Thesis summary

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

1 Info

Title: Data center resource management for in-network processing
Graduand name: Marco Micera
Supervisors: Prof. Fulvio Risso †, Prof. Patrick Eugster ‡, M.Sc. Marcel Blöcher ‡
Research areas: cloud computing, distributed systems, data centers

2 Thesis purpose

Nowadays there exist several In-Network Processing (INP) solutions that allow tenants to improve their application performance in terms of different metrics: DAIET [7] inventors claim to achieve an 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 to cut end-to-end latency; CloudMirror [2] allows client applications to specify bandwidth and high availability guarantees. 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 supports (partially or completely) INP. This thesis has mainly two goals: (i) model and evaluate an Application Programming Interface (API) through which applications can ask for INP resources and (ii) discuss the importance of a scheduler which can reject INP requests and propose their server-only equivalent when needed (e.g., high switch utilization).

3 Personal contribution

3.1 System design

The thesis introduces a system design comprising all steps needed to translate resources expressed using the tenant-side model to physical ones so that they could eventually be placed. The whole system design is depicted in Figure 1. The system also introduces the concept of composite: "template describing a high-level logical component. It can be made out of other composites and/or logical resources". Composites can be translated into logical resources by means of a *template database*.

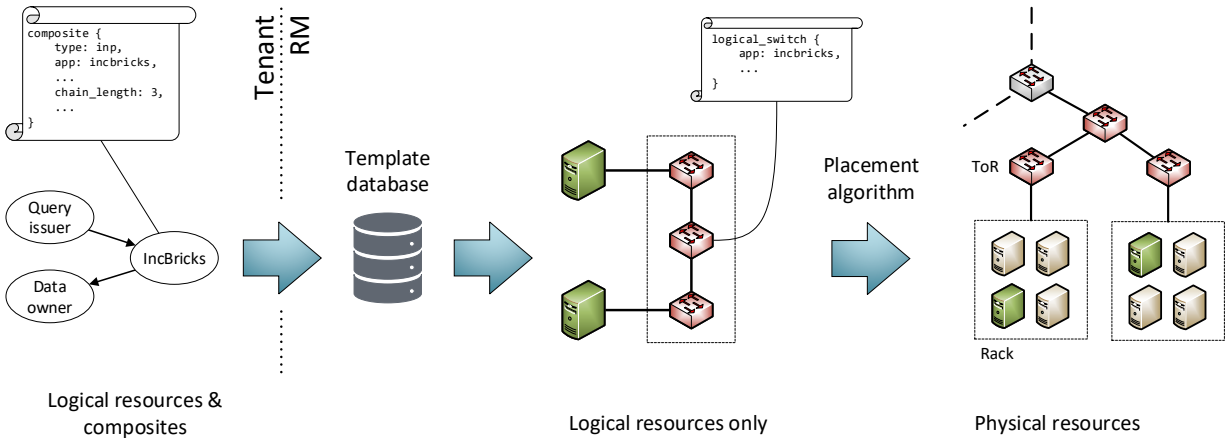

Figure 1: From the tenant-side model to physical resources

† Computer Networks Group, Politecnico di Torino, Italy
‡ Distributed Systems Programming Group, Technische Universität Darmstadt, Germany

3.1.1 Composites translation methods

Composites can have multiple properties specifying some application requirements or constraints. A first approach called *passive mapping* would require tenant applications to explicitly express internal composites' properties that **directly** affect their equivalent expressed in terms of just logical resources. This, of course, increases the expressiveness of the tenant application with the cost of making the interface more complex to use. With the opposite approach (*active mapping*), tenant applications do not have to specify internal composites' properties, but instead more abstract performance goals. These high-level composites' goals will be then translated by the Resource Manager (RM). This approach simplifies the interface exposed to tenant applications by not letting them taking care of internal composites' properties that might be unknown to developers.

3.2 Resource model

Alongside the system previously described (subsection 3.1), the design chapter of this thesis introduces a resource model proposal that is capable of describing existing INP solutions as well as future ones: the extended-Tenant Application Graph (eTAG). The eTAG allows tenant applications to specify (i) all types of composites and logical resources, (ii) different bandwidth demands for different entities and (iii) any kind of network topology. An example of an eTAG is depicted in Figure 2.

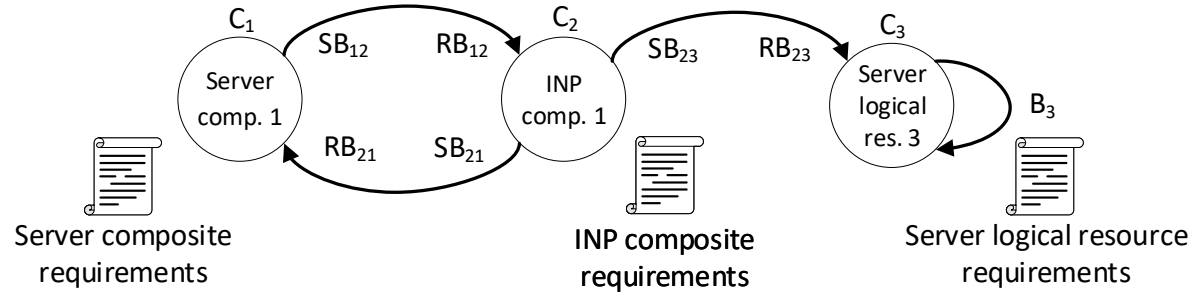

Figure 2: The eTAG proposed model

The introduction of composites in the model delegates their translation into logical resources to the RM. The *template database* allows tenants to express INP composite requirements in terms of high-level properties.

3.3 Generic groups

The *template database* takes care of translating composites into a set of logical resources in order for the placement algorithm to work. Ideally, the template database should map individual INP solutions to their equivalent made out of logical resources only, but this may be not feasible as the number of INP solutions will likely increase in the future. That is to say, this approach is not scalable. INP solutions may generally seem very different from each other as they pursue different goals, but grouping them based on common aspects might be a way to decrease the number of entries of a template database. That said, there is no unique way of categorizing INP solutions: for instance, one could group them based on their final purpose, on their logical architecture, and so on. This thesis tries to do this based on the latter aspect resulting in two groups: in-network aggregation and in-network storage.

4 Obtained results

4.1 Simulation

A simulator very similar to the Omega's [8] *lightweight simulator*[†] has been built up from scratch for this evaluation in Scala [4] so it could support multiple resource dimensions, physical switch resources, composites and more. The scheduler used for this evaluation uses a simple greedy placement algorithm that cycles through all physical resources every time a job needs to be scheduled. It tries to allocate as many tasks as possible on the same physical machine simply by (i) checking the amount of available numerical resources and (ii) performing

[†] Available at github.com/google/cluster-scheduler-simulator

the *properties* check for switch tasks (i.e., whether a physical switch supports a specific INP composite). Experiments ran on Dell C6420 servers on CloudLab [6], simulating a 3-days long workload. The simulated data center architecture is a fat-tree with 4 pods.

These simulations focus on the average resource utilization in the data center for all dimensions. Different simulations vary in the ratio between the number of tenant requests including INP composites over the overall amount of requests.

4.2 Results

Results show how easily one kind of physical resource can become the bottleneck for the other, and that the ratio of incoming INP requests is a key factor for the overall data center utilization. Ideally, all types of physical resources in data center should be fully utilized. When plotting the less relatively-used resource dimension as a function of the percentage on INP requests, a peak around a certain percentage of INP requests appears. This remarks the importance of a scheduler which can reject INP requests and propose their server-only equivalent when needed (e.g., high switch utilization).

4.3 Conclusions

Based on the achieved results and on the analysis of network-aware RMs, the conclusions chapter aims to foresee some features that an ideal fully-INP aware Resource Manager should have, as well as some open problems in the INP resource management field.

4.3.1 Fully INP-aware RM features

Conjunct placement [5] considers both server and logical switch resources during allocation, but it does not place them during the same scheduling round. The drawback of not scheduling these two kinds of resources conjunctly causes the placement algorithm to derive a sub-optimal placement. Ideally, an RM should allocate a job considering its server and logical switch resources at the same time, i.e., during the same placement round.

INP alternatives Results showed an underutilization of physical resources depending on the number of tenant requests containing INP composites. To maximize the (relative) minimum physical resource utilization, the RM would need to adjust the ratio of INP requests over servers-only ones. With the assumption that physical switch resources can become the bottleneck of whole the system more rapidly (e.g., less switches than servers in a fat-tree with $k > 5$ or for the scarceness of switch resources in general), RMs should be able to propose alternatives to INP composites made out of logical server resources or server composites only, as soon as physical switch resources become heavily utilized. Those alternatives could be stored, for instance, in the same *template database* used to translate INP composites into a set of logical switch resources.

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] M. Odersky, P. Altherr, V. Cremet, B. Emir, S. Maneth, S. Micheloud, N. Mihaylov, M. Schinz, E. Stenman, and M. Zenger. An overview of the scala programming language. Technical report, 2004.

[5] M. G. Rabbani, R. Esteves, M. Podlesny, G. Simon, L. Z. Granville, and R. Boutaba. On tackling virtual data center embedding problem. In *IM 2013: IFIP/IEEE International Symposium on Integrated Network Management*, 2013.

[6] R. Ricci, E. Eide, and C. Team. Introducing cloudlab: Scientific infrastructure for advancing cloud architectures and applications. *; login:: the magazine of USENIX & SAGE*, 39(6):36–38, 2014.

[7] 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.

[8] 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.