

On the Benefits of Predictable Traffic in Virtual Switches: A Case Study

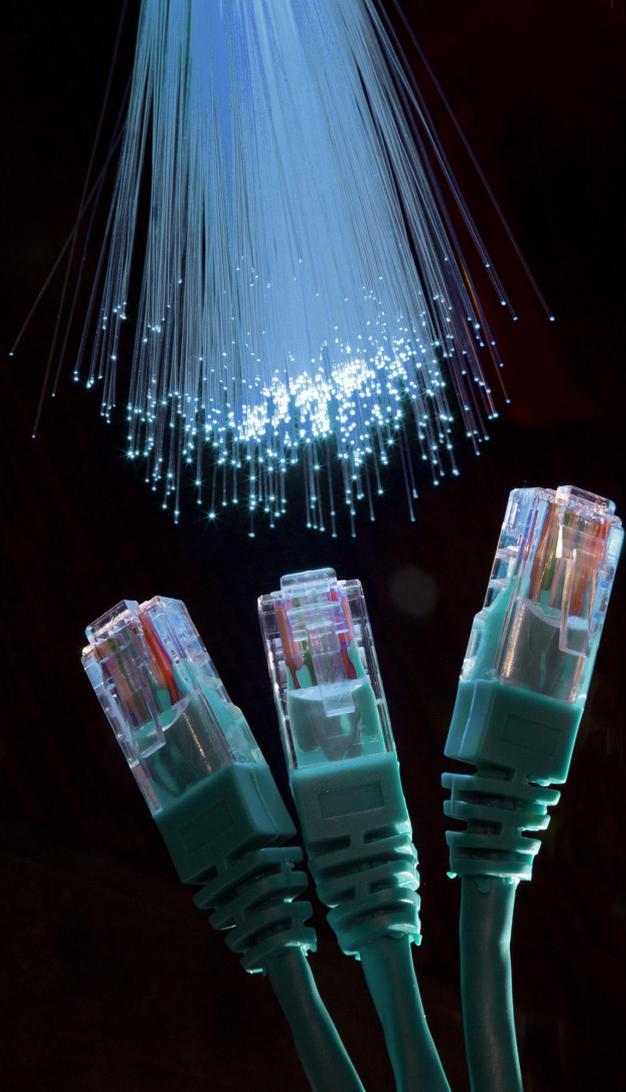
Emil Ståhl - KTH Royal Institute of Technology

Marco Chiesa- KTH Royal Institute of Technology

Eelco Chaudron - Red Hat

Simone Ferlin-Reiter - Red Hat and Karlstad University

ANRW 2025 – July 22, Madrid



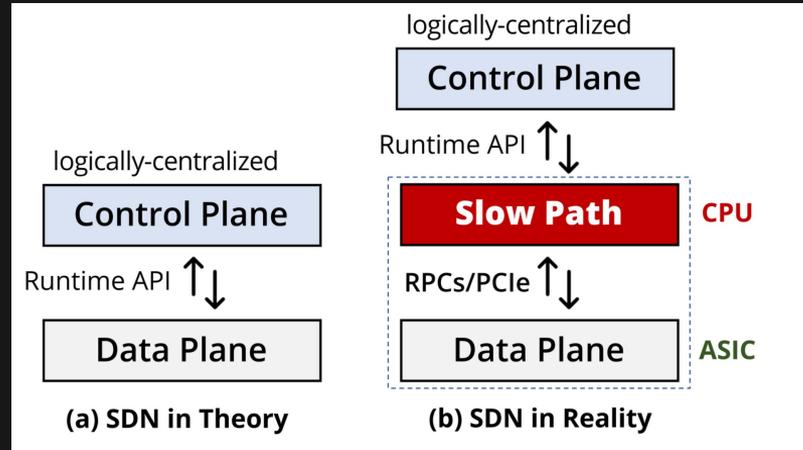
Motivation

● Software-Defined Networking

- SDN introduces a slow path bottleneck in OVS
- Reactive flow rule installation adds significant latency
- Bursty and short-lived flows in containers exacerbate the issue

● Recent Studies

- Emerging research identifies the slow path as a potential bottleneck, highlighting the need for a dedicated accelerator similar to data plane advancements.



Related work

- Slow path bottlenecks in OVS widely recognized

E.g., Zulfiqar et al. [SIGCOMM '23]: slow path needs acceleration

- **NuevoMatch [NSDI '22]:**

Neural-network-based classification

Adds inference cache layer or replaces data path entirely

- **Seer [NSDI '24]:**

Predictive caching for future packet arrivals

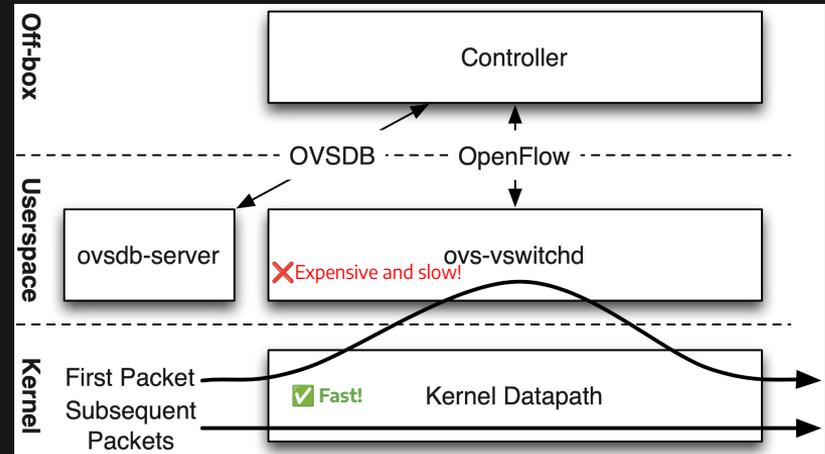
Reduces cache misses up to 65%

- **Existing approaches analyze past or current traffic**

Our work: Preemptively installs rules based on future flow prediction

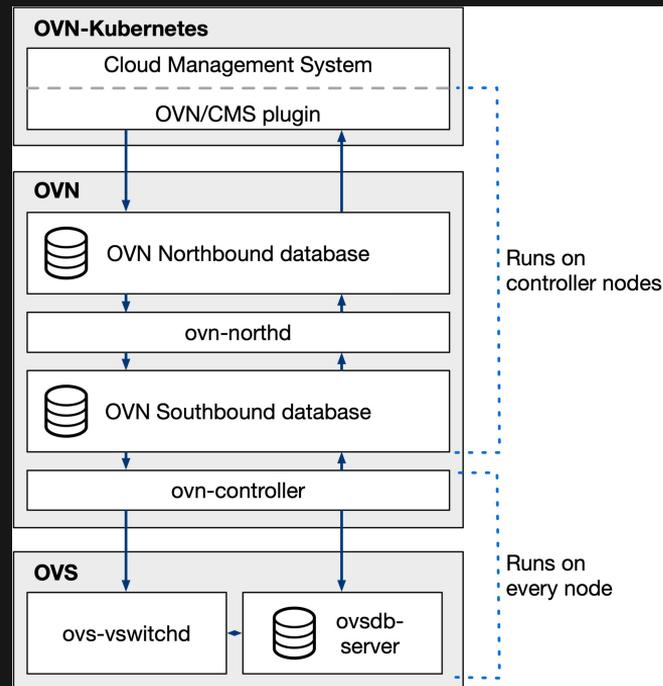
Problem Statement

- Slow path = upcalls from kernel to user space
- Flow cache misses cause latency and high CPU usage
- Question: Can predicting flows help OVS performance?



OVS Architecture in the K8s networking stack

- **Fast path (kernel): microflow + megaflow cache**
- **Slow path (user): ovs-vswitchd handles upcalls**
- **Costly transitions & CPU load on slow path**



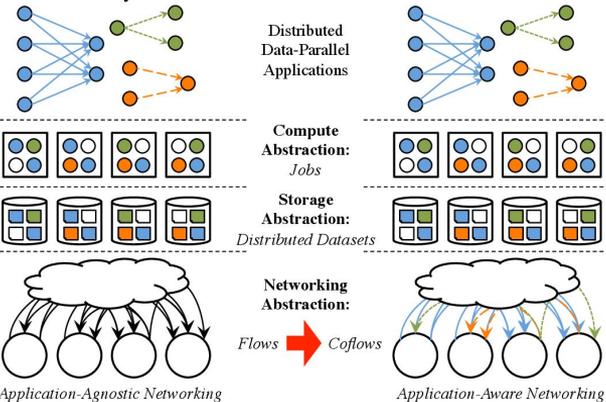
Key Idea

- Predictable traffic allows preloading flow rules
- Use coflows to identify predictable flow groups [1]
- Evaluate performance gains via synthetic coflow workloads

A semantically related collection of flows between two groups of machines is referred to as a coflow. Mathematically, a coflow c is defined as:

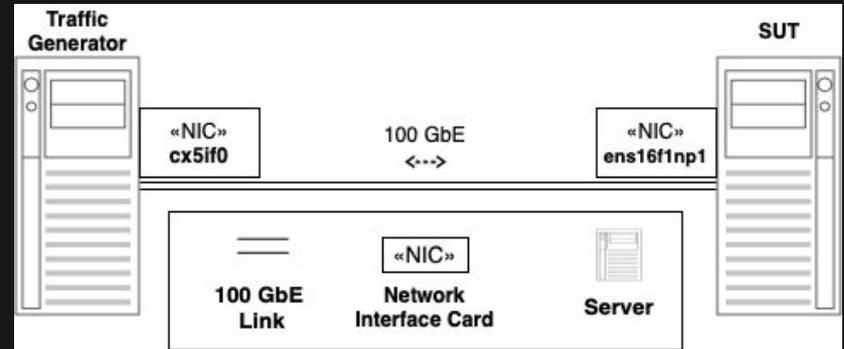
$$c(S, D) = \{f_1, f_2, \dots, f_{|c|}\} \text{ where:}$$

- c represents the coflow,
- S is the source machine group,
- D is the destination machine group,
- $|c|$ represents the cardinality, denoting the number of flows within the set represented by c ,
- f_i represents individual flows between machines in groups S and D , and
- $size(f_i)$ represents the size of individual flow f_i , typically measured in terms of bytes transmitted.



Experimental Setup

- Two-node testbed with 100GbE NICs
- SUT: OVS 3.3, OVN, OpenShift-like topology
- Traffic generator: FastClick + DPDK



Measure Latency

- Custom built XDP program inserts 64-bit timestamps into UDP packets
- FastClick inserts packet numbers
- Daemon running on each pod parses timestamps and packet numbers

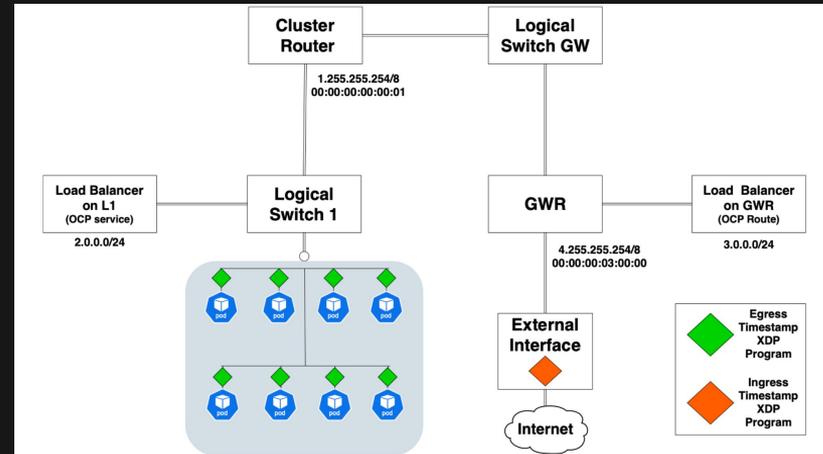


Figure 4: OVN network cluster of a single-node OCP: Ingress timestamp (orange) and egress timestamp (green).

Coflow Workloads

- **Based on real-world traces**
(MapReduce & Google Search RPC)
- **Two Flow Size Distributions (FSDs)**
Long-lived (FB Hadoop) vs short, bursty (GSRPC)
- **193,000 5-tuple UDP flows**
1M packets/sec; 12 Gbps

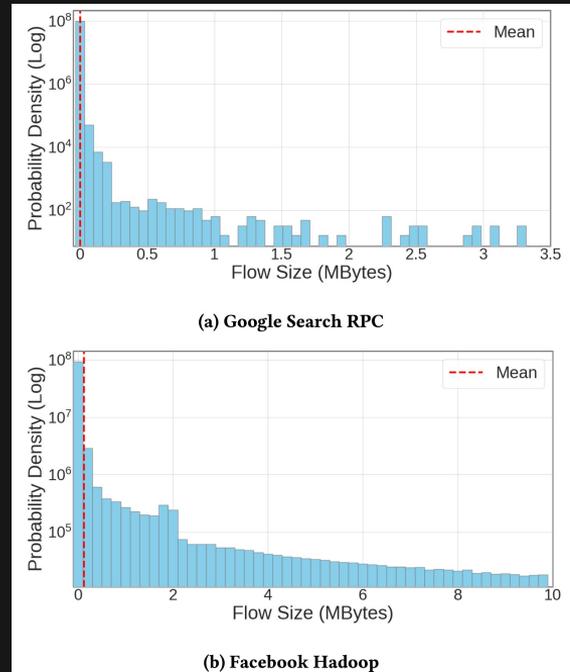


Figure 5: Histogram of the flow size distribution. Note the difference in the x-axis.

Coflowiness Parameter

- Quantifies predictability of associated flows
- Range: 0 (random flows) → 1 (fully predictable)
- Enables evaluation of performance vs prediction accuracy

Table 1: Coflowiness levels and associated flows.

Coflowiness Level	(%) Associate Flows Preloaded
0.0	0% (Baseline, no preloading)
0.1	10%
0.25	25%
0.5	50%
0.75	75%
0.9	90%
1.0	100% (Optimal, complete preloading)

Benchmarking Scenarios

- **Baseline: No prediction**
- **Optimal: All associated flows preloaded**
- **Varying coflowiness: 10%–90% flow prediction levels**

Table 1: Coflowiness levels and associated flows.

Coflowiness Level	(%) Associate Flows Preloaded
0.0	0% (Baseline, no preloading)
0.1	10%
0.25	25%
0.5	50%
0.75	75%
0.9	90%
1.0	100% (Optimal, complete preloading)

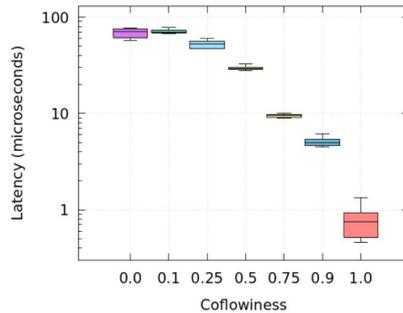
Mean packet latency

Facebook Hadoop

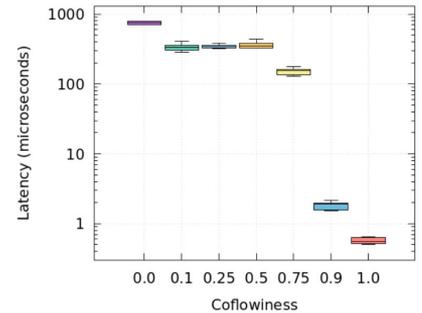
Predicting 25% of flows \rightarrow ~24% latency reduction. Consistent gains with increased coflowiness. Larger flows benefit more from caching

GSRPC

Gains only visible after 75% flow prediction. High flow rate overwhelms cache effectiveness. Baseline latency up to 10 \times higher than FB Hadoop



(a) Facebook Hadoop



(b) Google Search RPC

Figure 4: Mean ingress-to-egress packet latency calculated from the first packet of each unique flow processed by OVS, using two different flow size distributions (note the difference in the y-axis).

CPU Utilization

- Facebook Hadoop

25% prediction → ~12% CPU reduction

- GSRPC

Only full prediction shows significant CPU gain

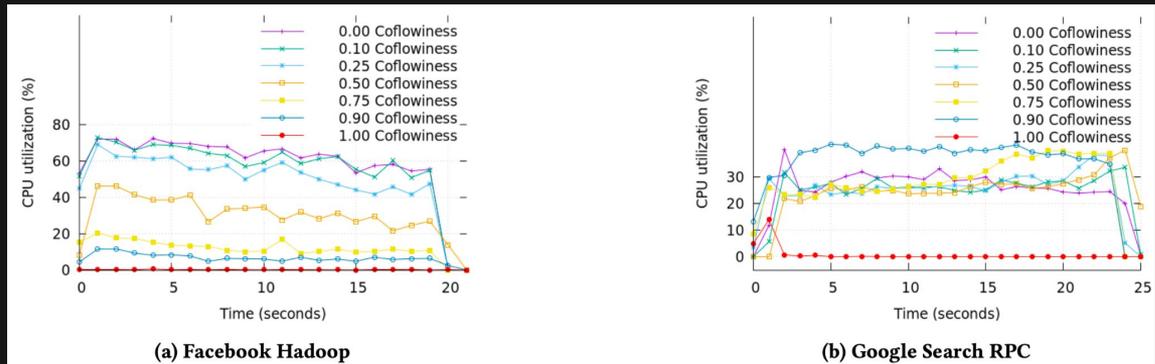


Figure 5: Total CPU utilization for OVS handler threads under varying flow size distributions.

Bottlenecks & Trade-offs

- Prediction benefits larger, long-lived flows
- High flow rate limits caching benefits
- Batch upcalls help CPU but increase latency

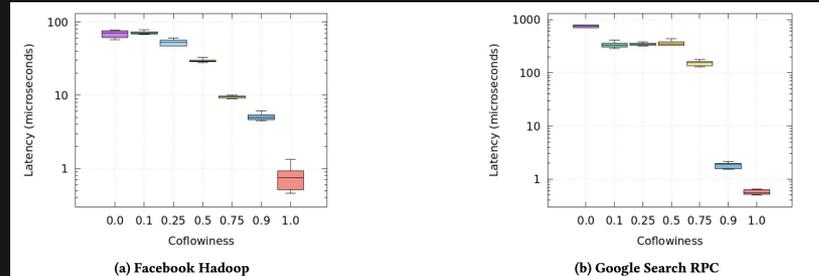


Figure 4: Mean ingress-to-egress packet latency calculated from the first packet of each unique flow processed by OVS, using two different flow size distributions (note the difference in the y-axis).

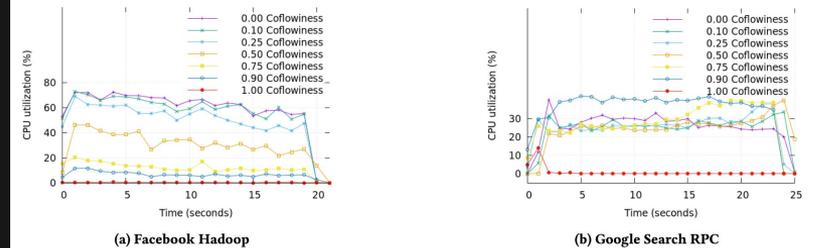


Figure 5: Total CPU utilization for OVS handler threads under varying flow size distributions.

Future Work

- Develop realistic prediction models (ML/statistical)
- Hardware support for predictive caching
- Evaluation in multi-node OCP clusters with real traffic

Conclusions

- **Predicting just 25% of flows improves latency and CPU usage**
- **Benefits depend on flow size, rate, and burstiness**
- **Simple prediction + preloading = effective OVS optimization**

Thanks for listening

Questions?