Software Data Planes: You Can't Always Spin to Win

Hossein Golestani, Amirhossein Mirhosseini, Thomas F. Wenisch
University of Michigan

ACM Symposium on Cloud Computing (SoCC)
November 22, 2019

adacenter.org @ADA_Center

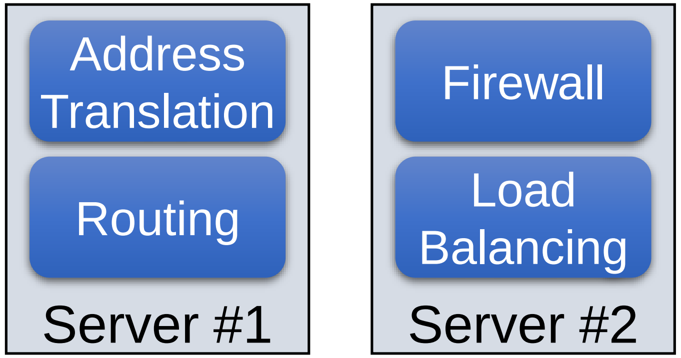
This work is supported by the Semiconductor Research Corporation (SRC) and DARPA

JUMP

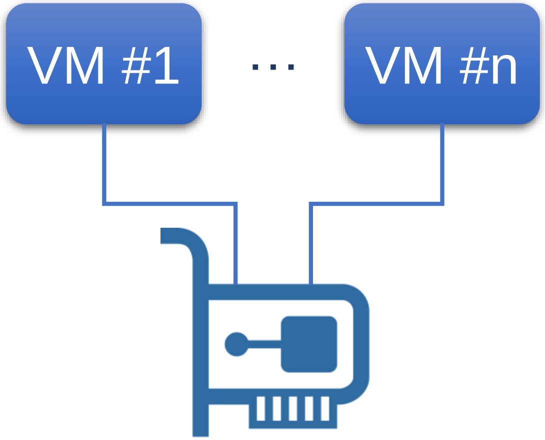
Applications Driving Architectures

What's Up in the Cloud?

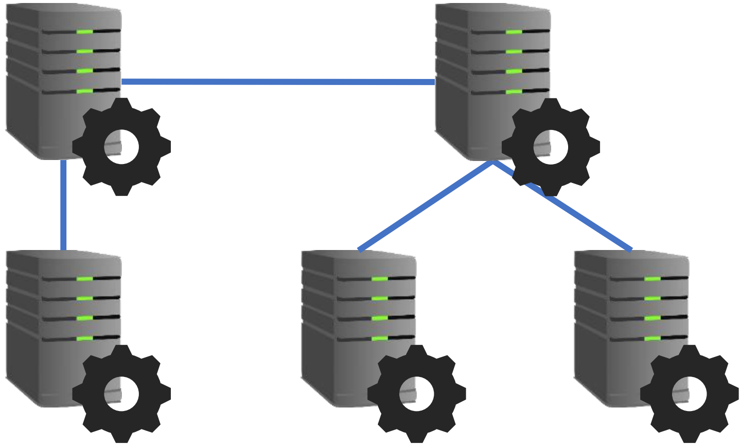
- Virtual µs-scale computing era



Network function virtualization

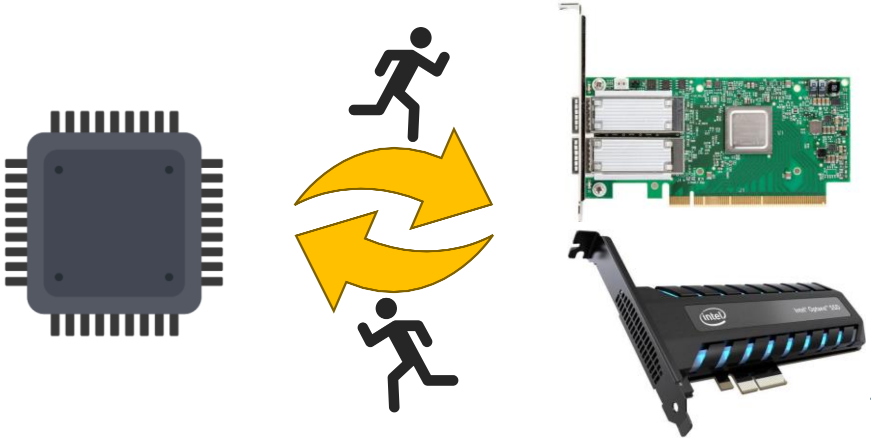


I/O virtualization



Microservices

- Service objectives
  - High throughput
  - Low average/tail latency

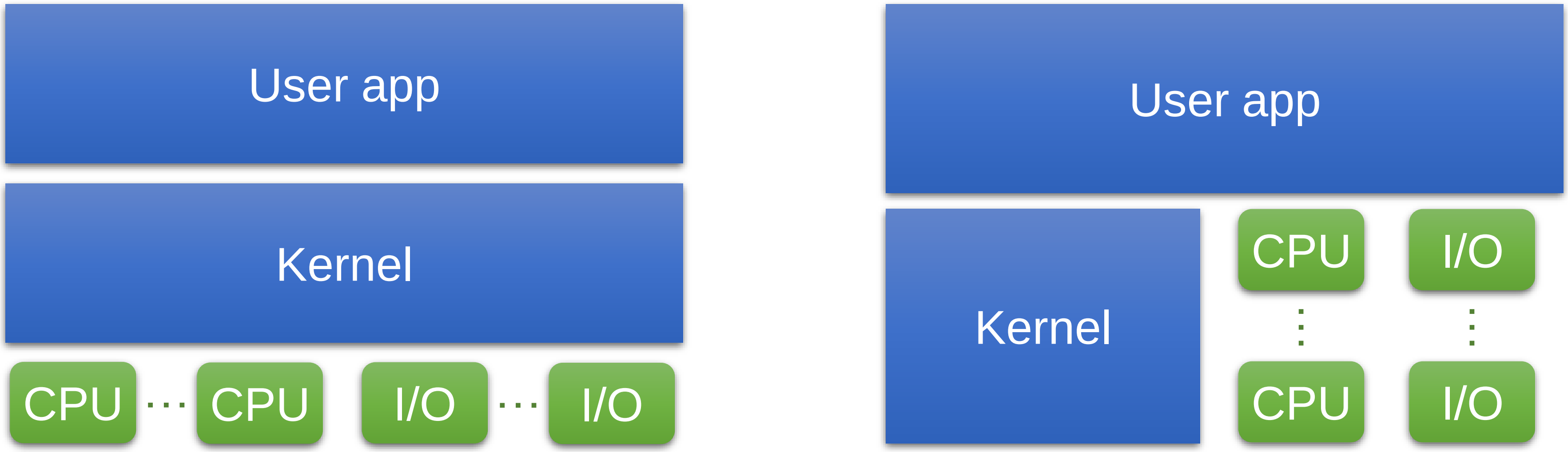


*High-speed I/O**

* Image credits: Mellanox, Intel

Software Stacks: Under Revision

- Then vs. now

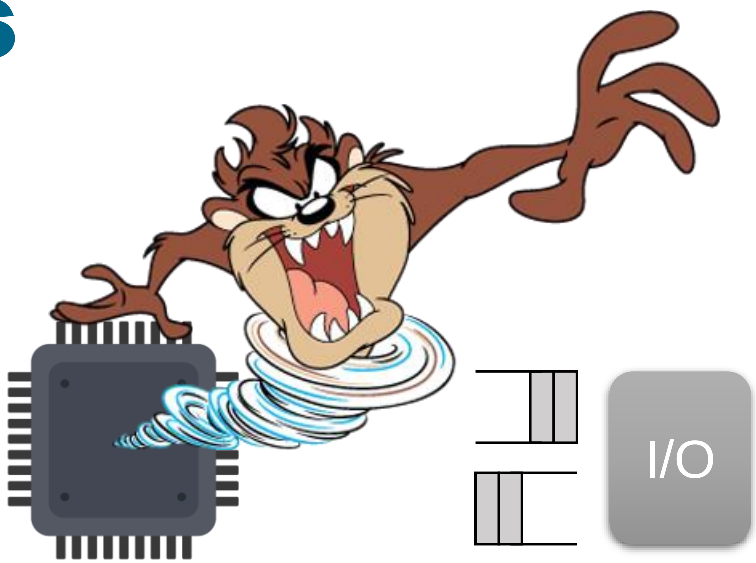


- Kernel-bypass architectures (just a handful)

Andromeda [NSDI'18]
Arrakis [OSDI'14]
IX [OSDI'14]
mTCP [NSDI'14]
ReFlex [ASPLOS'17]
Shenango [NSDI'19]
Shinjuku [NSDI'19]
Snap [SOSP'19]
ZygOS [SOSP'17]

Software Data Planes

- Key mechanisms
  - User-level shared queues
  - Spin-polling cores



- Fast notification by cache coherence write signals

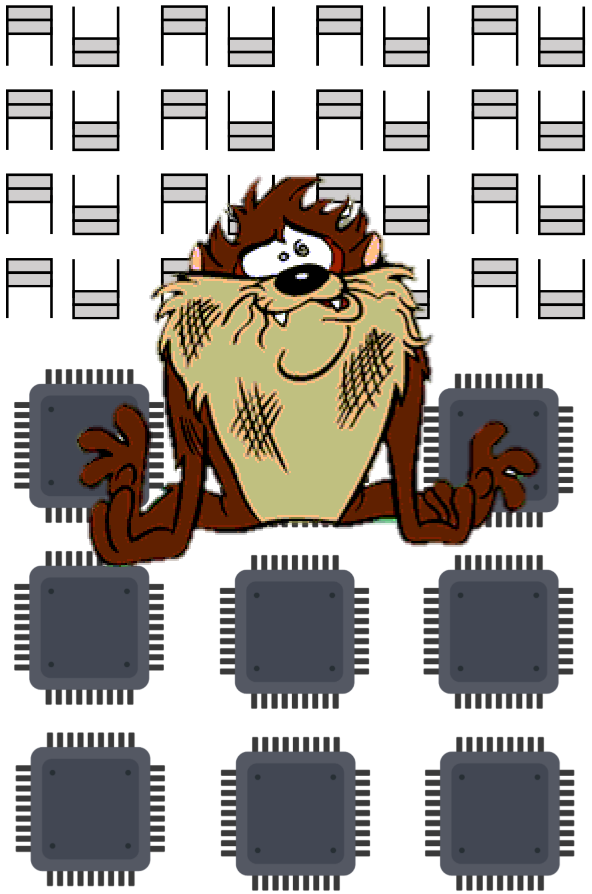
- Widely adopted in industry

DPDK
DATA PLANE DEVELOPMENT KIT

SPDK
STORAGE PERFORMANCE DEVELOPMENT KIT

Spin-polling: Not a Panacea

- An easy-to-use and fast model for communication and signaling
- But far from ideal, especially when scaled
- We show that spin-based data planes:
  - Perform more work when there is less
  - Are not scalable to many cores
  - Are not scalable to many queues
  - Are not well-suited for shared queues



Outline

- Introduction to Software Data Planes
- Methodology
- Characterization of Software Data Plane Challenges
- Solution Directions
- Conclusion

Methodology

- Setup
  - DPDK-based applications
  - Skylake cores
  - 100GbE Mellanox NIC

- Experiments

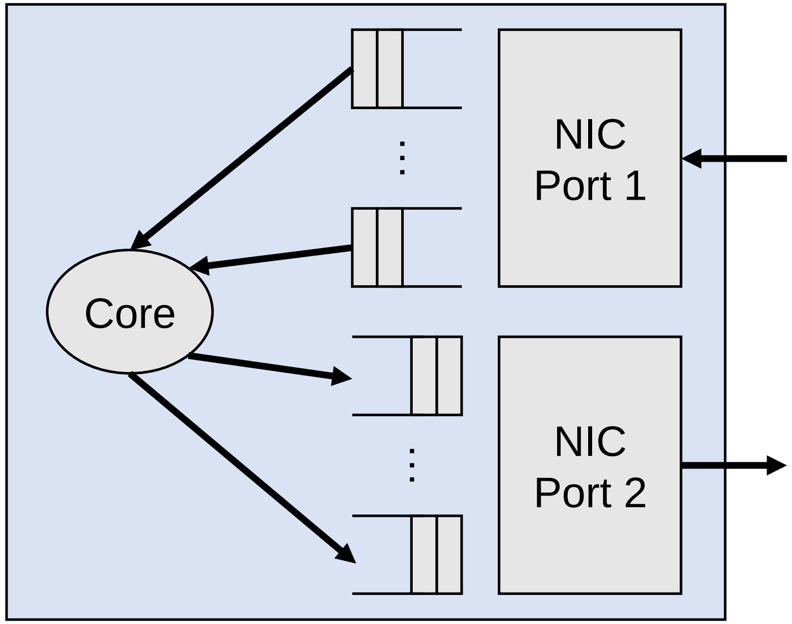
1. Inefficiencies of spin-polling
2. Lack of queue scalability
3. Impracticality of queue sharing

Inefficiencies of Spin-polling

- Polling "tax"
  - Body of poll loop
  - Useless polling on idle queues (possibly causing cache misses)
- Affects throughput scalability with cores

```
While forever:
    For each RX queue:
        Read packets from RX queue;
        If there are any packets:
            Route packets using LPM*;
            Send packets to TX queue(s);
```

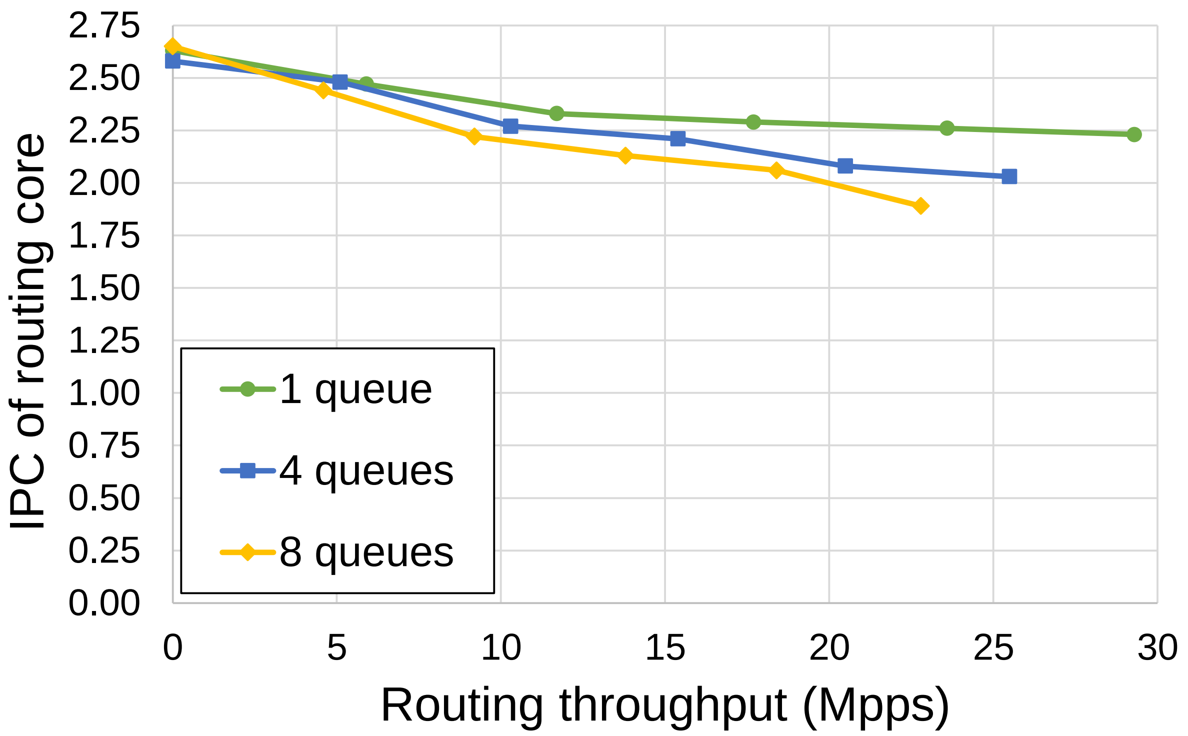
* LPM: Longest Prefix Match



Polling tax can be 20-28% of total CPU cycles even in 100% load

IPC != Useful Work

- IPC (Instructions Per Cycle) of routing core at varying loads



IPC decreases as load increases, resulting in **energy inefficiency**, **fast aging**, and **severe co-runner interference**

1

2

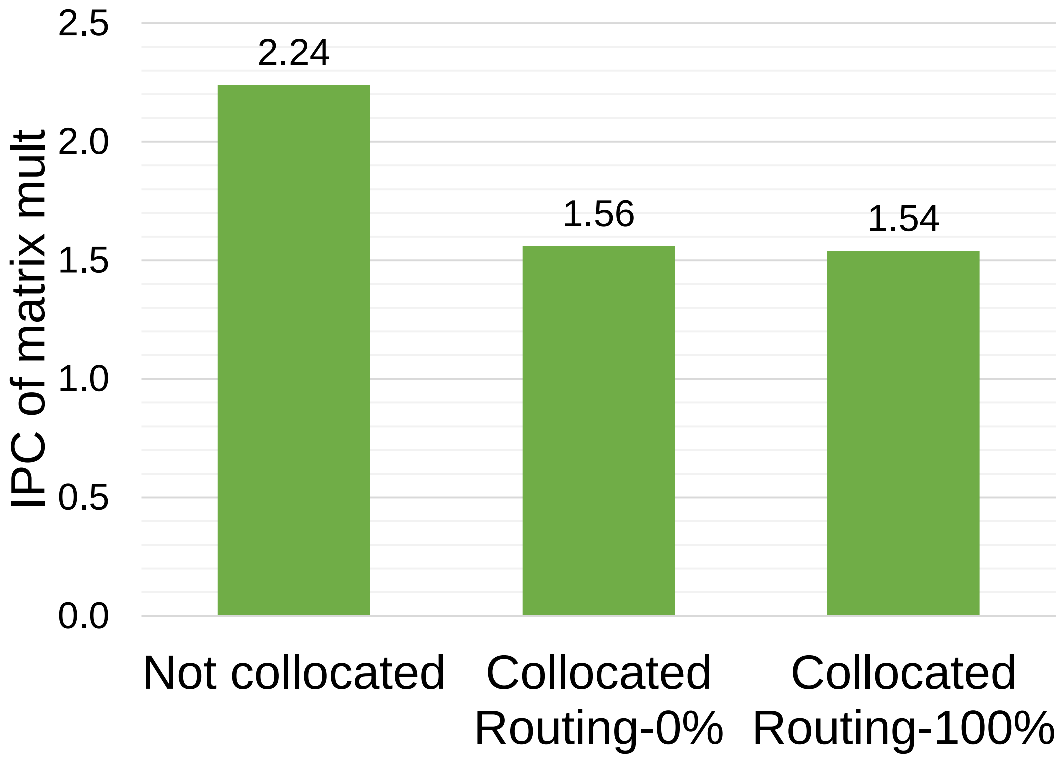
3

Effect on SMT Co-runner

- More (useless) instructions executed in lighter traffic

- Co-running:
  - Matrix mult
  - Spin-based routing (0-100% load)

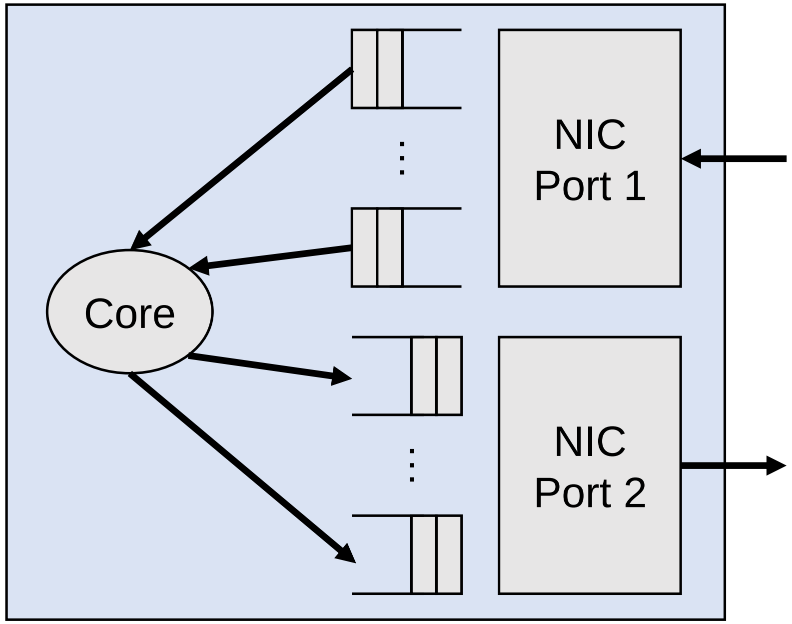
- Executed on:
  - SMT cores of a physical CPU
  - Different physical CPUs



Useless spinning wastes execution resources of an SMT co-runner

Lack of Queue Scalability

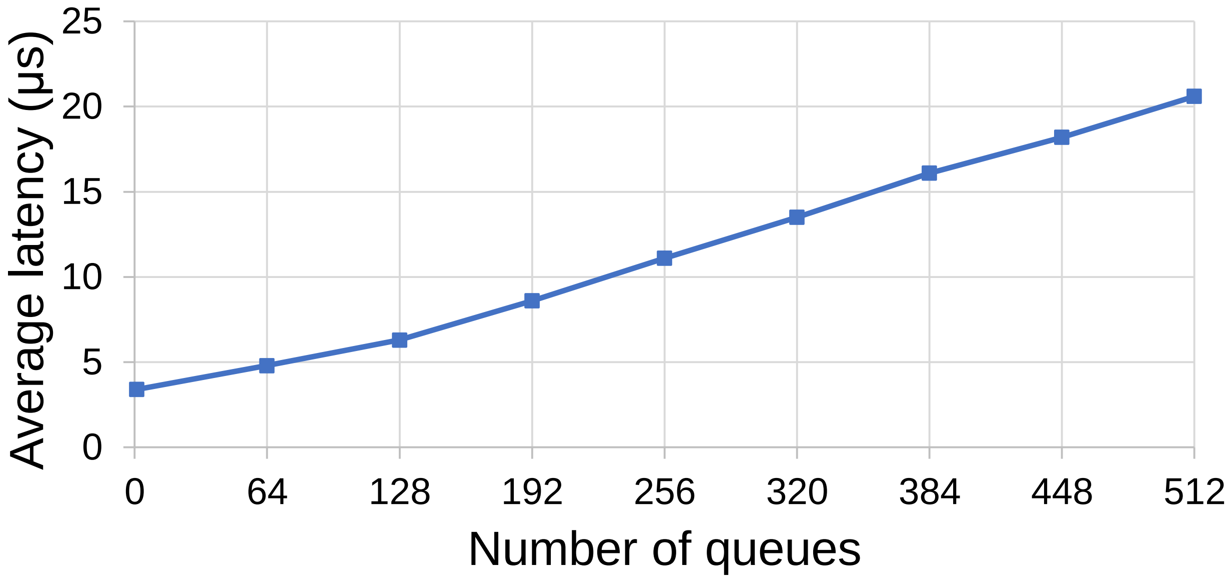
- Traffic flows spread among multiple queues
- Limited size of CPU caches: a performance antagonist
- Experiment
  - Forwarding packets by a single core
  - Scaling up the number of queues



1
2
3

Effect on Latency

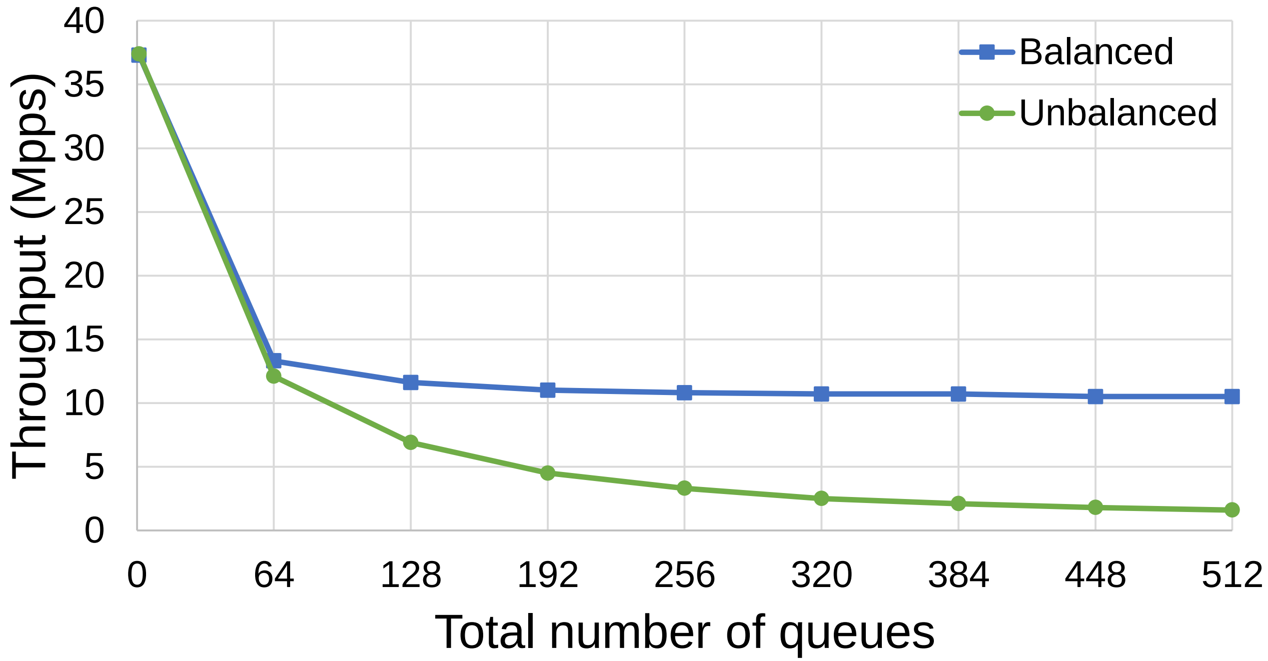
- Round-trip latency of packet forwarding
- Light traffic (minimal queuing delay)



Latency is severely affected as queue heads fall out of L1/L2 caches

Effect on Peak Throughput

- Balanced traffic: Passing through all queues
- Unbalanced traffic: Passing through only one queue

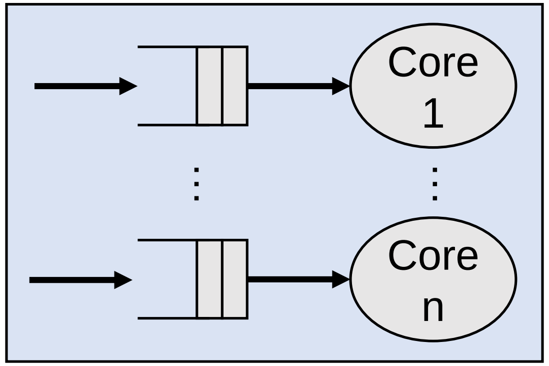


Cache misses not interleaved with transmits severely hurt peak throughput in unbalanced traffic

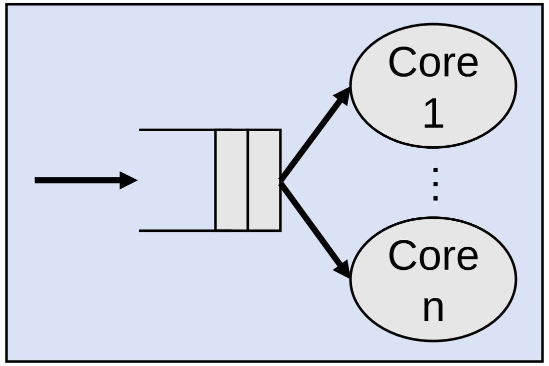
1
2
3

Scale-up Queuing Is Impractical

- (a) Scale-out vs. (b) Scale-up queuing (shared queue)



(a)



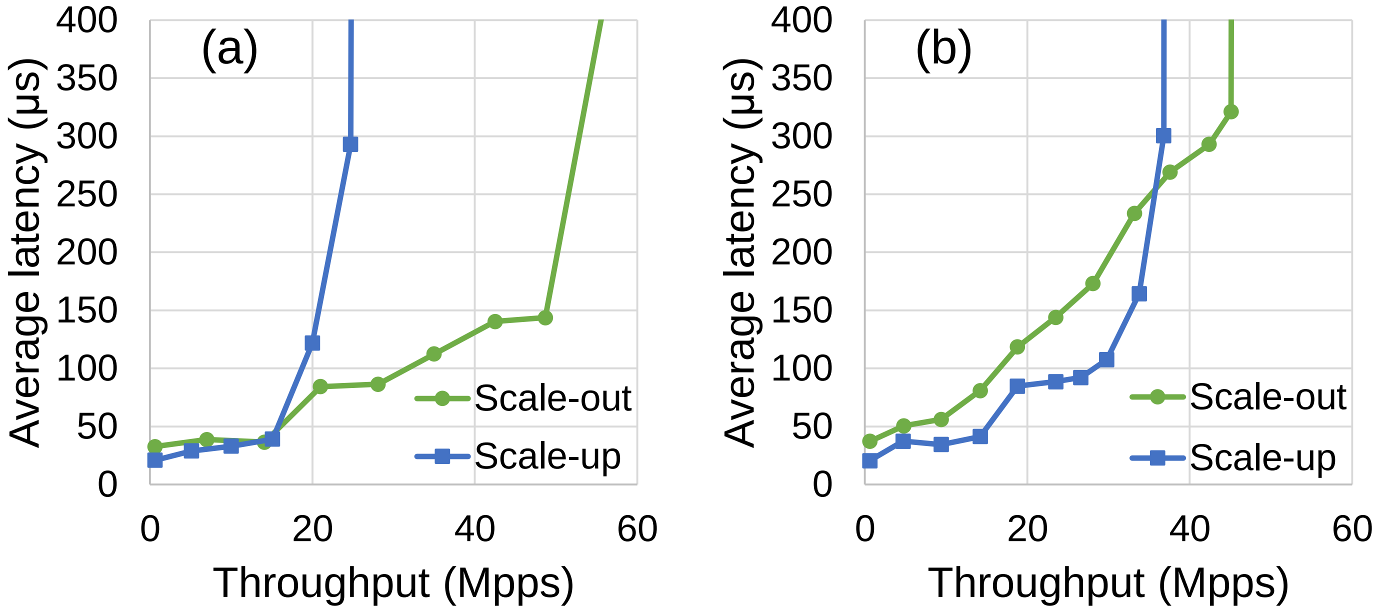
(b)

- Scale-up queuing
  - Strong theoretical merits
  - Synchronization disadvantage

1
2
3

Scale-out vs. Scale-up

- Processing hiccups cause head-of-line (HoL) blocking in scale-out
- Round-trip latency with 10 parallel cores
  - (a) No hiccups
  - (b) 1µs processing hiccup with 1% probability



1
2
3

Although effective in avoiding HoL blocking,
spin-polling in scale-up queuing saturates at lower loads

Future Data Planes

Solution Direction(s)

- `QWAIT`, a multi-address monitoring scheme
  - Inspired by x86 `MWAIT`
  - Avoids polling tax, useless polling, and disruption to SMT co-runners
  - Needs hardware support

- Programming model similar to `select-case` in Go

```
QWAIT (queue_set):
    case queue_1:
        process_queue_1();
    …
    case queue_n:
        process_queue_n();
```

Conclusion

- Key mechanisms of software data planes
  - User-level shared queues
  - Spin-polling cores

- Although easy-to-use and low-latency, software data planes have deficiencies, especially when scaled

- Using DPDK, we quantified these deficiencies:
  - Incurring polling overhead and useless work
  - Not scalable to many cores/queues
  - Not well-suited for scale-up queuing

Q & A

Thank you!