Towards Elastic Performance Guarantees in Multi-tenant Data Centers

Carlo Fuerst⋆, Stefan Schmid⋆†, Lalith Suresh⋆, Paolo Costa‡

⋆ Technische Universität Berlin, † Telekom Innovation Laboratories, ‡ Microsoft Research

Introduction

Cloud-based applications, including MapReduce and scale-out databases, generate a significant amount of network traffic and a considerable fraction of their runtime is due to network activity. Unfortunately, as reported in previous studies [1], in existing cloud infrastructures, the bandwidth available to the tenants varies significantly over time, even within the same day. Given the time spent in network activity by these applications, this variability has a non-negligible impact on the application performance [3].

Related Work: Virtual Cluster Embeddings

Several systems have been proposed which leverage admission control and support explicit bandwidth reservations to overcome this problem. [1, 5] Many of these systems offer *virtual cluster* abstractions, which provides the tenants with the illusion of having their own dedicated network: A virtual cluster offers the tenant the illusion for all her *Compute Units (CUs)* to be attached to a single non-oversubscribed switch with a minimum bandwidth b guaranteed. A virtual cluster $\mathrm{VC}(n, b)$ has two parameters: n, the number of (identical) CUs in the cluster, and b, the bandwidth reservation from each CU to the virtual switch.

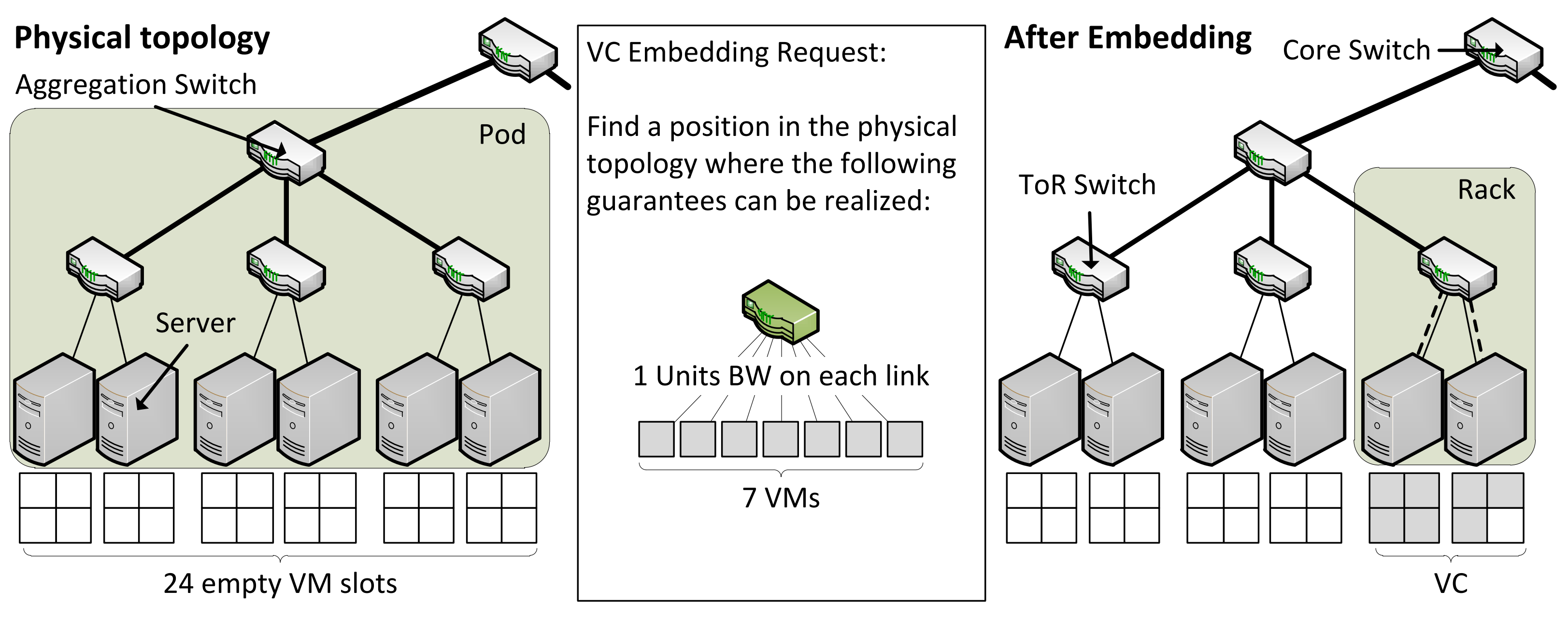

Figure 1: Embedding of a virtual cluster with $n = 7$ and $b = 1$.

However, *all* existing solutions providing absolute bandwidth guarantees are based on *offline reservations schemes* [1, 2, 5]: they require that tenants announce the entire resource reservation schedule *ahead of time*, i.e., at job submission time.

Problems of offline resource reservations

We argue that in most cases it is very hard to accurately estimate application resource needs ahead of time, rendering offline reservation schemes inadequate. To highlight this point, we demonstrate that even with the same workload and a dataset of the same size being re-executed, it is difficult to predict how a job progresses over time. Concretely, we conduct an "idealized" experiment wherein we run a Hadoop cluster in our OpenStack-based testbed; environments such as Amazon EC2 are far more noisy. Figure 3 indicates the variance in job completion times across the runs: a range of 150 seconds.

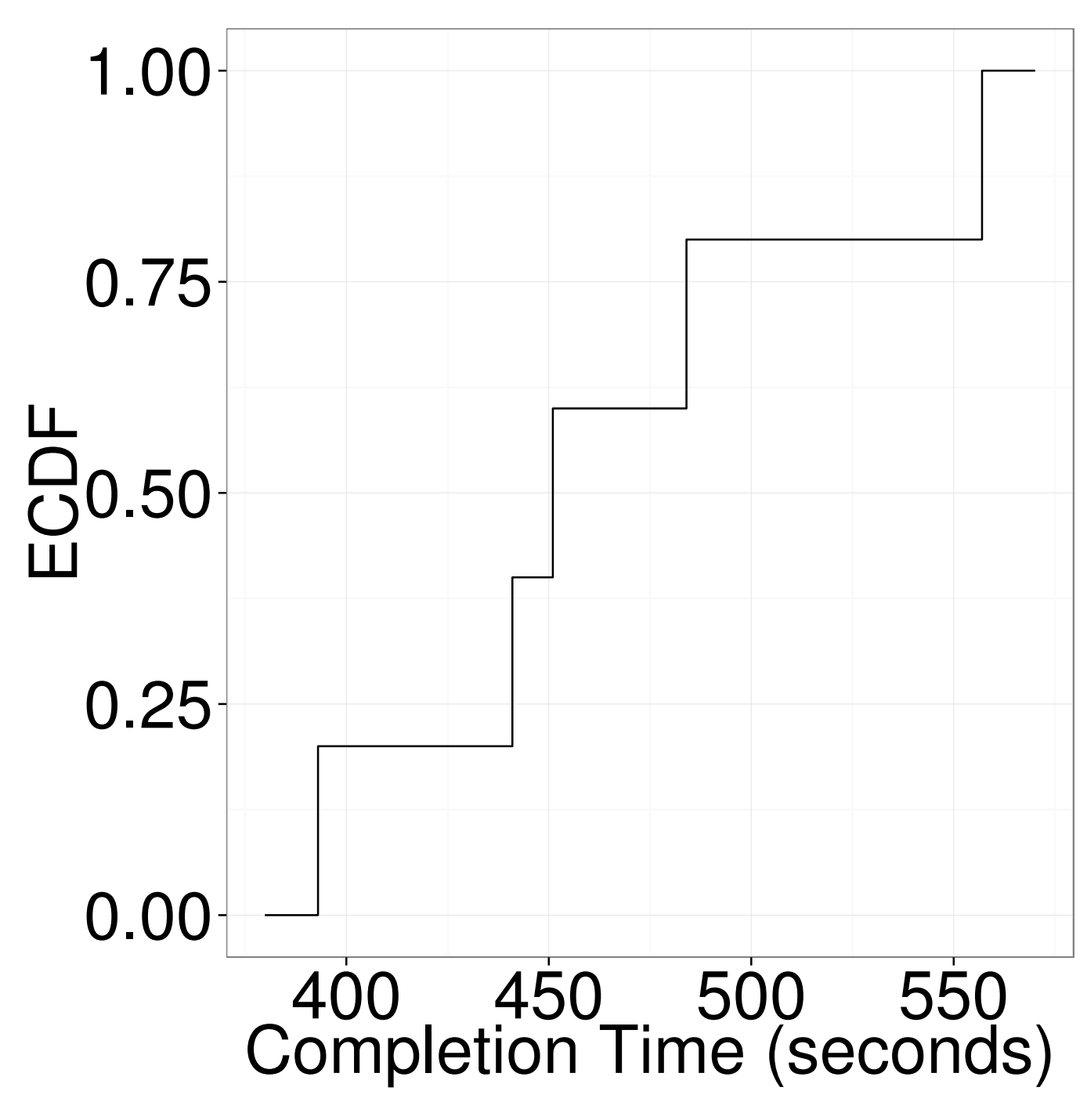

Figure 2: Execution unpredictability.

Algorithmic improvements to the state of the art

Previous algorithms for the virtual cluster embedding problem have resorted to weaker notions of optimality such as *spartial locality* [5]. While these notions in general generate good solutions, there are some instances, in which they have significantly worse footprints than necessary.

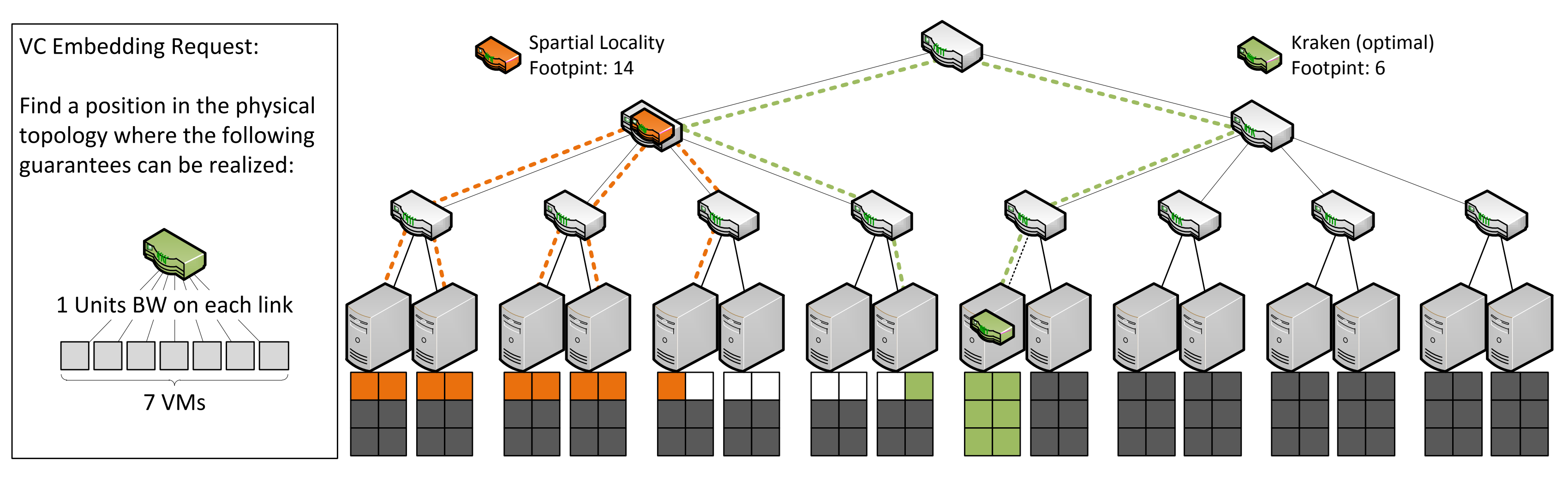
VC Embedding Request:
Find a position in the physical topology where the following guarantees can be realized:
1 Units BW on each link
7 VMs
Spartial Locality Footpint: 14
Kraken (optimal) Footpint: 6

Figure 3: Pitfalls of spartial-locality

Our Contribution: Kraken

An *online* resource reservation system to *adjust* the guarantees of existing virtuals *at runtime*.

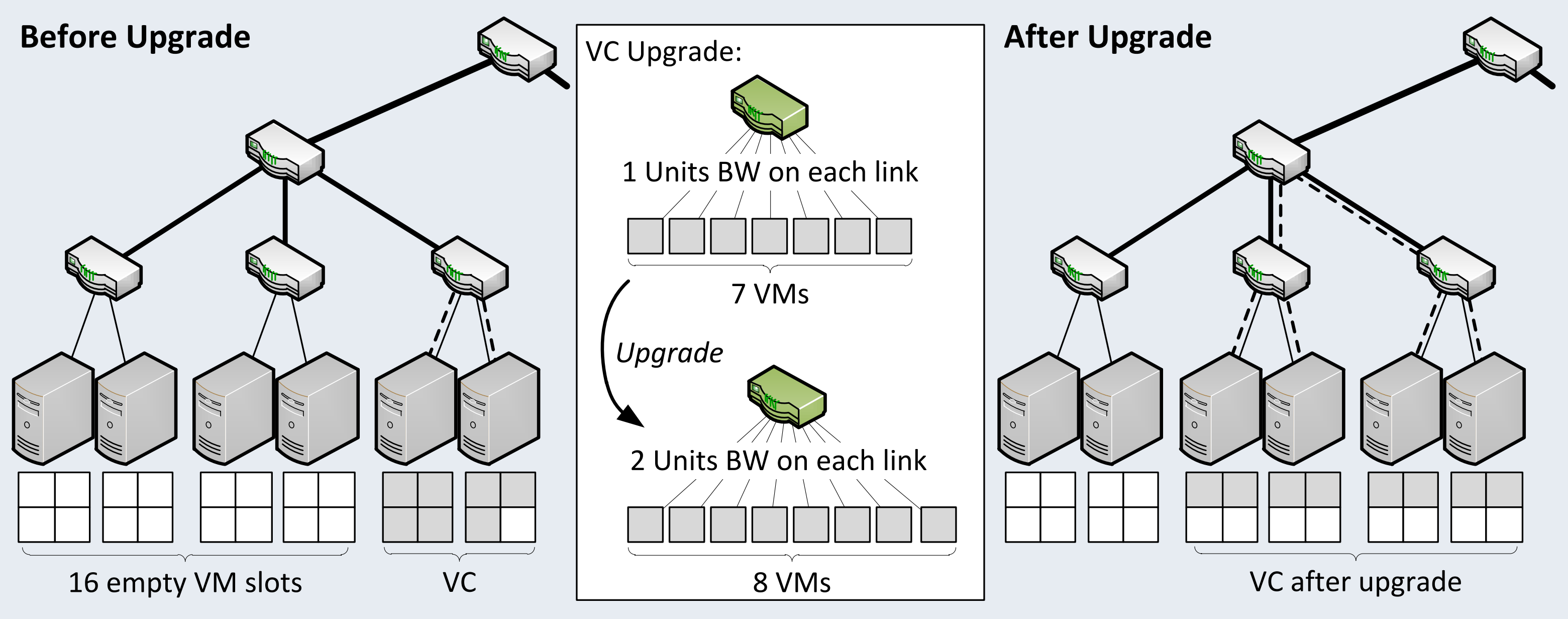

Figure 4: Example: upgrade of virtual cluster requiring migration.

- Optimal: Kraken *provably* minimizes the number of reconfigurations (migrations) necessary to modify a virtual cluster and the *footprint* (bandwidth consumption) of a virtual cluster embedding.
- Low complexity: The runtime depends linear on the size of the physical network and quadratic on the size of the virtual cluster.
- First polynomial optimal algorithm: The embedding algorithm can also be used to embed virtual clusters from scratch, showing that the virtual cluster embedding problem is *not* NP-hard. A comprehensive study on the complexity of different variants of the virtual cluster problem will appear in July [4].

Preliminary Evaluation

To demonstrate a basic version of Kraken (without support for elastic computations or migrations), we implemented a simple controller for Hadoop-YARN. The controller runs inside a virtual machine and uses the Linux `tc` utility in order to make bandwidth reservations. We instrumented the Hadoop source code such that tasks inform the controller prior to executing a shuffle. If there is spare bandwidth to be allocated, the controller increases the corresponding endpoints bandwidth reservation. Once the shuffle is completed, the Hadoop framework informs the controller of the same in order to release its reservations.

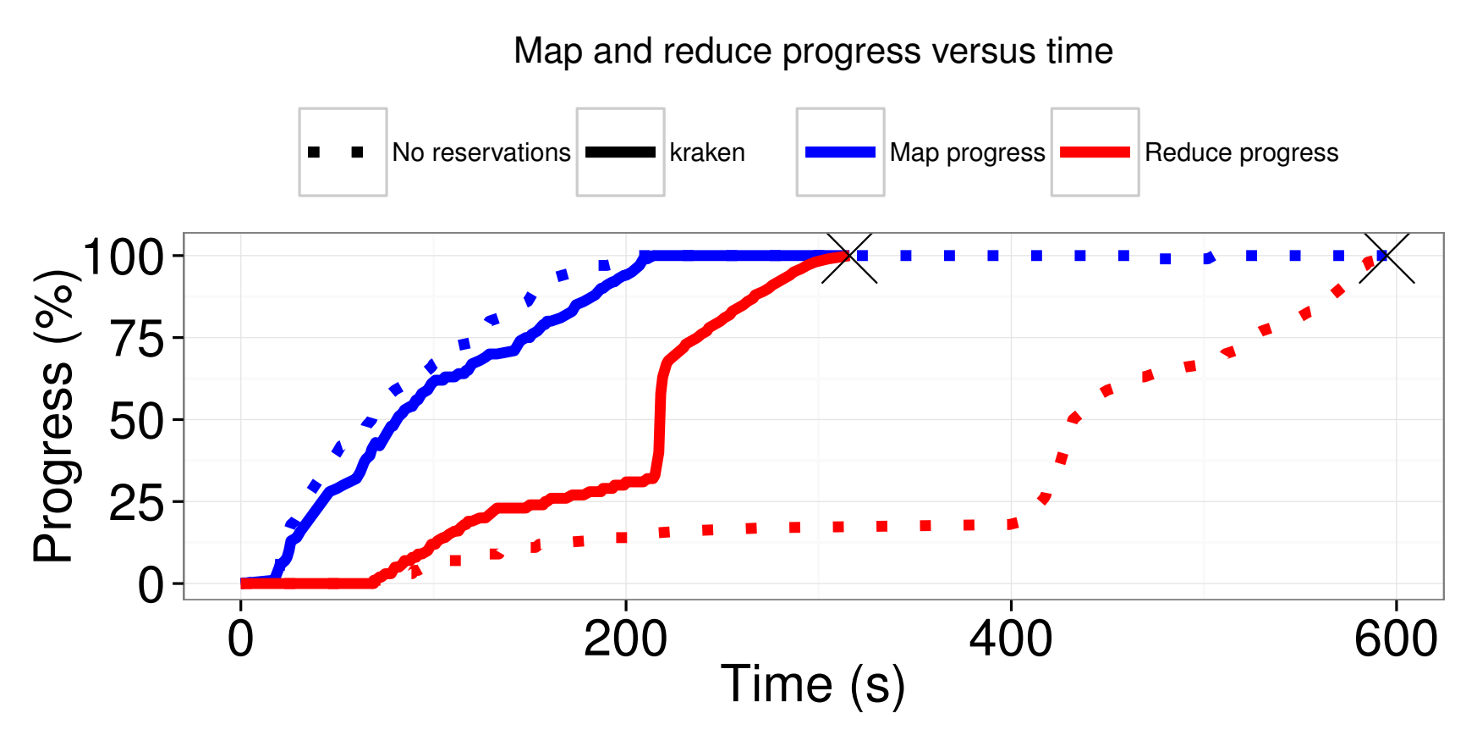

Figure 5: Progress of a TeraSort job.

References

[1] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron.
Towards predictable datacenter networks.
In *Proc. ACM SIGCOMM*, 2011.

[2] Chuanxiong Guo, Guohan Lu, Helen J. Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang.
SecondNet: A data center network virtualization architecture with bandwidth guarantees.
In *Proc. ACM CoNEXT*, 2010.

[3] Jeffrey C. Mogul and Lucian Popa.
What we talk about when we talk about cloud network performance.
SIGCOMM CCR, 42(5):44–48, 2012.

[4] Matthias Rost, Carlo Fuerst, and Stefan Schmid.
Beyond the stars: Revisiting virtual cluster embeddings.
SIGCOMM CCR, July 2015.

[5] Di Xie, Ning Ding, Y. Charlie Hu, and Ramana Kompella.
The only constant is change: incorporating time-varying network reservations in data centers.
In *Proc. ACM SIGCOMM*, 2012.