systems. P.B. Hansen, ed. Springer New York. 2001, pp. 117-137.

[11] Grosvenor, M.P., Schwarzkopf, M., Gog, I., Watson, R.N., Moore, A.W., Hand, S. and Crowcroft, J., Queues don't matter when you can JUMP Them! Proc. NSDI, 2015.
[12] Hoganson, K. and Brown, J., Intelligent mitigation in multilevel feedback queues. Proceedings of the SouthEast Conference, New York, NY, USA, 2017, pp. 158-163. DOI: 10.1145/3077286.3077319
[13] Hong, C.-Y., Caesar, M. and Godfrey, P.B., Finishing flows quickly with preemptive scheduling. Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, New York, NY, USA, 2012, pp. 127-138. DOI: 10.1145/2342356.2342389
[14] Joy, S. and Nayak, A., Improving flow completion time for short flows in datacenter networks. 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 700-705. DOI: 10.1109/INM.2015.7140358
[15] Munir, A., Baig, G., Irteza, S.M., Qazi, I.A., Liu, A.X. and Dogar, F.R., Friends, not foes: synthesizing existing transport strategies for data center networks. Proceedings of the 2014 ACM Conference on SIGCOMM, New York, NY, USA, 2014, pp. 491-502. DOI: 10.1145/2740070.2626305
[16] Noormohammadpour, M. and Raghavendra, C.S., Datacenter traffic control: understanding techniques and trade-offs. IEEE Communications Surveys Tutorials. 99, pp. 1-1, 2017. DOI: 10.1109/COMST.2017.2782753
[17] Pfaff, B., Pettit, J., Koponen, T., Jackson, E.J., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P. et al., The design and implementation of Open vSwitch. NSDI, 2015, pp. 117-130
[18] Rojas-Cessa, R., Kaymak, Y. and Dong, Z., Schemes for fast transmission of flows in data center networks. IEEE Communications Surveys Tutorials. 17(3), pp. 1391-1422, 2015. DOI: 10.1109/COMST.2015.2427199
[19] Roy, A., Zeng, H., Bagga, J., Porter, G. and Snoeren, A.C., Inside the social Network's (datacenter) Network. Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, New York, NY, USA, 2015, pp. 123-137.
[20] Sivaraman, A., Cheung, A., Budiu, M., Kim, C., Alizadeh, M., Balakrishnan, H., Varghese, G., McKeown, N. and Licking, S., Packet transactions: high-level programming for line-rate switches. Proceedings of the 2016 ACM SIGCOMM Conference, New York, NY, USA, 2016, pp. 15-28. DOI: 10.1145/2785956.2787472
[21] Sivaraman, A., Kim, C., Krishnamoorthy, R., Dixit, A. and Budiu, M., DC.P4: programming the forwarding plane of a data-center switch. Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, New York, NY, USA, 2015, pp. 2:1-2:8. DOI: 10.1145/2774993.2775007
[22] Sivaraman, A., Subramanian, S., Alizadeh, M., Chole, S., Chuang, S.-T., Agrawal, A., Balakrishnan, H., Edsall, T., Katti, S. and McKeown, N., Programmable packet scheduling at line rate. Proceedings of the 2016 ACM SIGCOMM Conference, New York, NY, USA, 2016, pp. 44-57. DOI: 10.1145/2934872.2934899

S.A. Gutiérrez, received the BSc. Eng in Computer Science degree in 2008 from Universidad de San Buenaventura, the MSc., Eng in Computer Science in 2011 from Universidad Nacional de Colombia, sede Medellín, and he is currently candidate to the degree of PhD in Computer Science from Universidad Nacional de Colombia. From 2000 to 2012, he worked as System Administrator and later as Senior Engineer, managing, operating, supporting and developing telecommunication solutions at different scales and criticality levels. He is currently full time lecturer at Universidad de Medellin, and previously he has been also part time lecturer at Universidad de San Buenaventura and Universidad Nacional de Colombia. His research interests include computer networks, security in computer networks, data center networks, software defined networks, programmable devices and application of pattern recognition and machine learning to computer networks.
ORCID: 0000-0003-2880-4601

M. Barcellos, is a CNPq 1D level researcher, senior member of the Association for Computing Machinery (ACM) and the Brazilian Computer Society (SBC). He received BSc. and MSc. degrees in Computer Science from Federal University of Rio Grande do Sul, Brazil (1989 and 1993, respectively) and PhD degree in Computer Science from University of Newcastle Upon Tyne (1998). Between 1999 and 2008, worked as a professor at UNISINOS university, spending 2003-2004 at BT Research Labs working for University of Manchester on high-performance multicast transport. Since 2008 Prof. Barcellos has been with the Federal University of Rio Grande do Sul (UFRGS), where he is an associate professor. He has authored many papers in leading vehicles related to computer networks and security, also serving as PC member, PC chair and General Chair. Prof. Barcellos was the elected chair of the Special Interest Group on Computer Security of the Brazilian Computer Society (CESeg/SBC) 2011-2012. He is the appointed Director of Diversity and Outreach in ACM SIGCOMM for the term 2017-2021. His current research interests are Internet measurements, programmable data planes, and security aspects of those networks.
ORCID: 0000-0002-1505-6408

J.W. Branch, received the BSc. in Mining Engineering, his MSc. degree in System Engineering and his PhD. in Engineering from Universidad Nacional de Colombia, Campus Medellín, in 1995, 1997 and 2007 respectively. Currently, He is full professor in the Department of Computer Science at Universidad Nacional de Colombia, Campus Medellín. His main research interests encompass computer vision, image processing and their applications to the industry field and applications of pattern recognition in the field of Computer Networks.
ORCID: 0000-0002-0378-028X