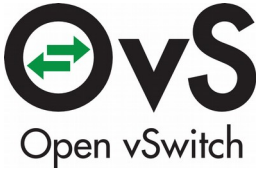
Converging Approaches in Software Switches

Ben Pfaff
VMware



The Point

There are two main ways to build software switch pipelines:

“code-driven”

and

“data-driven”

Usually, these are considered to be alternatives.

They can actually be complementary.

Open vSwitch

Irrelevancies

The distinction I am making is not about packet I/O methods like:

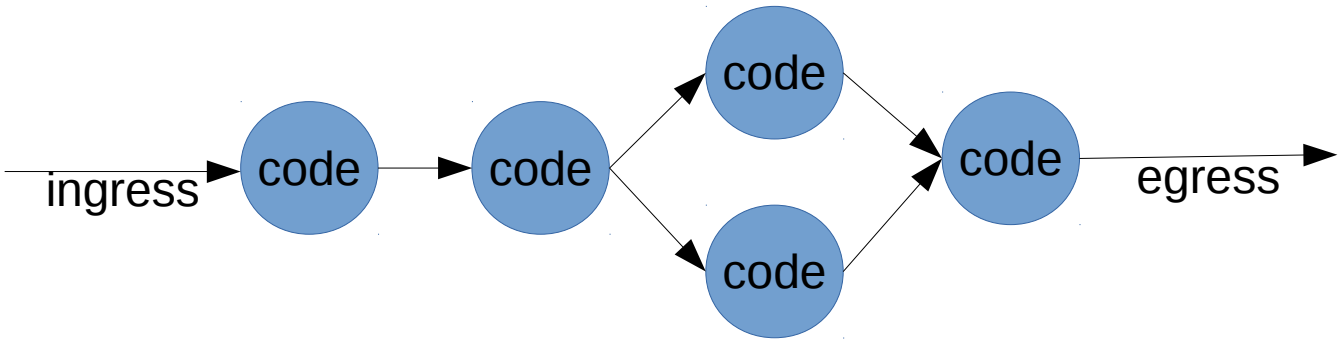
- Custom kernel module
- AF_PACKET sockets
- DPDK
- Netmap

Packet I/O is key to performance but not to switch pipelines.

Open vSwitch

Code-Driven Switch Pipeline

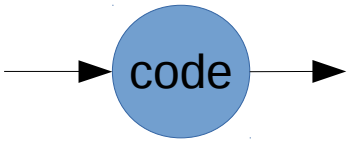
Executes series of code fragments (“stages”) per packet.



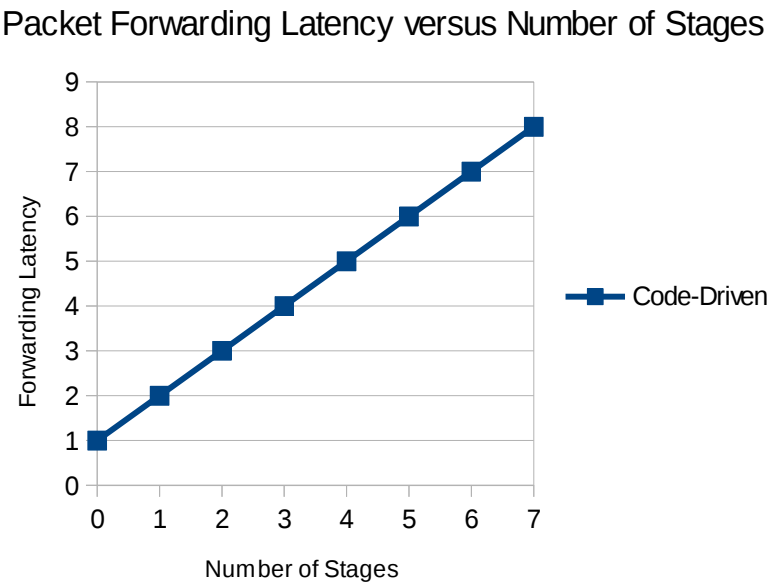
- Obvious.
- Loose coupling.

Open vSwitch

Code-Driven Pipeline Stages



- Can do anything or nothing.
- Each stage increases per-packet latency.
- Near-zero fixed overhead.
- Therefore: null pipeline is very fast.



Open vSwitch

DPDK and Netmap Are Not Software Switches

DPDK and Netmap are packet I/O methods.

Early publications compared them against software switches.

This is unfair: compare them against other packet I/O methods instead.

Some Code-Driven Switches

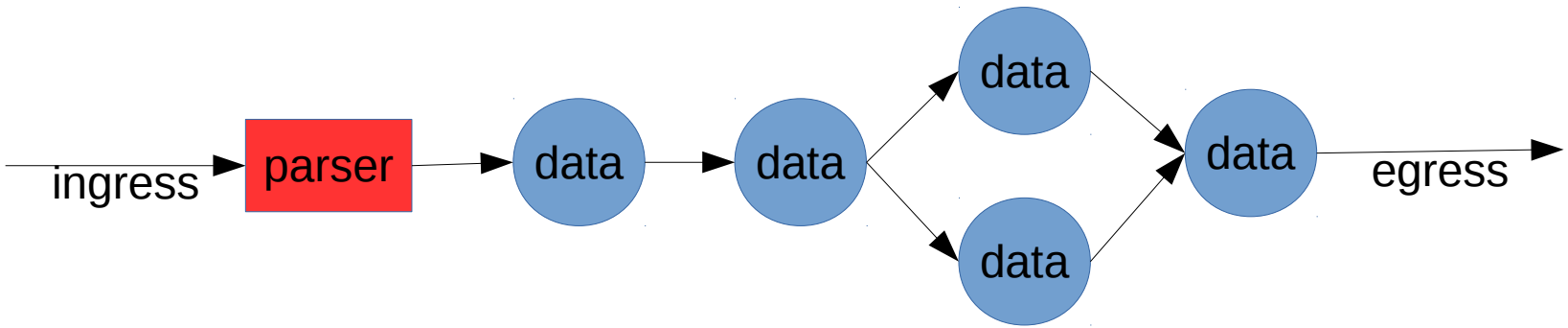
in chronological order

1) Linux bridge + iptables + ebtables + …
2) Click
3) VMware VDS
4) VMware NSX Edge
5) VPP
6) BESS

Open vSwitch

Data-Driven Switch Pipeline

A single engine drives each packet through all the stages,
each of which is a data table.



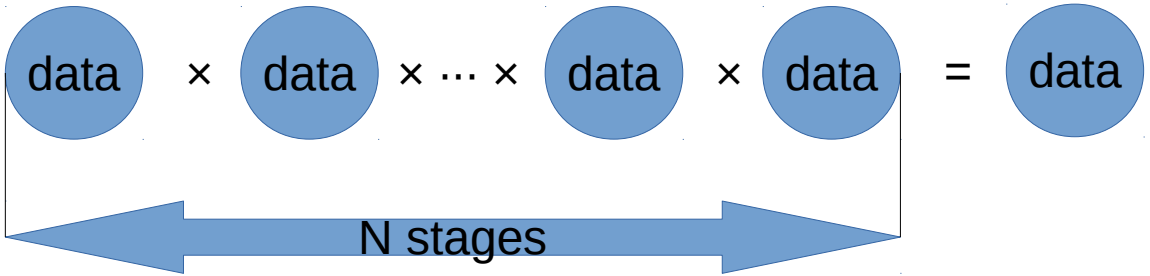
- Unnatural for programmers
- Limited by engine's capabilities
- Parsing is expensive

+Parsing happens only once per pipeline

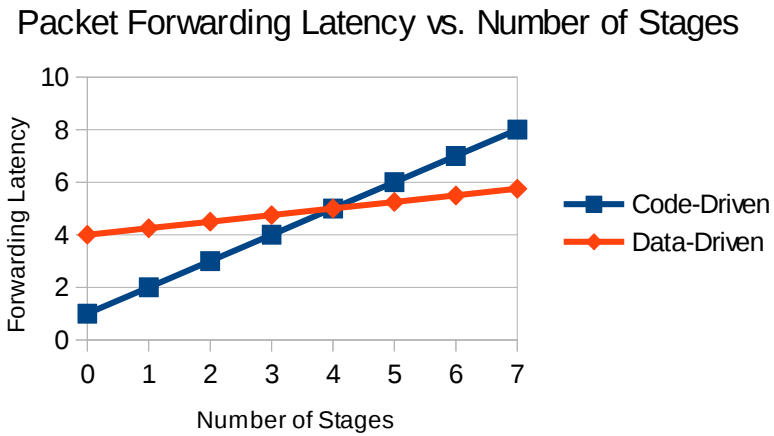
Open vSwitch

Data-Driven Pipeline Stages

N stages can be cross-producted into 1 stage (see NSDI 2015 paper):



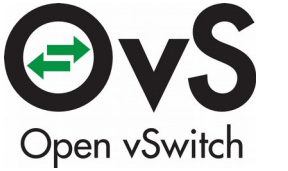
- High fixed cost.
- Adding stages adds little per-packet latency.
- Therefore: null pipeline is slow, complex pipeline is fast.
- Hardware classification offload is possible.



Open vSwitch

Some Data-Driven Switches

- Open vSwitch
- MidoNet



Crossover

Can we combine strengths of both approaches?

Code-driven:

+ Low fixed overhead.

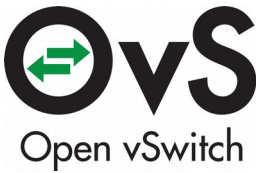
+ Flexibility.

Data-driven:

+ Low per-stage overhead.

+ Common parser.

I don't have a complete answer but I have some thoughts.



Code-Driven Moving Toward Data-Driven

Are you skeptical?

“If a data-driven pipeline is faster than a code-driven one, for some application, then the code-driven pipeline code is badly written.”

But I have two data points:

1. VMware VDS
2. VMware NSX Edge

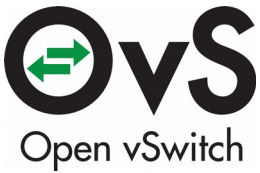
Data-Driven Moving Toward Code-Driven

Attack sources of fixed overhead:

- Cost of parsing, by parsing less.
- Cost of classification, by hardware offload (which is not just for high-priced specialized hardware).

Increase flexibility:

- Integrate arbitrary code, via eBPF/P4.
- Integrate external code, e.g. kernel conntrack, NAT.
- Integrate into pipelines of middleboxes: SoftFlow.



Conclusion

Two seemingly different software switch pipelines, “code-driven” and “data-driven,” may ultimately move closer to one another than they started out.

Open vSwitch