ResQ: Enabling SLOs in Network Function Virtualization

Amin Tootoonchian*
Aurojit Panda‡
Chang Lan†
Melvin Walls§
Katerina Argyraki●
Sylvia Ratnasamy†
Scott Shenker†‡

*Intel Labs  †UC Berkeley  ‡ICSI  ▲NYU  §Nefeli  ●EPFL
NFV Builds on Resource Sharing

Classic approach
Dedicated hardware
Individual functions

NFV approach
Shared hardware
Functions in software
• Performance depends on neighbors’ activity.

• Due to sharing of network, server, and processor resources.
Assumptions on Resource Sharing and Isolation

Traffic isolation through fabric and NIC QoS mechanisms.

But share on-die *uncore* resources.

Independent NFs do not share the same core.
Does Resource Contention Matter?

Solo run

Traffic Generator

port_1 → core_1
port_2 → core_2
port_3 → core_3
... → core_n

Target NF’s throughput $T_{solo}$
Target NF’s latency $L_{solo}$

Consolidated runs

$T_1$ $L_1$
$T_2$ $L_2$
... $T_m$ $L_m$

How far off is $\min(T_i)$ and $\max(L_i)$ from $T_{solo}$ and $L_{solo}$?
Does Resource Contention Matter?

Significant degradation for most NFs.
Approaches to Offer Performance SLOs

Prediction (indirect)
• Contention-aware placement.
• Accurate prediction is hard.
  • Optimistic → SLO violation.
  • Conservative → inefficient.
• Algorithmically complex.
• No isolation with SLO violations.
  • May lead to neighbor violations.

Isolation (direct)
• Neighbor-indep. placement.
• No need for prediction.
• Algorithmically simpler.
• Isolation despite SLO violations.
  • Never affects neighbors’ SLOs.

Enabler: emergence of hardware resource isolation mechanisms.
ResQ: SLO Enforcement by Direct Isolation

1. Direct performance isolation

2. Performance SLO enforcement
Direct Performance Isolation
Intel Cache Allocation Technology (CAT) for LLC isolation:

• Classify cores/threads/VMs.
• Assign parts of LLC to classes.

Is LLC isolation sufficient to ensure NF performance isolation?
LLC Isolation Is \textit{Not} Sufficient!

- Achieves a high level of isolation with small packets.
- But \textit{up to 15\% degradation} with large packets.
  - Despite small-packet traffic being more resource intensive.
- Observed high memory utilization with large-packet traffic.
  - But, in general, we expect NFs to generate low memory traffic.
  - Also, NF LLC miss rates with large & small packets are comparable.
- Root cause: high I/O-related mem. traffic due to LLC misses.
The Leaky DMA Problem

- NICs do DMA transfers to part of LLC.
  - Enabled by Intel Data Direct I/O Technology (DDIO).
  - By default, uses 10% of LLC to allocate buffers.

- Contention for DDIO LLC space.
  - Large packets require 12x more space than small packets.
  - CAT does not apply to I/O.

Solution: limit # on-the-fly packets, e.g., buffer sizing.
Accuracy of ResQ’s Isolation Mechanism

LLC isolation and buffer sizing ensures performance isolation with a high degree of accuracy (<3% error).
Performance SLO Enforcement
ResQ SLOs

• Reserved SLOs: static allocation.
  • Input: NF, expected config and traffic profile.
  • Target: throughput, latency.

• On-demand SLOs: dynamic allocation.
  • Input: NF.
  • Target: latency.
ResQ Admission Process

- Profile NFs.
  - Construct a performance model.
  - Fast and scalable.

- Fast greedy allocation.
  - Deny admission if infeasible.
  - Compute # of instances.
  - Compute core & LLC allocation per instance.
ResQ Optimal Scheduler

• MILP formulation for the optimal solution.
  • Slow compared to greedy allocation.

• Run in the background (i.e., not in the admission path).
  • Rearrange NFs if necessary.

• Practical for small clusters.
  • Takes seconds to minutes.
  • Larger clusters: divide into smaller ones with independent solvers.
Resource Efficiency

Highly inefficient (conservative predictor)

Only up to 18.5% worse than optimal

Cost of hard partitioning is <3% compared to greedy.

- ResQ Optimal: 22 servers
- ResQ Greedy: 22 servers
- Dynamic (no isolation): 22 servers
- Prediction [1] (no isolation): 22 servers

ResQ achieves better accuracy & efficiency than prior work.
   • Despite using simple heuristics and algorithms.

Enabled by direct performance isolation.
   • Plenty of room for improvement with software mechanisms.

Code available at [https://github.com/netsys/resq](https://github.com/netsys/resq)
   • Useful for general NFV experimentation.