Data center resource management for in-network processing

Marco Micera

March 31, 2020

Politecnico di Torino, Technische Universität Darmstadt

Introduction

"**Modern Internet services**, *such as search, social networking, and e-commerce,* **critically depend on high-performance key-value stores**. *Rendering even a single web page often requires hundreds or even thousands of storage accesses.*"

NetChain[1] authors

"As the number of compute elements grows, and the need to expose and utilize higher levels of parallelism grows, **it is essential to [...] focus on developing architectures that lend themselves better to providing extreme-scale simulation capabilities**."

SHArP[2] authors

[1][Jin et al., 2018], [2][Graham et al., 2016]

In-Network Processing (INP)

- INP refers to the technique of **offloading parts of the computation to network devices** (e.g., programmable switches, network accelerators, middleboxes, etc.), hence reducing the load on servers
- Advantages:
  1. Serve network requests on the fly with low latency
  2. Reduce data center traffic and mitigate network congestion
  3. Save energy by running servers in a low-power mode
- Few solutions out there already: Daiet[1], SHArP[2], NetChain[3], IncBricks[4]

[1][Sapio et al., 2017], [2][Graham et al., 2016], [3][Jin et al., 2018], [4][Liu et al., 2017]

Thesis goals

Problem statement

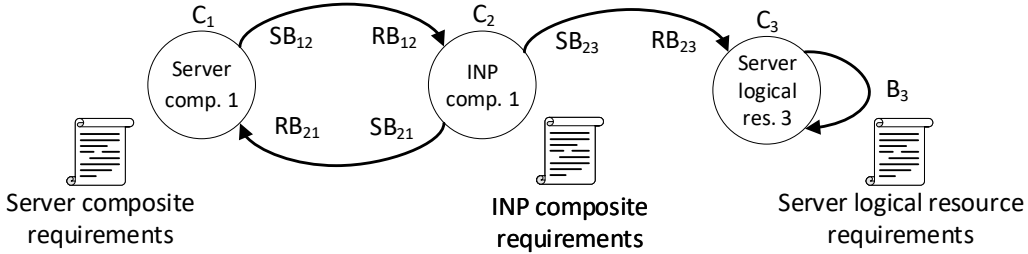
For the time being, it seems that there is still no Resource Manager (RM) that takes into account the presence of a network having a data plane that supports (partially or completely) INP

Goals

1. Propose a system that allows tenant applications to ask for INP resources
2. Discuss the importance of a scheduler which can reject INP requests and propose their server-only equivalent when needed (e.g., high switch utilization)

Design

The Extended-Tenant Application Graph (eTAG)



- A composite is a template that describes high-level logical components
  - It can be of two types:
    - Server (e.g., “web server”, “database”, ...)
    - INP (e.g., “IncBricks caching system”, “NetChain locking system”, ...)
  - It can be made out of
    - Logical resources
    - Other composites

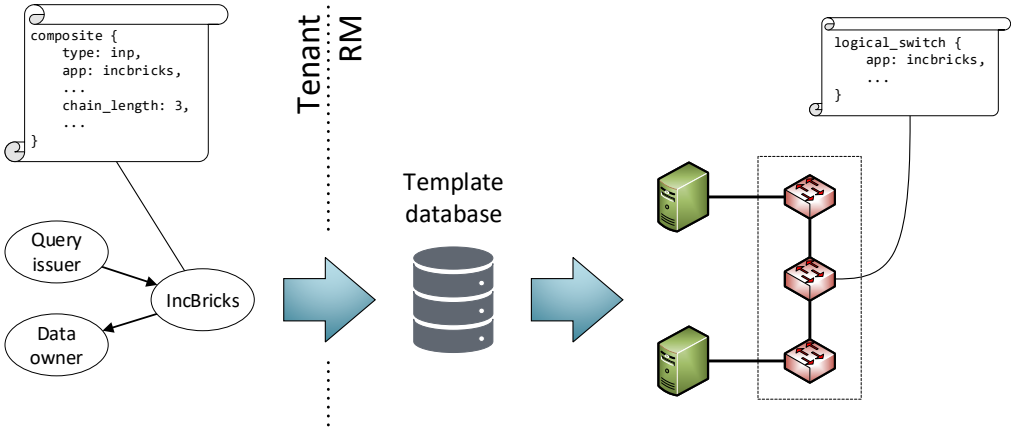
Generic groups

- In-network **storage**
  - Switches must
    - dedicate part of their local memory to store a distributed map
    - form a chain
  - IncBricks[1], NetChain[2]
- In-network **data aggregation**
  - Switches must
    - form a tree whose root is connected to data consumers and whose leaves are connected to data producers
    - dedicate part of their local memory to store a key-value map
    - be able to perform basic operations on data, such as writing and hashing
    - wait for all its children to send aggregated data
  - Daiet[3], SHArP[4]

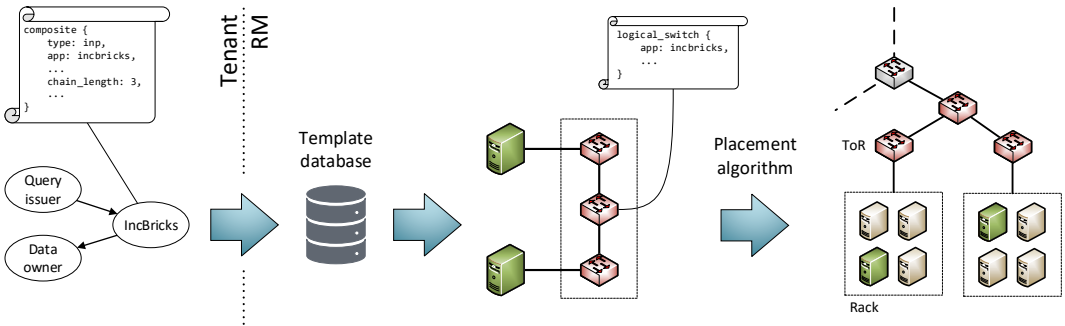
[1][Liu et al., 2017], [2][Jin et al., 2018], [3][Sapio et al., 2017], [4][Graham et al., 2016]

Mapping composites to logical resources

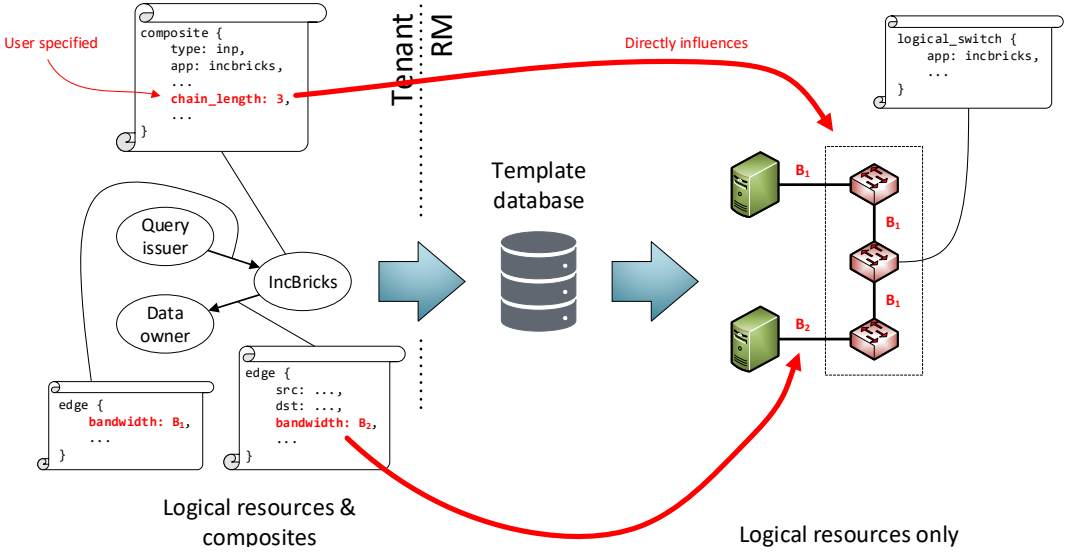
- The *template database* maps composites (or generic groups) to their equivalent made out of just logical resources



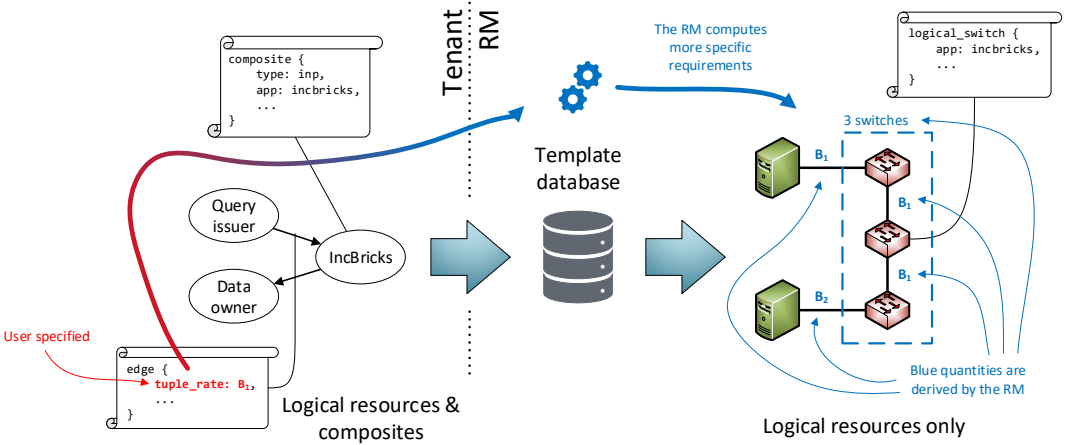
The whole picture



1st approach: passive template mapping



2nd approach: active template mapping



Evaluation

Simulation 1/2

- Simulator built from the ground up
  - Inspired by Omega's[1] *lightweight simulator*[2]
  - Supports multiple resource dimensions, switch resources, and composites
- Simulated data center physical architecture: fat-tree with 4 pods
  - Switches have *properties* (e.g., list of supported INP solutions)
- 3 days-long randomly-generated workload
  - Job properties (e.g., requirements, requests' interarrival time, etc.) are sampled from exponential distributions
- Simple greedy scheduler

[1][Schwarzkopf et al., 2013], [2]available at github.com/google/cluster-scheduler-simulator

Simulation 2/2

- The template database contains two entries for the previously-mentioned generic groups
  - In-network storage (switch chain)
  - In-network data aggregation (switch tree)
- Sweep: percentage of requests including INP composites
- Server Tasks Cutback (STC): the reduction of server tasks once an INP solution is introduced

$$STC = \frac{\#server\ tasks\ without\ INP}{\#server\ tasks\ with\ INP}$$

Results 1/3

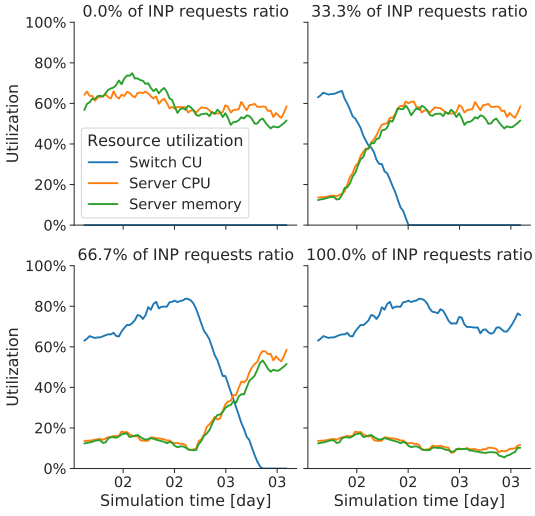


Figure 1: physical resource utilization for different amounts of INP requests

Results 2/3

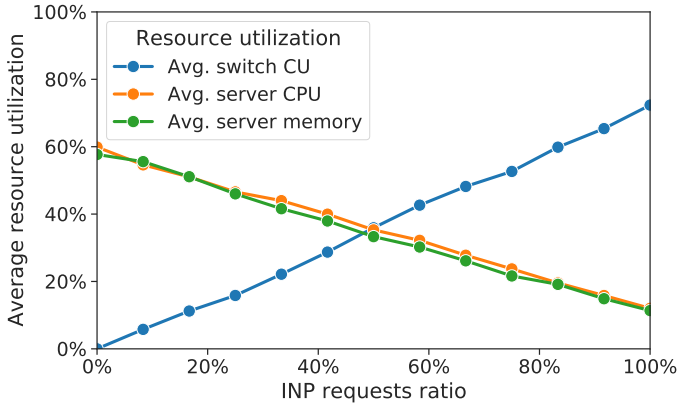


Figure 2: Average resource utilization as a function of the INP requests ratio

Results 3/3

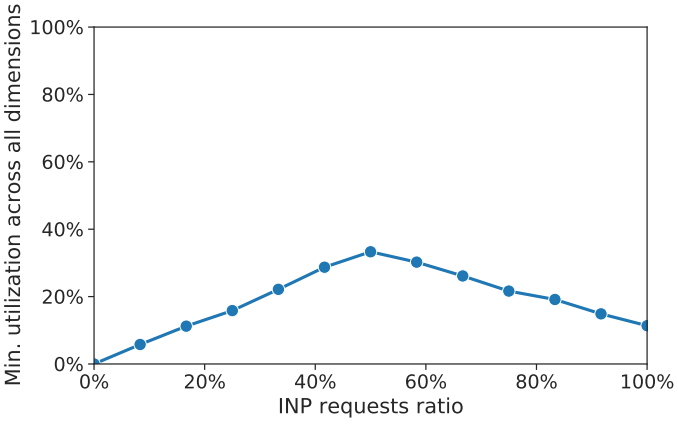


Figure 3: Minimum resource utilization across all dimensions

Conclusions

Conclusions

Fully INP-aware RM features

- Conjunct placement of server and switch resources
- INP alternatives

Open problems

- Accurately determine STC values for all INP solutions
- Determine the number of needed switch tasks for INP solutions
- Differentiate INP solutions based on their life cycle (e.g., short-term batch jobs vs. long-term services)

Questions?

Thank you

Bibliography i


Graham, R. L., Bureddy, D., Lui, P., Rosenstock, H., Shainer, G., Bloch, G., Goldenerg, D., Dubman, M., Kotchubievsky, S., Koushnir, V., et al. (2016).
Scalable hierarchical aggregation protocol (sharp): a hardware architecture for efficient data reduction.
In *Proceedings of the First Workshop on Optimization of Communication in HPC*, pages 1–10. IEEE Press.


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

Bibliography ii


Liu, M., Luo, L., Nelson, J., Ceze, L., Krishnamurthy, A., and Atreya, K. (2017).
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. ACM.


Sapio, A., Abdelaziz, I., Aldilaijan, A., Canini, M., and Kalnis, P. (2017).
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. ACM.


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