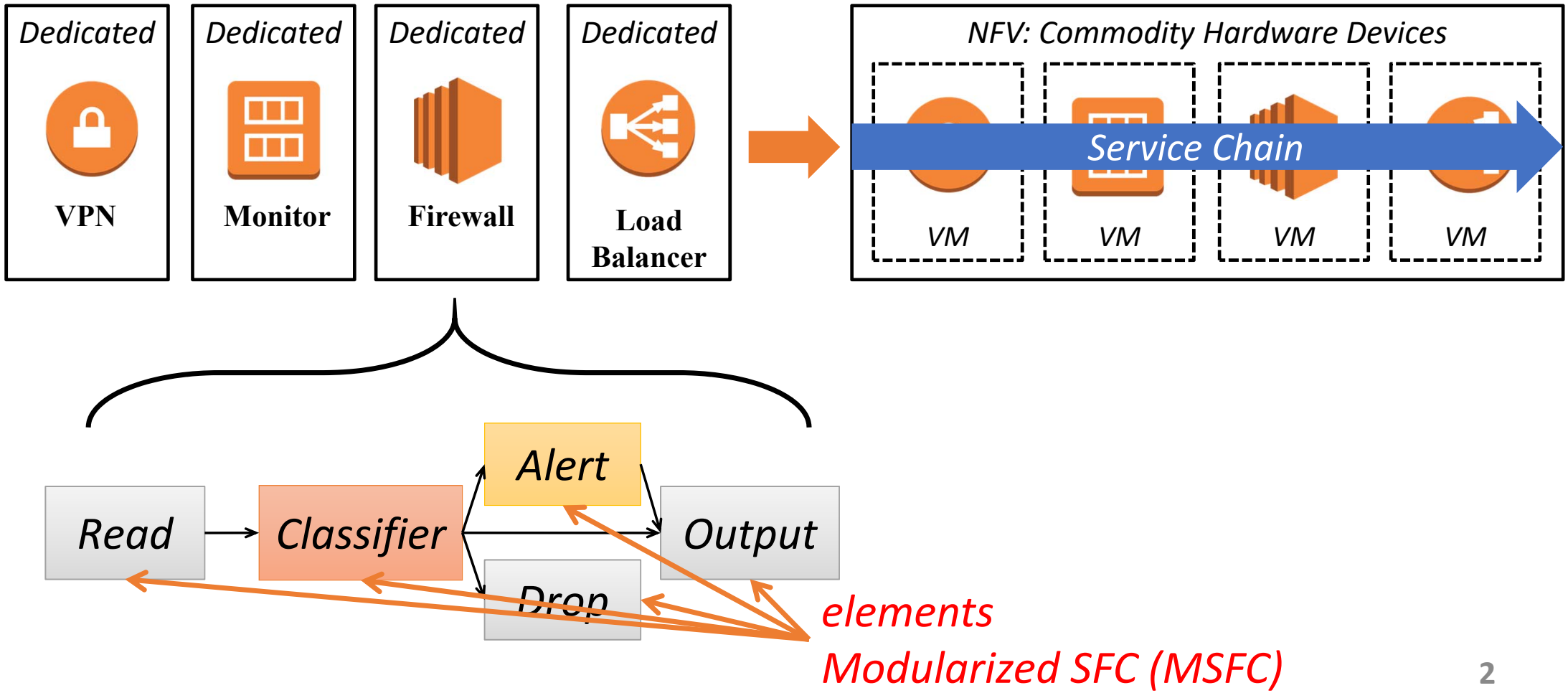
IEEE International Conference on Communications
20-24 May 2018 // Kansas City, MO, USA
Communications for Connecting Humanity

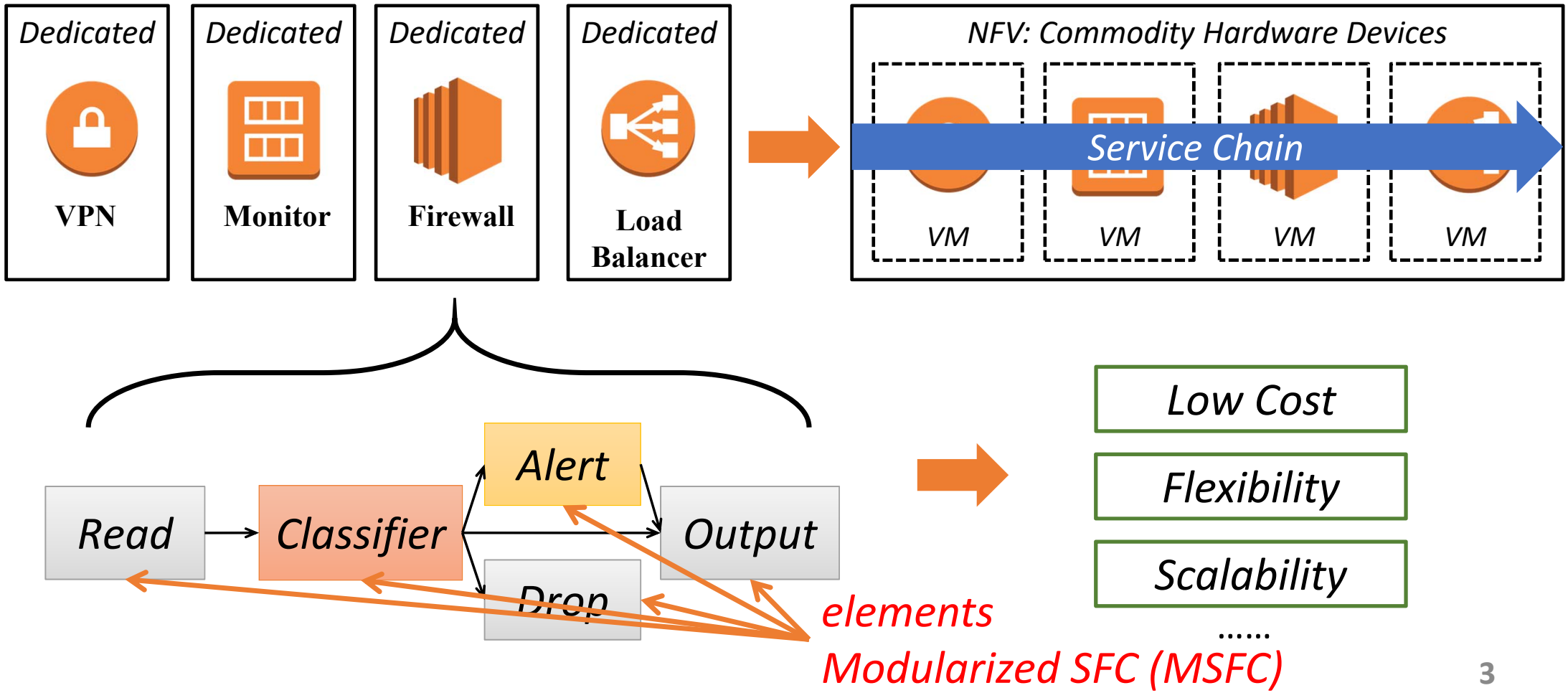
CoCo: Compact and Optimized Consolidation of Modularized Service Function Chains in NFV

Zili Meng Jun Bi Haiping Wang Chen Sun Hongxin Hu

NFV & Modularization

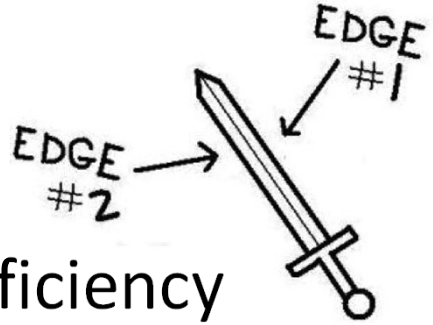


NFV & Modularization



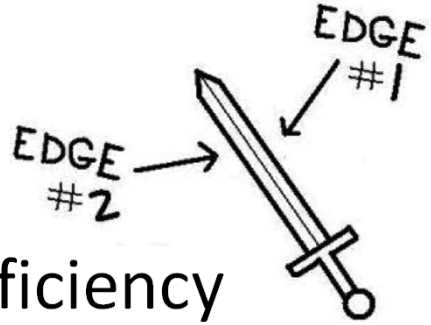
However…

- Two drawbacks:
  - High latency
  - poor resource efficiency



However…

- Two drawbacks:
  - High latency
  - poor resource efficiency

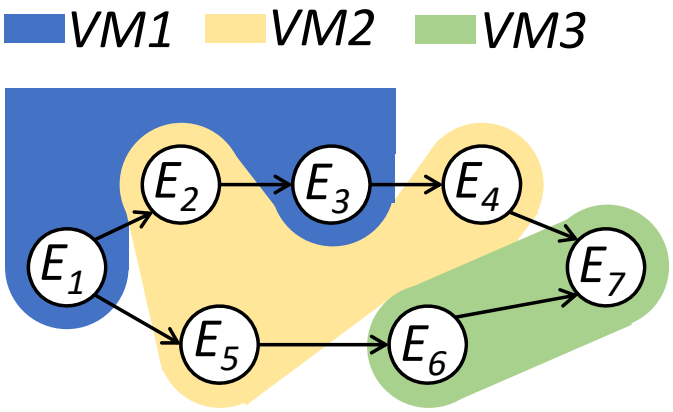


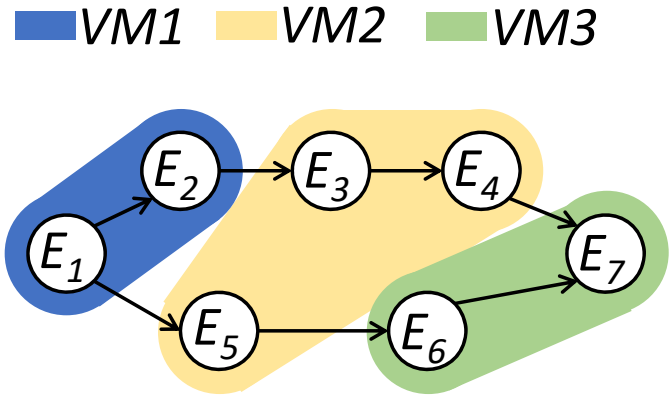
- OpenBox [Sigcomm'16]
  - *Element reuse*
- NFVnice [Sigcomm'17]
  - *NF consolidation: containers in one VM (core).*

Which elements to consolidate?

Key Observations

placement affects MSFC performance by affecting inter-VM transfers





CoCo…

identifies ***inter-VM transfer between elements***

optimizes ***placement of elements on VMs***

optimizes ***dynamic scaling mechanism***

Challenges

- Optimized Placement
  - How to model the inter-VM transfer?
  - How to find optimal solutions efficiently?
- Optimized Dynamic Scaling
  - How to reduce inter-VM transfers during scaling out?

Challenges

- Optimized Placement
  - How to model the inter-VM transfer?
  - How to find optimal solutions efficiently?

Optimized Placer

- Optimized Dynamic Scaling
  - How to reduce inter-VM transfers during scaling out?

Individual Scaler

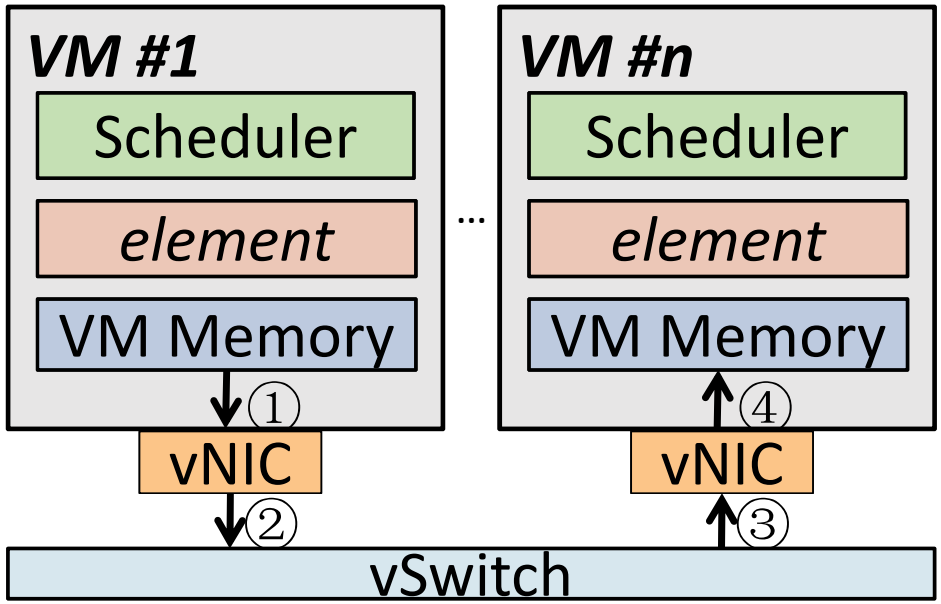
Optimized Placer

- Packet Transfer Cost:
  - Four-step transfer delay: t_d
  - Service chain throughput: Θ
  - Delayed Bytes:

$$DB = \Theta \cdot t_d$$

- Resource Analysis:
  - *Observation:*

The CPU utilization of an element is linear to processing speed



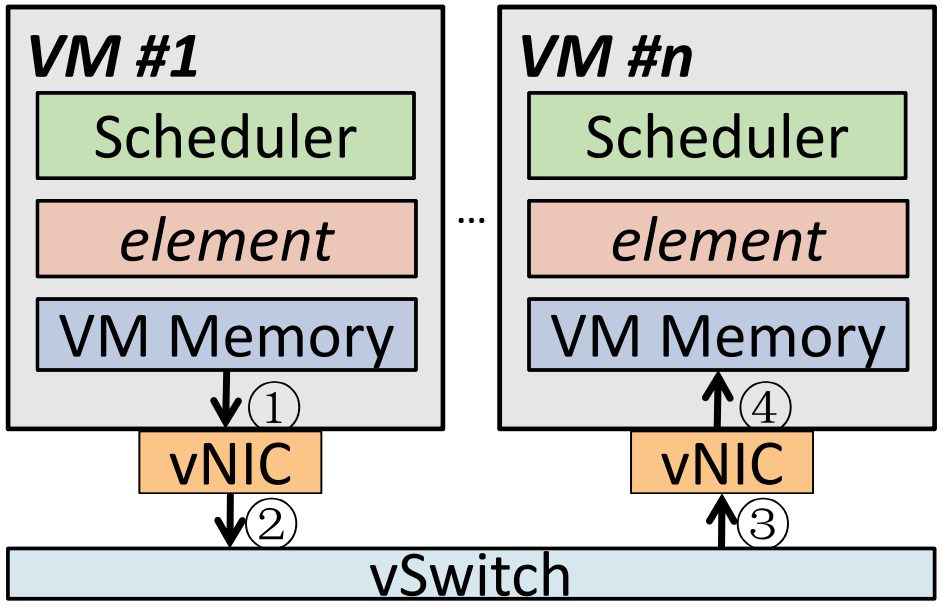
Optimized Placer

- Packet Transfer Cost:
  - Four-step transfer delay: t_d
  - Service chain throughput: Θ
  - Delayed Bytes:

$$DB = \Theta \cdot t_d$$

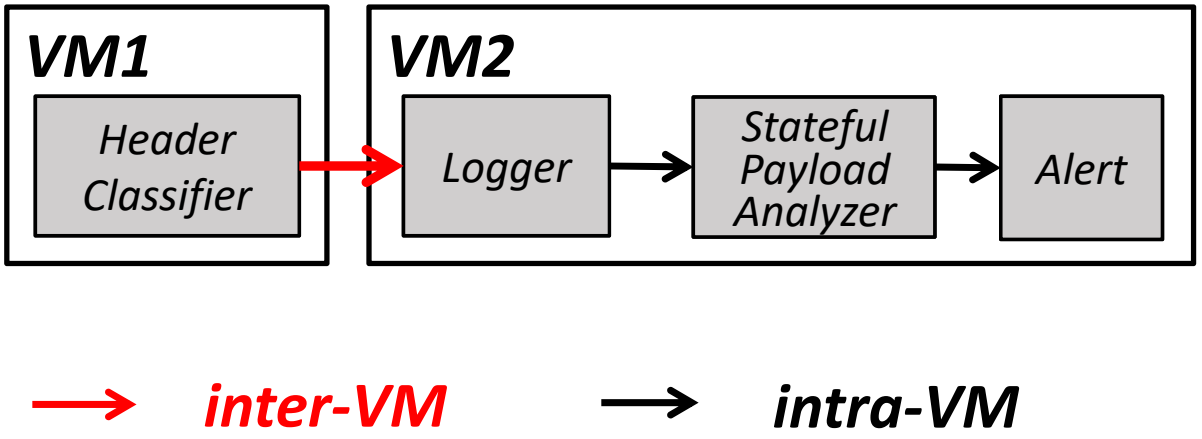
- Resource Analysis:
  - *Observation:*

The CPU utilization of an element is linear to processing speed



Optimized Placer – 0-1 Quadratic Programming

- Intuition: Consolidate adjacent elements together
  - *If we place two adjacent elements together to one VM, there will be no inter-VM packet transfer.*



Optimized Placer – 0-1 Quadratic Programming

- $x_{i,k} \in \{0,1\}$: indicating element i is placed onto instance j
- *Challenge: How to express two elements are placed together?*

k	1	2	3	4	5	6
$x_{i,k}$	0	1	0	0	0	0
$x_{j,k}$	0	0	1	0	0	0
$x_{i,k} \cdot x_{j,k}$	0	0	0	0	0	0

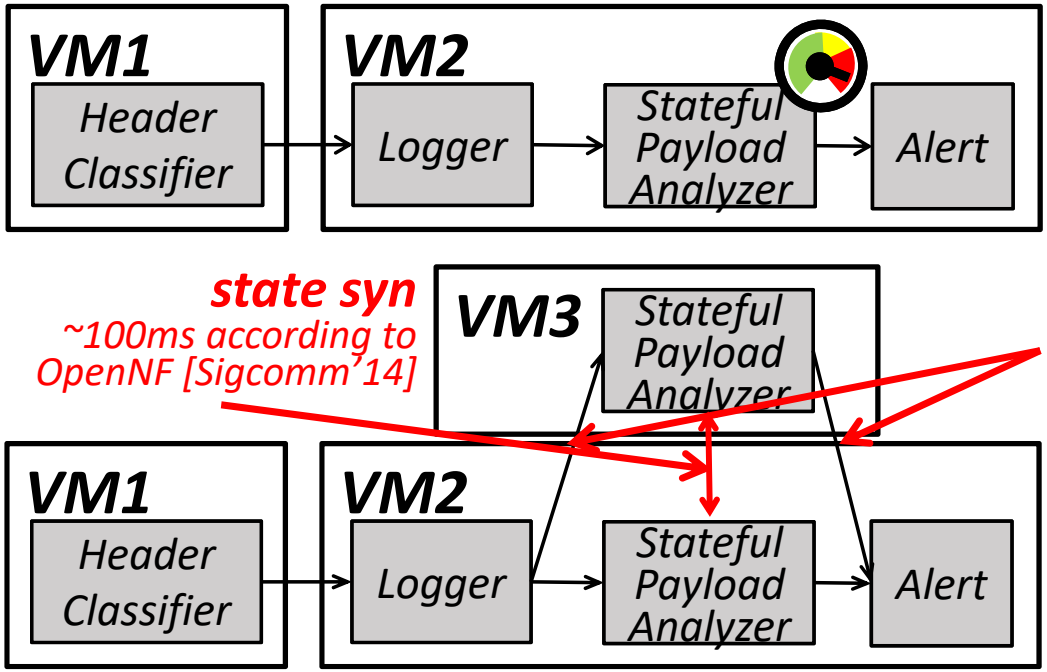
k	1	2	3	4	5	6
$x_{i,k}$	0	0	1	0	0	0
$x_{j,k}$	0	0	1	0	0	0
$x_{i,k} \cdot x_{j,k}$	0	0	1	0	0	0

indicator: $\sum_{k} x_{i,k} \cdot x_{j,k}$ *(quadratic)*

Optimized Placer – 0-1 Quadratic Programming

- Objective
  - The total inter-VM *Delayed Bytes*.
- Constraints
  - The placement cannot lead to the overload of any instances.
- For other mathematical details, please refer to our paper.

Optimized Individual Scaling



MSFC before scaling

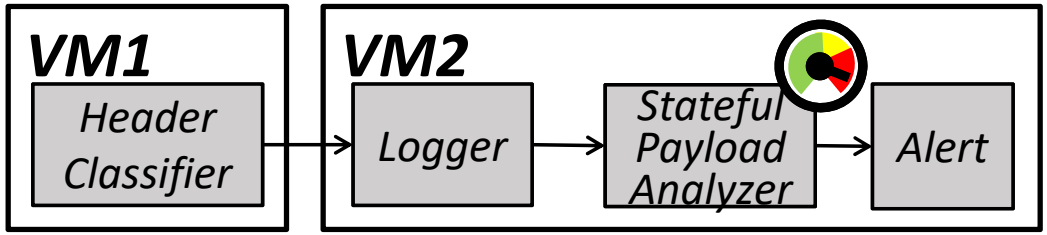
Scaling with traditional method

Optimized Individual Scaling

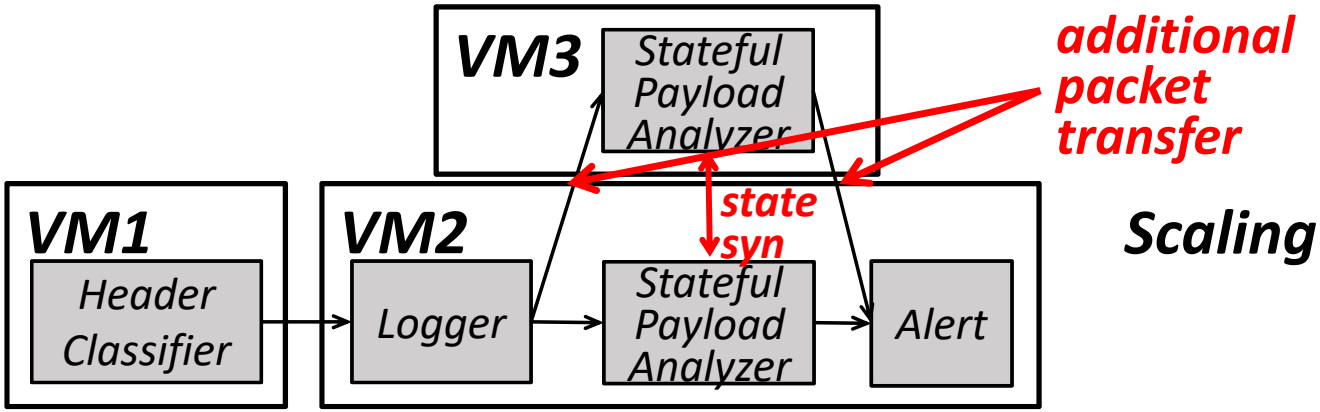
- Key novelty

Migrate other elements consolidated together to release resources for the overloaded element.

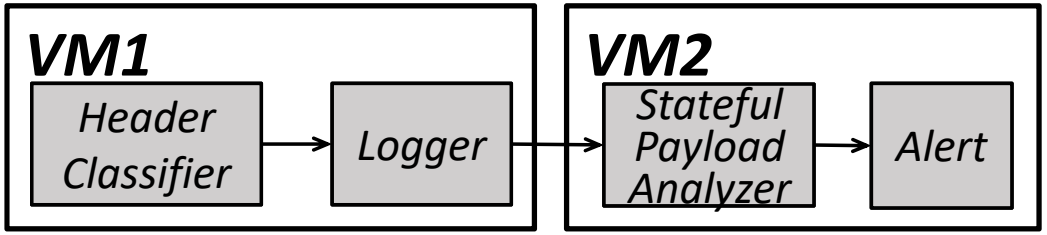
Optimized Individual Scaling



MSFC before scaling



Scaling with traditional method



CoCo

Optimized Individual Scaling

- Consistency guarantee mechanism
  - Overload should be alleviated.
  - Migration will not lead to new hotspots.
- Advantage of CoCo Individual Scaler
  - No new hardware resource consumed
  - Additional packet transfer avoided
  - State synchronization avoided
- Application scenario of CoCo Individual Scaler
  - Imbalance between VMs (OFM [IWQoS'18])

Optimized Individual Scaling

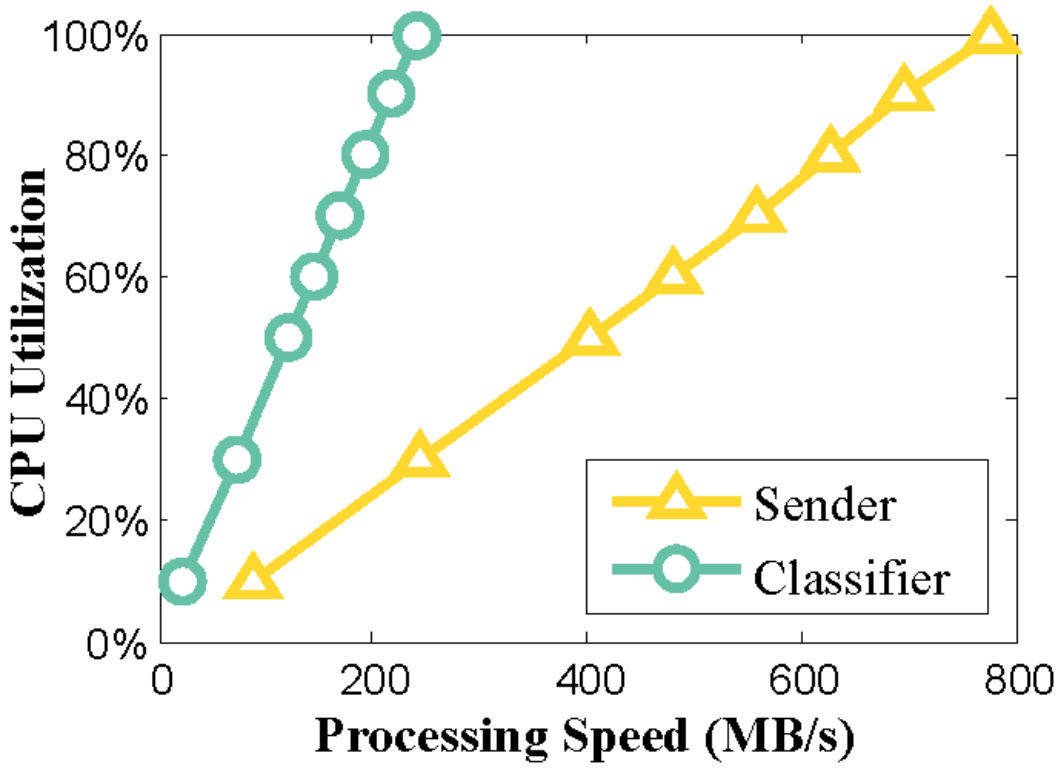
- Consistency guarantee mechanism
  - Overload should be alleviated.
  - Migration will not lead to new hotspots.
- Advantage of CoCo Individual Scaler
  - No new hardware resource consumed
  - Additional packet transfer avoided
  - State synchronization avoided
- Application scenario of CoCo Individual Scaler
  - Imbalance between VMs (OFM [IWQoS'18])

Implementation and Evaluation

- Evaluation Setup
  - Docker for consolidation, DPDK version 16.11
  - OpenNF [Sigcomm'14] and TFM [ICNP'16] for migration mechanisms.
  - MATLAB for solving 0-1 Quadratic Programming
  - Intel(R) Xeon(R) E5-2690 v2 CPUs, 256G RAM, 2 × 10G NICs
- Evaluation Goal
  - demonstrate the assumption of linearity
  - demonstrate the effectiveness of CoCo placement
  - demonstrate the performance of CoCo scaling

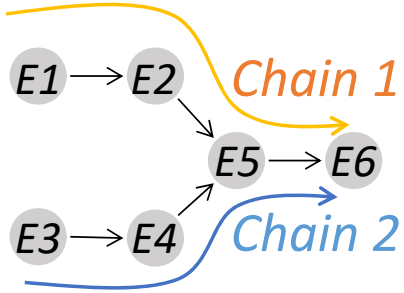
1. Throughput-CPU Utilization

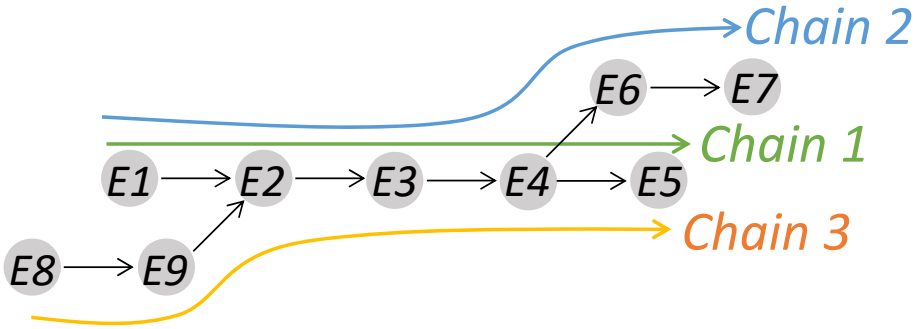
- *For one core only*
- *Sender*
  - $R^2 = 0.9997$
- *Classifier*
  - *100 rules on IP header*
  - $R^2 = 0.9999997$



2. Simulations on Placement

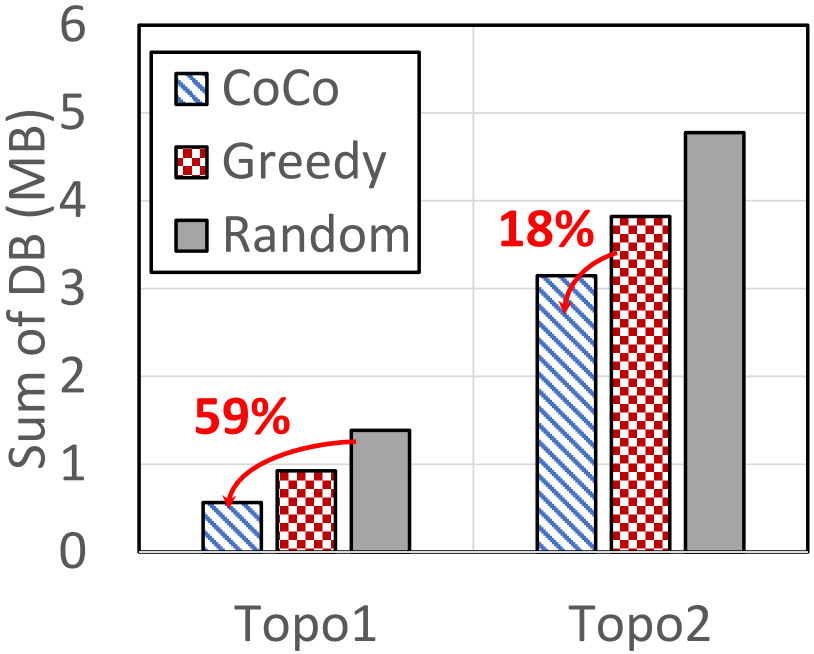
- Evaluation Target
  - Random: select available VMs randomly
  - Greedy: place elements *in sequence* chain-by-chain
- Traffic: Randomly pick flows from CAIDA traffic
- Two topology:



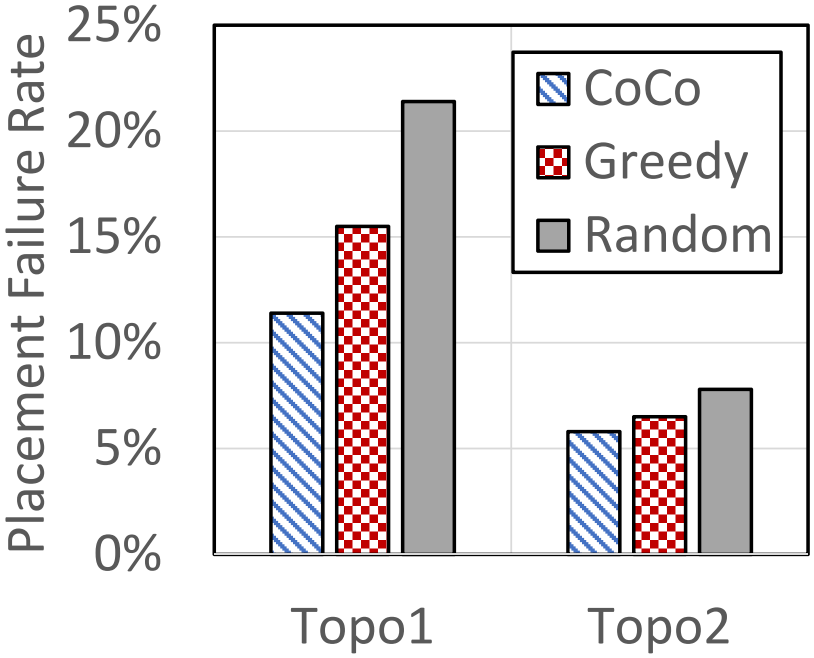


2. Simulations on Placement

- Performance

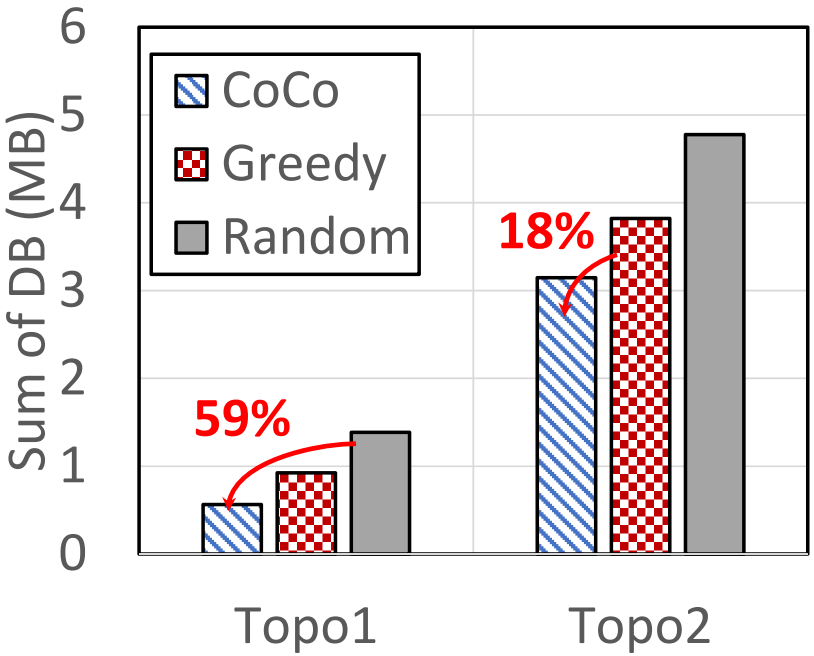


- Resource Utilization

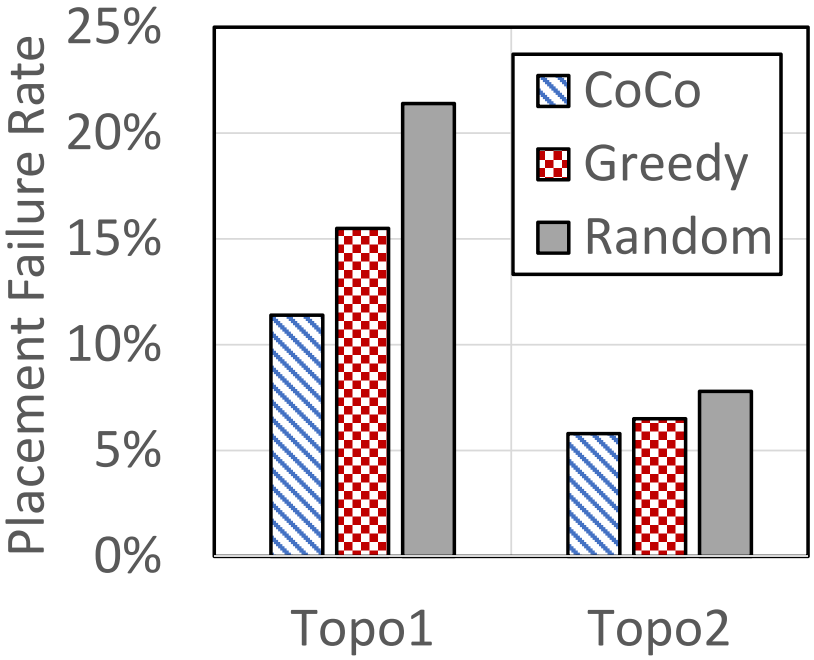


2. Simulations on Placement

- Performance

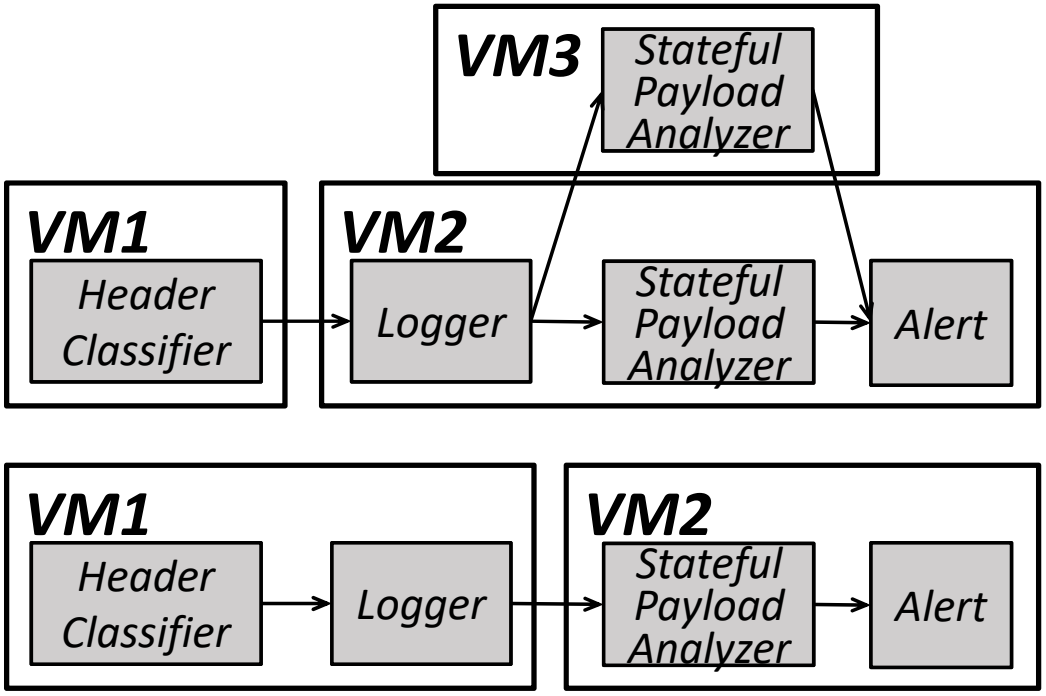


- Resource Utilization

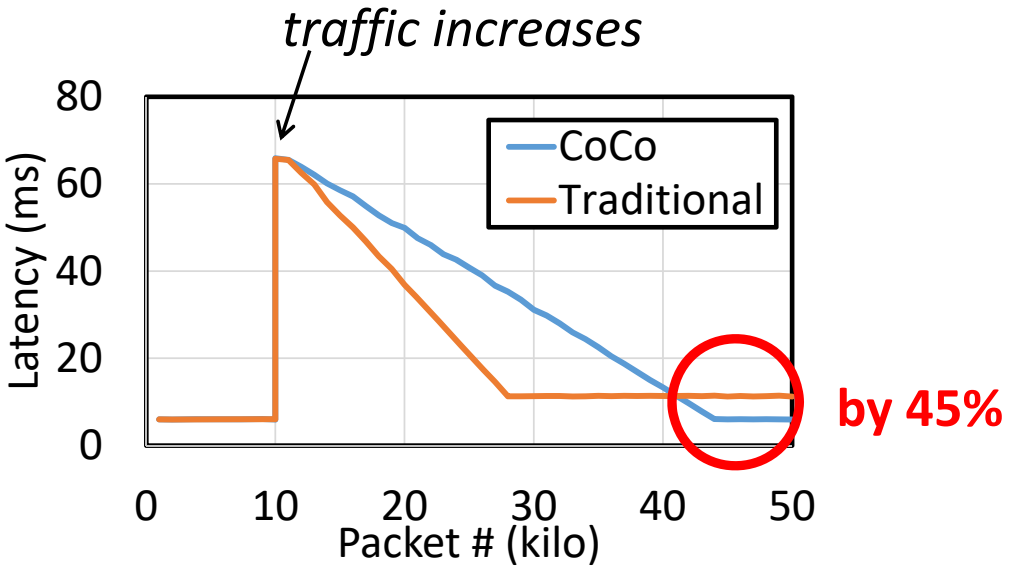


3. Evaluation on Dynamic Scaling

- Based on OpenNF [Sigcomm'14]



- Per-packet latency



Conclusion

- CoCo: Compact and Optimized Consolidation of MSFCs in NFV
  - Optimized Placer
  - Individual Scaler
- Significant Performance Improvement
  - Up to 59% Delayed Bytes reduction in initial placement.
  - 45% latency reduction when dynamic scaling.
- Future work
  - Multi-core placement
  - Intra-core cache analysis

Thank you!

netarchlab.tsinghua.edu.cn

mengzl15@mails.tsinghua.edu.cn