Master's thesis project proposal

Marco Micera
Politecnico di Torino
marco.micera@gmail.com

Introduction

This document briefly describes my thesis proposal as well as a possible project workflow. This Master's thesis will be written at the Technische Universität Darmstadt, Germany, during a six-month Erasmus+ exchange and under the supervision of Prof. Patrick Eugster [†], M.Sc. Marcel Blöcher [†] and Prof. Fulvio Risso [‡].

1 Motivation

Data centers distributed systems can nowadays make use of in-network computation to improve several factors: DAIET [4] inventors claim to achieve a 86.9%-89.3% traffic reduction by performing data aggregation entirely in the network data plane. Other solutions like NETCHAIN [1] and INCBRICKS [3] let programmable switches store data and process queries in order to cut end-to-end latency. It is now even possible to provide guarantees to applications with specific requirements: for instance, CLOUDMIRROR [2] enables applications to reserve a minimum bandwidth.
For the time being, it seems that there is still no valid resource allocation algorithm that takes into account the presence of a network having a data plane that is (in part o completely) capable of basic in-network processing (INP) operations. The objective of this thesis would be to model and evaluate an API through which applications can ask for resources in a data center exploiting INP capabilities while providing guarantees (e.g., bandwidth).

[†] Distributed Systems Programming Group, Technische Universität Darmstadt

[‡] Computer Networks Group, Politecnico di Torino

2 Possible workflow

Here I list a possible workflow I could be following during this project: however, last-minute changes could drive me to vary it, hence it may not be strictly followed.

2.1 Literature research

A first step could consist in studying all the material (e.g., scientific papers, books, etc.) needed to have a better comprehension of INP and guarantee provisioning in data centers.

2.2 Resource modeling

As mentioned in § 1, existing applications exploit different INP capabilities, such as data aggregation, data storage, query processing, etc.; applications could also require different guarantees, such as minimum bandwidth. Current cloud computing platforms allow clients to request server resources only. In order to include INP and guarantees in the resource management, these two features need to be modeled as well. It might be convenient to come up with a list of generic groups of INP applications describing all common constrains and needed resource types shared by similar INP application types: for instance, a *data aggregation group* could require a certain amount of local memory in each INP-capable network node needed by the client application and a hierarchical topology for those nodes. In this way, client applications could specify all their needed resource types and requirements more easily, just by selecting the most appropriate generic group.

2.3 Defining an API for resource acquisition

Finding a good API through which client applications can request both server and INP resources is not a trivial

task: it is not clear a priori how the API granularity and the resource allocation algorithm's performance are correlated. For instance, the more detailed the API is, the more difficult it could be for the resource allocation algorithm to find a feasible solution so that the application resources could be correctly deployed.

2.4 Resource allocation algorithm development

As soon as the client application has (i) specified a generic group and (ii) requested all the needed resources through the API , an allocation algorithm could start trying to acquire all requested resources, while providing the specified guarantees. During this part of the project, an allocation algorithm should be developed so that, at the end, the allocation request will be either successfully approved or denied (in case at least one hard requirement could not have been satisfied).
One thing to notice is that the resource allocation algorithm's result strongly depends on the characteristics of the physical network, such as the types of operations supported by programmable switches, the bandwidth of the physical links and the network topology itself.

2.5 Evaluation

The final step of this thesis would consists in the evaluation of the previously-mentioned resource allocation algorithm (§ 2.4), and hence on the modeled API on which the algorithm is based on. During this step, the Omega simulator [5] could be adapted in such a way that it also considers INP resources, and the evaluation could then be done by means of this modified tool. The API performance could be measured by calculating different factors, such as (i) the allocation time or (ii) the ratio between the number of accepted allocation requests and the total amount of requests issued by the application.

References

[1] X. Jin, X. Li, H. Zhang, N. Foster, J. Lee, R. Soulé, C. Kim, and I. Stoica. Netchain: Scale-free sub-rtt coordination. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 35–49, Renton, WA, 2018. USENIX Association.

[2] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, and P. Sharma. Application-driven bandwidth guarantees in datacenters. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 467–478, New York, NY, USA, 2014. ACM.

[3] M. Liu, L. Luo, J. Nelson, L. Ceze, A. Krishnamurthy, and K. Atreya. Incbricks: Toward in-network computation with an in-network cache. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, pages 795–809, New York, NY, USA, 2017. ACM.

[4] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis. In-network computation is a dumb idea whose time has come. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets-XVI, pages 150–156, New York, NY, USA, 2017. ACM.

[5] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes. Omega: flexible, scalable schedulers for large compute clusters. In *SIGOPS European Conference on Computer Systems (EuroSys)*, pages 351–364, Prague, Czech Republic, 2013.