Converging Approaches in Software Switches

Combining code- and data-driven approaches to achieve a better result

Ben Pfaff
blp@ovn.org

August 4, 2016

Agenda

1	The Point
2	Definitions and Distractions
3	Code-Driven Switches
4	Data-Driven Switches
5	Code-Driven → Data-Driven
6	Data-Driven → Code-Driven

The Point

There are two main ways to build software switches:

"code-driven"

and

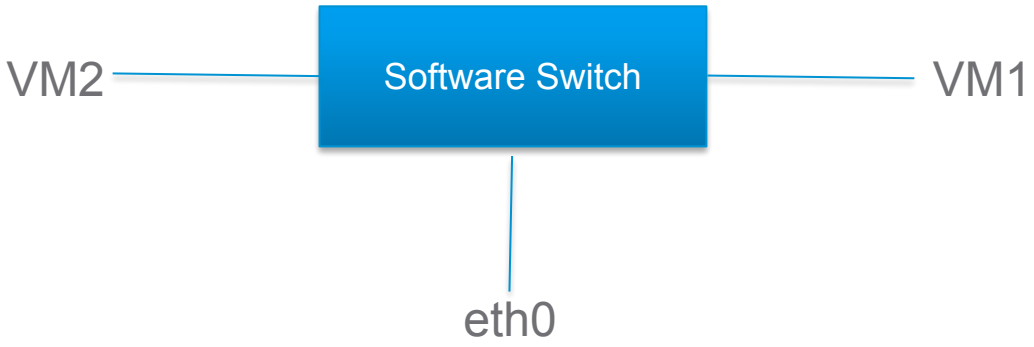
"data-driven"

Usually, these are considered to be alternatives.

They can actually be complementary.

What's a Software Switch?

- A software switch forwards Ethernet packets among physical or virtual ports.
- Necessary for VMs to share physical Ethernet ports.
- An old idea.
- Many exist today, including the following developed at VMware (and many others):
  - vSphere Distributed Switch (VDS)
  - NSX Edge Switch
  - Open vSwitch



Distractions

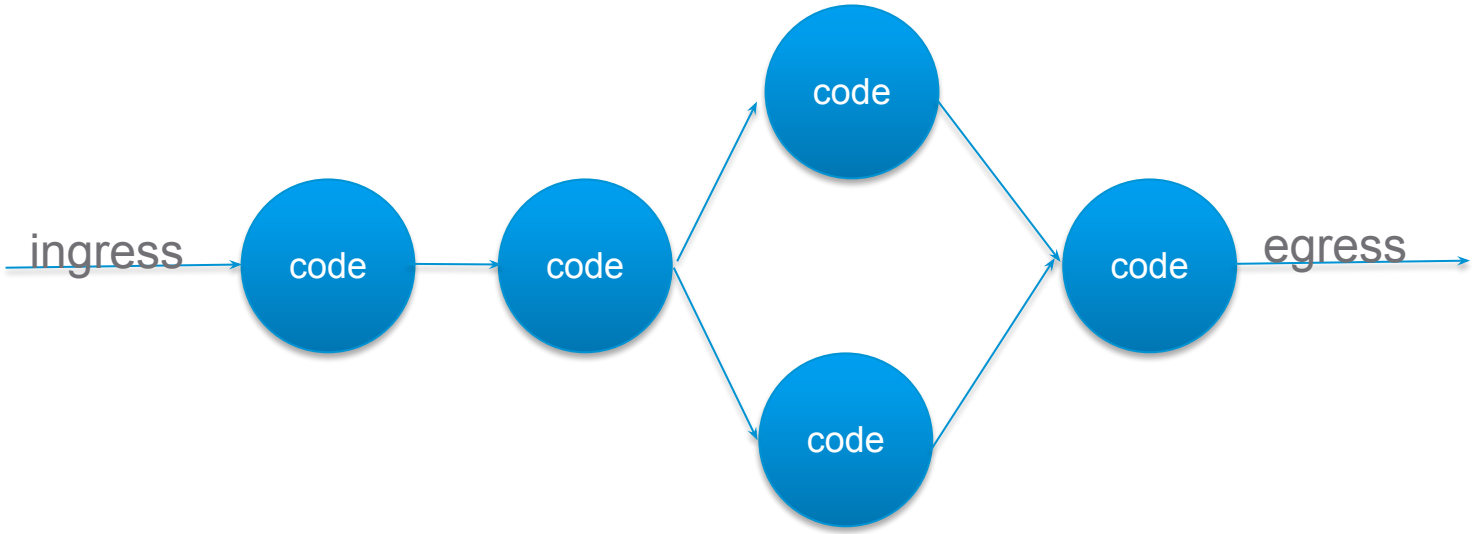
These choices affect implementation and performance, but they are not related to the architectural distinction I'm talking about.

- Kernel vs. Userspace
  - All-kernel
  - All-userspace
  - Mix
- Packet I/O method
  - Custom kernel module
  - AF_PACKET sockets
  - DPDK (kernel or userspace)
  - Netmap

It's unfair to compare software switches against packet I/O methods, but advocates of packet I/O methods often do it.

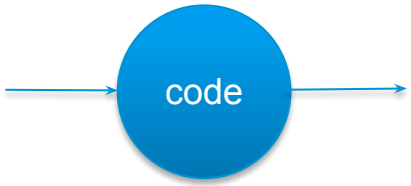
Code-Driven Switch Pipeline

Executes series of code fragments ("stages") per packet.

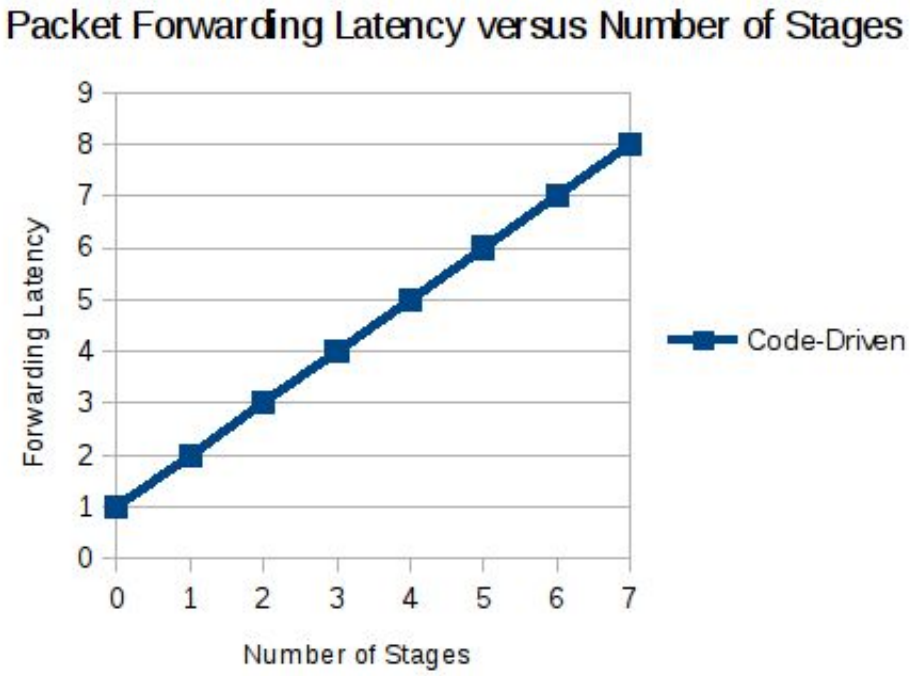


- Obvious
- Loose coupling

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.



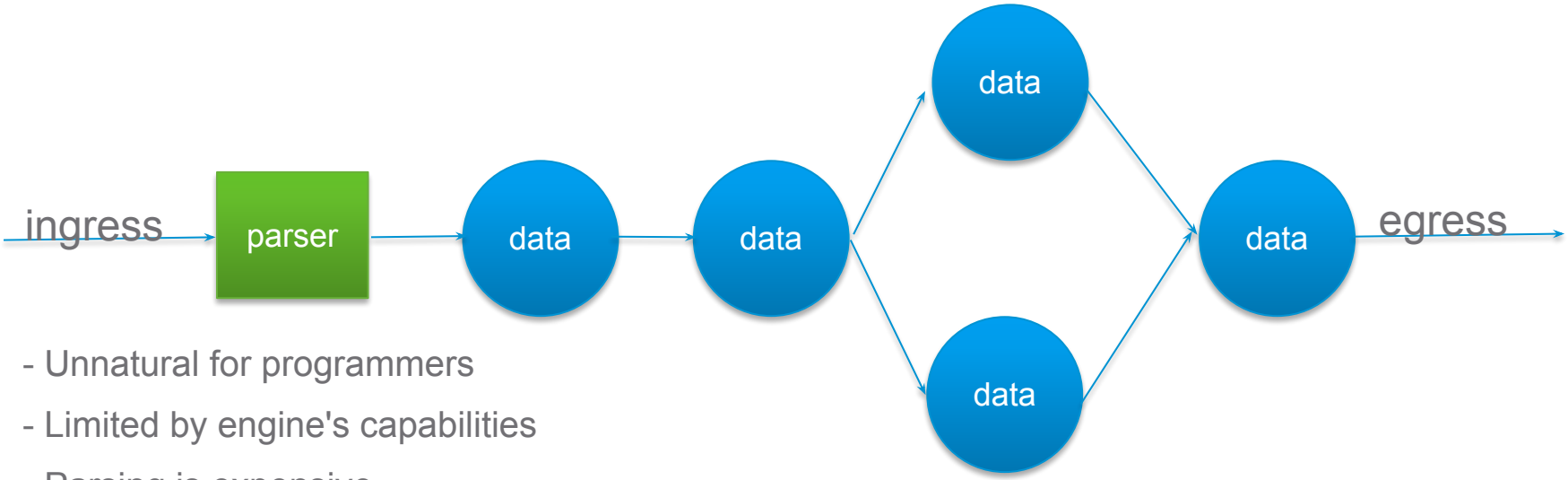
Some Code-Driven Switches

- Linux bridge + iptables + ebtables + …
- Click
- VSphere Distributed Switch (VDS)
- NSX Edge Switch
- VPP
- BESS

This super-dramatic slide was in the corporate slide template and I felt I had to use it somewhere.

Data-Driven Switch Pipeline

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

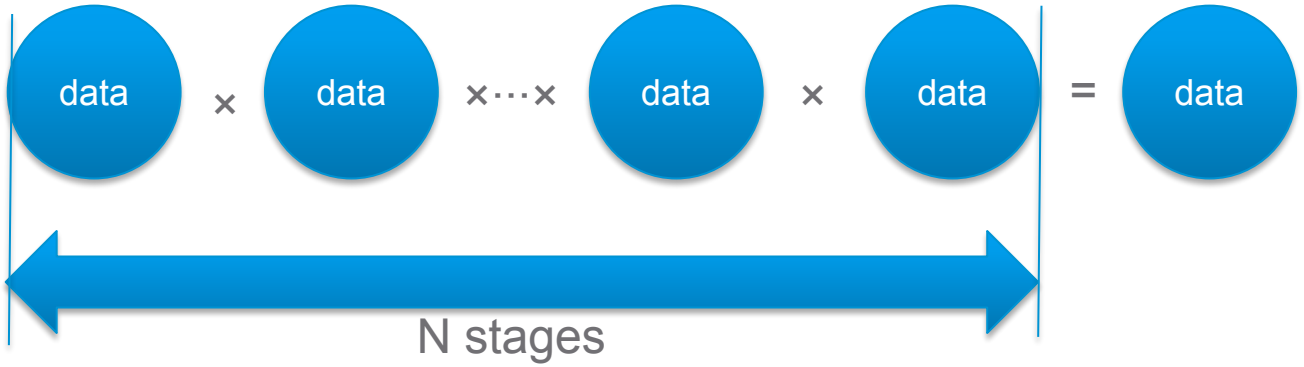


- Unnatural for programmers
- Limited by engine's capabilities
- Parsing is expensive

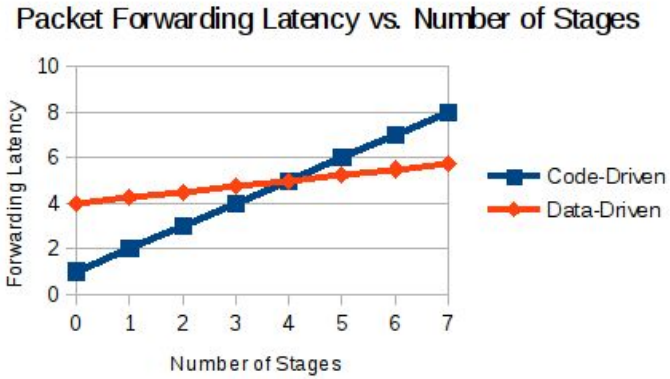
+ Parsing only happens once per pipeline.

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.



Some Data-Driven Switches

- Open vSwitch
- MidoNet

Code-Driven ↔ Data-Driven Crossover

Can we combine the 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 → Data-Driven

Are you skeptical?

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

But I have two data points:

1. VSphere Distributed Switch
2. VMware NSX Edge

Data-Driven → Code-Driven

Attack sources of fixed overhead:

- Cost of parsing, by parsing less.
- Cost of classification, by hardware offload (which can really be real on common 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.

Thank You

Ben Pfaff
blp@ovn.org

For more information about Open vSwitch,
please visit openvswitch.org.