Poster: Fast Scheduling for Load Migration in Distributed Network Controllers

Sepehr Abbasi-Zadeh, MohmmadAmin Beiruti, Yashar Ganjali
Department of Computer Science, University of Toronto, Canada
{sepehr, beiruti, yganjali}@cs.toronto.edu

Zhenhua Hu
Huawei Research, Canada
zane.hu@huawei.com

***Abstract*—As network traffic and conditions change, the load on different instances of control plane changes. To ensure various control applications can operate continuously and efficiently, we need to migrate the load among controller instances. For this, we need a migration schedule that minimizes the overall migration time while ensuring the quality of service and controller resource constraints. In this poster, we show this problem is NP-hard, and show how a heuristic algorithm performs close to the best existing solution with orders of magnitude reduction in scheduling time.**

I. Introduction

The performance of a distributed software-defined network controller is a function of how its load is distributed among various controller instances. With rapidly changing traffic and network conditions, we need internal mechanisms to adjust the load and shift it inside the control plane. Typically, load migration is performed by changing the master controller associated with a given switch, thus shifting the traffic away from the original master, and towards the new master [1], [2].

Unfortunately, even if we assume we know the desired load distribution for a given network[1], we cannot simply shift the load associated with controller instances by migrating switches all in parallel. Load migration requires careful scheduling for two reasons: First, switch migration protocols require significant processing and memory resources. Migrating many switches away from or towards a given controller instance can lead to significant performance reduction in that instance, and consequently degrade the performance of the entire control plane. For that reason, we need to respect *controller resource constraints* during migration.

Second, during the handover process of a given switch between two controller instances, there is a short time period that the control plane pauses processing events related to that particular switch. These messages are processed after the handover is complete. In the absence of any coordination and switch migration scheduling, network services will experience interruptions that impact the QoS provided to end users. We call these the *QoS constraints*.

The load migration problem in the context of SDN controllers has been studied by Beiruti and Ganjali [3] before. They model the problem as an Integer Linear Programming (ILP) problem, taking controller resource and QoS constraints as input, and generating a schedule with the minimum number of rounds.

In this poster, we provide a heuristic algorithm for load migration scheduling. We formulate our problem as a *Vector Bin Packing* (VBP) instance, and use fast greedy algorithms to solve the problem. From the theoretical point of view, we show that this optimization problem is NP-Hard, and also show the approximation factor of our solution.

Our experiments show that VBP provides schedules that are very close to ILP in terms of the number of rounds. In particular, in 95% of our experiments, VBP and ILP lead to the same number of rounds, and there is only a difference of one round between the two in the remaining cases. On the other hand, VBP has a superior run-time, finishing in only a few milliseconds, while ILP taking several hours in some situations (and growing exponentially).

II. Problem Formulation

Let us assume that a set of k migrations are planned as $M = \{m_1, m_2, \ldots, m_k\}$. Each migration m_i can be described by a tuple $(sw_i, src_i, dst_i, w_i, G_i)$, where the first three components show the switch to be migrated, the switch's current controller, and its destination controller, respectively. The fourth component, i.e., w_i, is the assigned weight for this migration, and it implies the imposed load (e.g., memory or computational capacity requirements) of the migration protocol on controller instances[2]. Finally, G_i is a subset of $G = \{g_1, g_2, \ldots, g_\ell\}$, where each member of G represents a QoS group in our network. In other words, G_i contains all the service groups for which switch sw_i is providing a QoS guarantee.

As discussed before, we cannot run all migrations in a single round due to two main constraints. First, every controller instance c_j has resource limitations of a_j, which caps the sum of the weight of simultaneous migrations that the controller instance can handle. For a set of controllers C, we define the set C^a as the controller resource constraints.

Similarly, for each QoS group $g \in G$, there is a bound α_g that limits the weight of concurrent migrations in this group. We refer to the set of α_g values as G^α. Now, we are ready to define the migration problem.

Problem 1: Let G be the set of all QoS groups, and let G^α be their corresponding QoS constraints. Further, let C be

[1]This can be as simple as distributing load evenly among controller instances, or a more complex function of network load, and network application requirements.

978-1-7281-6992-7/20/$31.00 ©2020 IEEE

[2]We note that this parameter can be a multidimensional weight vector to capture different resource constraints separately.

the set of all controllers (as well as their corresponding C^a) involved in a given set of migrations $S = \{m_1, m_2, \ldots, m_k\}$. Find the minimum integer value $R \leq |S| = k$ such that there exists a partitioning of S into R partitions $S_1, \ldots, S_R$ that does not violate the following constraints:

- Controller constraints:
  $\forall i \in [R], \forall c_j \in C \quad \sum_{\{m \in S_i | src_m = c_j \vee dst_m = c_j\}} w_m \leq a_j.$
- QoS constraints:
  $\forall i \in [R], \forall g \in G \quad \sum_{\{m \in S_i | g \in G_m\}} w_m \leq \alpha_g.$

III. Proposed Solution

In this section, we first show that migration scheduling problem is an NP-Hard problem. Then, we propose a formulation of the problem that enables us to solve the migration scheduling problem efficiently using a fast heuristic algorithm. In the following section, we show this algorithms runs orders of magnitude faster than existing solution with similar results.

Complexity. The first question that we are interested in, is the complexity of our problem on hand. By a way of reduction, we can show that any instance of the Bin Packing problem can be cast as a migration scheduling problem, and therefore, as the Bin Packing problem is known to be NP-Hard, the migration scheduling problem is also inherently NP-Hard. For this reduction we relax the QoS constraints in our formulation and view each item of the Bin Packing problem as a migration from a dummy controller to another controller instance. Any solution to the migration scheduling problem results in a solution to the original Bin Packing problem.

New Solution. As mentioned earlier, each migration m_i can be described as a tuple $(sw_i, src_i, dst_i, w_i, G_i)$. For each m_i, we assign a *migration vector* of length $|C| + |G|$. Each component of this vector either represents a controller in C, or a QoS group in G. We set all the elements of this vector to 0, except the fields that correspond to src_i, dst_i, and all the QoS groups that belong to G_i. For these elements, we simply use the weight w_i (or in a more general form, if w_i is a vector, we decompose it to single elements and fill the appropriate resource elements in the migration vector). Similarly, we construct a *constraint vector* representing the available resources. With these two main sets of vectors, we can simply run the FirstFit algorithm to obtain the final schedule of the migrations, as explained next.

The FirstFit algorithm (which we refer to as Greedy as well) is one of the classic solutions for the Vector Bin Packing (VBP) problem and it works as follows. Initially, we allocate a free bin and iterate over the set of all items in an arbitrary order. For each item, we go through the list of existing bins, and add the item to the first bin that can fit. If the item does not fit in any of the existing bins, we create a new bin for this item, and continue with the next item in the list.

As a generalization of the normal Bin Packing, VBP is harder, and in fact it is APX-Hard even for the case $d = 2$. Interestingly, the FirstFit algorithm is known to produce a $(|C| + |G| + 1)$-approximation for the problem, i.e., the total number of the rounds (or bins in the Bin Packing terminology) that this algorithm produces is not larger than $(|C| + |G| + 1) \times OPT$, where OPT is the optimal number of the required rounds.

This approximation factor is the best we have achieved for the current VBP formulation of our scheduling problem. An interesting problem is to determine whether there are any simplifying/system-specific assumptions that make our problem simpler than the general VBP problem. We might be able to achieve better approximation algorithms for our migration scheduling problem in that case. We leave this as a future work.

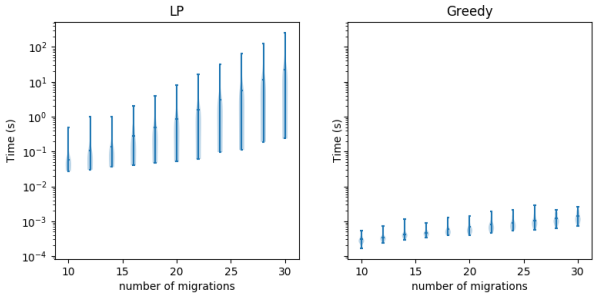

Fig. 1. Comparison of the migration scheduling time.

IV. Experiments

To evaluate our new approach, we use the ILP solution presented in [3] as the baseline. We generate 100 sets of random migration plans, each consisting of 30 switch migrations. We assume that there are 5 destination controllers and 20 QoS groups in total in our system. For each switch migration, we randomly choose its destination controller as well as a subset of size d from the QoS groups, where $d \sim \mathcal{N}(6, 6)$. The other weights and limitations are also chosen randomly from proper normal distributions with the constraint that none of the weights can surpass the resource limitations.

We measure the run-time of the scheduling algorithm, and the number of rounds generated by the algorithm as our evaluation metrics. Interestingly, the performance of our solution is similar to ILP, i.e., it generates schedules that match the ILP solution 95% of the time. For the remaining experiments, our solution is off by only 1 round.

Figure 1 shows the running time of the ILP solution (left) and our Greedy VBP solution (right) in logarithmic scale as the number of migrations changes from 10 to 30. We also set a timeout as ILP solution sometimes take hours to finish, especially as the number of migrations grows. Throughout the experiments, our solution finishes in a few milliseconds and the growth seems to be linear with the number of migrations.

References

[1] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. R. Kompella, "ElastiCon; an elastic distributed SDN controller," in *2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. New York, NY, USA: IEEE, 2014, pp. 17–27.

[2] M. A. Beiruti and Y. Ganjali, "Load migration in distributed sdn controllers," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[3] ——, "Migration scheduling in distributed sdn controllers," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 2019, pp. 1–2.