Deterministic Dynamic Network-Based Just-in-Time Delivery for Distributed Edge Computing

Subhadeep Sahoo[1, 2], Ning-Hai Bao[2], Sébastien Bigo[1], Nihel Benzaoui[1]

[1] Nokia Bell Labs, 91620 Nozay, France, subhadeep.sahoo4@gmail.com
[2] School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China

Abstract For distributed Edge Computing, we propose to leverage the deterministic data delivery in optical Determinist Dynamic Networks to jointly optimize compute and network resource allocation. Numerical results show that our DDN-based approach can accelerate the application total execution time by 50- 70% from baseline approaches.

Introduction

The ultimate Edge Computing (Fig.1) is an aggregate of distributed Edge Data Centers (DC) acting as a single fabric used to deliver time-sensitive and computation-hungry applications requiring parallelized processing.

In this environment, compute and network resource are generally allocated independently which might lead to a high delivery and buffering delay or an inefficient use of computing resources[1]; e.g. a dataflow might take precedence over other flows and be delivered by some network resources to reach a busy CU and wait there, while these network resources would better be used to deliver another dataflow to an idle CU before. All dataflows should be delivered by the network *just-in-time* to be processed by a free CU[2], i.e. eliminate buffering delay at CUs. Just-in-time delivery relies on a precise knowledge of the CUs' busy time and the network delivery delay. Most past reports leveraging some form of delivery time estimation[3][4][5] left an optimization gap that we propose to close.

In previous work, we proposed the Deterministic Dynamic Network (DDN), a high bandwidth optical scheduled time-slotted network for Edge DC interconnexion. DDN allows for deterministic performance (low latency, low jitter) and fast connection reconfiguration[6]. The time slotted and scheduled nature of DDN opens the possibility to accurately predict the network latency at any time, on any path in the Edge DC. We leverage this latency prediction to enable the just-in-time delivery. Our approach jointly optimizes network and compute resource allocation to minimize workflow total completion time and maximize Edge DC utilization.

Partition-aggregate workflow

Referring to Fig.2, the conventional technique to parallelize a process is to use a partition-aggregate workflow as follows: (A) Import phase: the application request is *partitioned* into smaller jobs to be executed simultaneously over a group of compute units (CU) called mappers. (B) Shuffle phase: The output data chunks are sent from the mappers to a second group of CUs, called reducers, where they are *aggregated* to deliver the final data output. (C) Replication phase: the final aggregated data is duplicated to different servers to increase reliability[7]. The workflow latency is dominated by the shuffle and replication phases: (i) the delay to exchange data between CUs over the network, (ii) the buffering time and (iii) the computing time at CUs.

DDN-based just-in-time delivery

In a DDN network, bandwidth, latency (sub-ms) and jitter (sub-µs)[8] are guaranteed through slot reservation. Slots of a few µs are allocated by a central scheduler in a dynamic fashion (sub-ms)[9]. Once slots are allocated for a given data flow, no further switching is applied to that flow and its delivery delay is fixed.

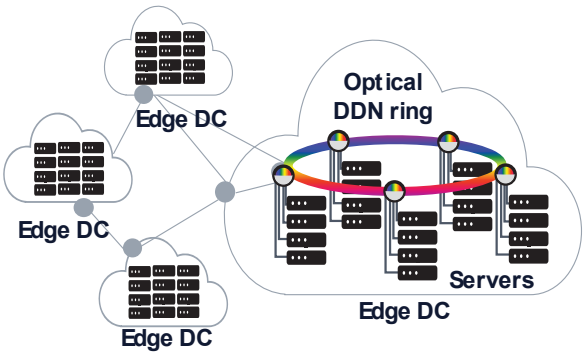

Fig.1: Distributed DDN-based Edge DC

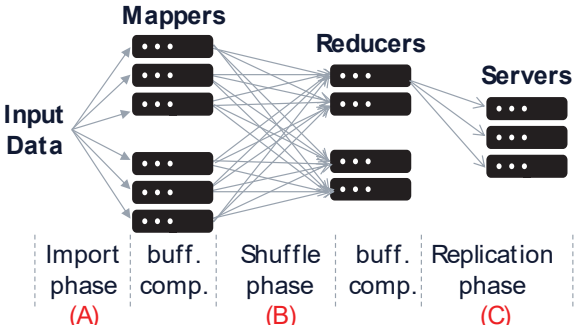

Fig.2: Partition-aggregate workflow

Our DDN-based approach for distributed computing comes in three phases: (i) we select mappers and reducers such that the compute and network load is balanced and workflow completion time is minimum; (ii) we perform a first optimization to steer workflows to execute simultaneously and decrease the total completion time of a workflow group; (iii) we perform a second optimization to enable the just-in-time delivery of all dataflows to CUs.

(i) First, we determine the available computing capacities of CUs and the bandwidth availabilities on the communication paths between them. The groups of CUs with maximum available computing capacity and bandwidth are selected as mappers and reducers. Then, during the replication stage, we select the path and the slot allocation between the reducer and the destination servers such that we minimize the delivery delay and deploy the path with the highest available bandwidth.

(ii) We leverage the determinism of DDN to predict the delivery delay of a flow between a mapper and reducer on a single path. The delivery delay of a path depends on the amount of available bandwidth, which is directly proportional to the number of available timeslots on the path, and the slot distribution over the reservation calendar. The completion of a workflow requires that the reducers receive all dataflows contributing to the workflow output. Hence, the workflow completion time cannot be smaller than the largest delivery delay among all the dataflows belonging to this workflow, which we refer to as *threshold delivery delay*. Our proposition is to decrease the number of slots (bandwidth) allocated for all other dataflows belonging to this same workflow, to make their delivery delays equal to the *threshold delivery delay*. We devote the released slots for the other queued workflows. In the replication stage, the same approach is followed.

This strategy maximizes the parallel processing of workflows, which contributes to minimization of the total completion time for the group of workflows.

(iii) We organize just-in-time delivery of flows. First, using slot scheduling information, we calculate the delivery delay of a given dataflow in a given path, and retrieve the busy time of a CU (i.e. the remaining time before a CU is free again). Then CUs are categorized by classes of similar busy time. We define the *threshold CU busy time* as the largest busy time of that group. Finally, we pair a dataflow to a CU, such that the *threshold delivery delay* and *threshold CU busy time* are equal. Thus, we guarantee no buffering of the dataflow at the CU.

This strategy minimizes the total completion time per workflow, hence per group of workflows, and maximizes the compute resources utilization efficiency.

Numerical Analysis

In this section, we evaluate the performance of our DDN-based just-in-time delivery in terms of Total Completion Time (Total-CT) for a group of workflows. The Total-CT is set by the completion time by workflow, the number of processes running in parallel and the compute resources utilization.

We compare the DDN-based approach with today's-typical approach that delivers dataflows as-fast-as-possible (AFAP) over the network. AFAP is a combination the scheme in phase (i) for CUs selection, and a greedy scheme for slots allocation. For each flow, AFAP grabs the whole available bandwidth (slots) on its path.

For the evaluation we simulate DDN-based single Edge DC (Fig.1), where 10Gb/s servers (CU) are connected using a 10-node DDN ring with four 10Gb/s channels. For a fair comparison we use the ring topology for the evaluation of both DDN-based and AFAP approaches. In each simulated scenario, the number of servers per DDN node is equal to the scenario total number of CUs divided by 10. The Compute delay of a workflow is not included in the Total-CT. The input data (Fig.2) for each workflow is randomly generated between 2 to 5 GB. In the Shuffle (B) phase, we cap the data rate of all-mappers-to-all-reducers dataflows to 1Gb/s. In the replication phase, we assume that 10% of the initial input data is replicated to 3 servers[7].

We evaluate the first optimization (ii) of the DDN-based approach in a scenario where all CUs are all fully available. We assume 40 CUs with 4 pairs of mappers and reducers. In Fig.3, we report the evolution of the Total-CT w.r.t. the number of workflows for both DDN-based just-in-time delivery and AFAP. We observe that for both approaches the Total-CT is increasing with the number of workflows. In both the approaches, the completion time of each workflow is minimized

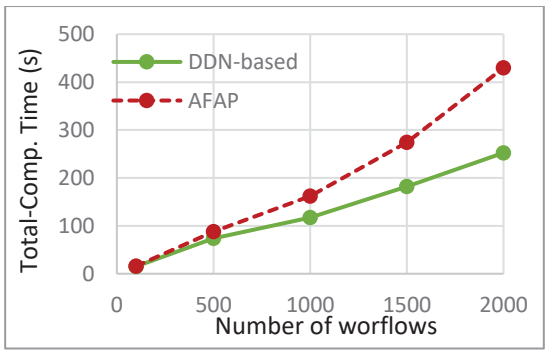

Fig.3 Total-CT when all CUs are free

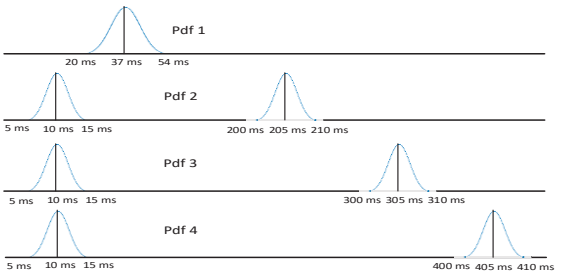

Fig.4 CU busy time distributions.

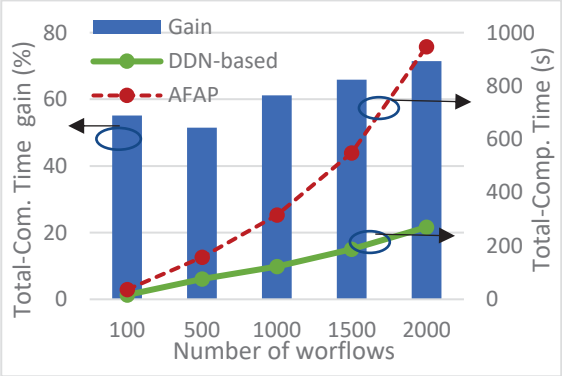

Fig. 5 Total-CT when all CUs are busy

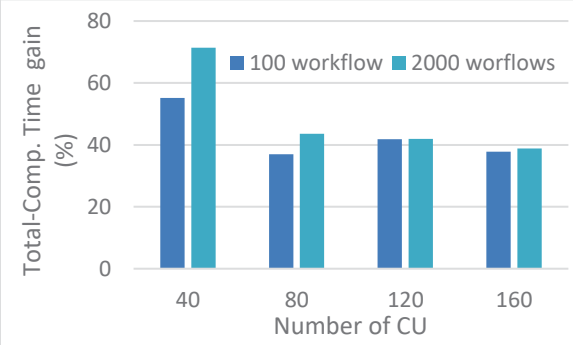

Fig.6 Evolution of the gain on Total-CT with no. of CUs

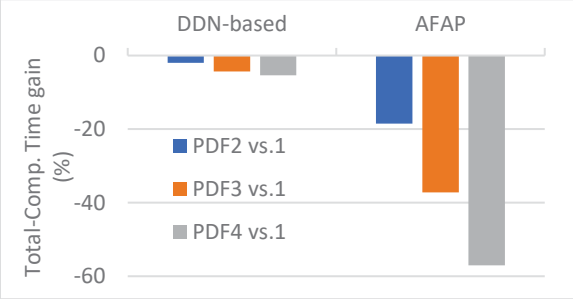

Fig.7 Evolution of the gain on Total-CT for different PDF of CU busy time

through the optimization of delivery delay per dataflow. As the DDN-based approach performs an additional optimization (ii) that equalizes the delivery delay of all dataflows contributing to the same workflow w.r.t. the worst delay among them and reallocates the released network resources for the execution of additional workflows, the DDN-based approach outperforms AFAP in terms of Total-CT for a group of workflows.

We evaluate the second optimization (iii) of the DDN-based approach in a scenario where all CUs are partly busy. Here also, we assume 40 CUs with 4 pairs of mappers and reducers, and a distribution of the CU busy time, (i.e. the remaining time before a CU is free again) which follows PDF1 in Fig.4. In Fig.5, we report the evolution of the Total-CT w.r.t. the number of workflows for both DDN-based just-in-time delivery and AFAP, as well as the relative gain of DDN-based approach over AFAP. As the DDN-approach synchronizes the arrival of a dataflow just in time when the CU is free, it avoids dataflow buffering at CU and increases the utilization of CUs. For this reason, we observe in Fig.5 that the DDN-based approach outperforms AFAP with a gain of 50 to 70% on the Total-CT.

In Fig.6, we evaluate the impact of increasing the number of CUs on the Total-CT gain for a small (100) and large (2000) number of workflows. For 40, 80, 120 and 160 CUs we take, respectively, 4, 8, 12 and 16 pairs of mappers and reducers. The result shows that even if the gain on Total-CT decreases, its drop is found to depart from a linear decrease by a significant amount, e.g., if we compare 40 and 160 CUs cases, we quadruple the number of CUs while the gain drops at worst by only a factor of 1.75.

In the last scenario, we evaluate the impact of the distribution of the CUs busy time on the Total-CT. We consider an edge DC hosting multiple applications with very heterogenous needs in computing power, where CU busy times varies significantly from one application to the next. We assess the Total-CT using different busy time distributions (Fig.4) for both DDN-based and AFAP approaches. In Fig.7, we show the gain on Total-CT for 2000 number of workflows using PDF2, 3, and 4 w.r.t to PDF1. Fig.7 shows, for AFAP, a strong degradation of the gain on Total-CT with the increase of CU busy time heterogeneity. With the DDN-based just-in-time delivery the gain is almost unchanged, as an evidence of its optimal efficiency, even in a heterogenous application environment.

Conclusion

We proposed to leverage the deterministic data delivery in our optical DDN to jointly optimize compute and network resource allocation in an edge DC. Our approach increases the edge DC utilization by 1) decreasing the completion time per workflow, 2) increasing number of workflows running in parallel, hence decreasing the total completion time for a group of workflows and 3) avoiding idle compute resources. We achieved a gain up to 70% with respect to the baseline approach. The gain that we showed for a single edge DC can be extended to distributed edge computing.

References

[1] R. Nirmalan and K. Gokulakrishnan, “Survey on Map Reduce Scheduling Algorithms in Hadoop Heterogeneous Environments”, in Proc. 3rd International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, Nov. 2018.

[2] A. Ousterhout, A. Belay, and I. Zhang, “Just In Time Delivery: Leveraging Operating Systems Knowledge for Better Datacenter Congestion Control”, in Proc. 11th USENIX Conference on Hot Topics in Cloud Computing, Jul. 2019.

[3] R. Monisha and K. R. Sekar, “Heterogeneous Map Reduce Scheduling Using First Order Logic”, in Proc. 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, May. 2018.

[4] B. P. Lohani, A. Singh and V. Bibhu, “A hybrid optimization approach using Evolutionary Computing and Map Reduce Architecture”, in Proc. 2019 International Conference on Advances in Computing and Communication Engineering (ICACCE), Tamil Nadu, India, Apr. 2019.

[5] M. S. Shanoda, S. A. Senbel and M. H. Khafagy, “JOMR: Multi-join optimizer technique to enhance map-reduce job”, in Proc. 9th International Conference on Informatics and Systems, Cairo, Egypt, Dec. 2014.

[6] S. Bigo, “Overturning the Eight Fallacies of Distributed Computing with the Octopus Edge Network”, in Proc. Optical Fiber Communication Conference (OFC), San Diego, California, United States, Mar. 2020.

[7] Y. Tang *et al.*, “OEHadoop: Accelerate Hadoop Applications by Co-Designing Hadoop With Data Center Network”, *IEEE Access*, vol. 6, pp. 25849–25860, May. 2018.

[8] N. Benzaoui *et al*., “Deterministic Dynamic Networks (DDN)”, Journal of Lightwave Technology, vol. 37, no. 14, pp. 3465–3474, May. 2019.

[9] M. Szczerban *et al.*, “Real-time Control and Management Plane for Edge-Cloud Deterministic and Dynamic Networks,” IEEE/OSA Journal of Optical Communication and Networking, vol. 12, no. 11, pp. 312–323, Aug. 2020.