/delivers/

Dyn

Performance impact of contained and virtualized environments in Authoritative DNS Servers

João Damas

Dave Knight

DNS OARC - May 2014

What are we up to?

Virtualized servers are the new unit of deployment

- Flexible provisioning
- Resource control (RAM, CPU)
- Component isolation
- We want the operational benefits of Virtual Machines, but…

Our problem

DNS servers and VMs haven't gone well together in the past

- Bottlenecks in access to NICs
- Supervisor running single threaded for network activity
- Not really meant for network intensive systems

Resource utilisation

- Each VM runs its own kernel: RAM usage
  - lower density of resource utilization

Time goes by, things change

VM with SR-IOV

- Allows a PCIe device to appear to be multiple virtual devices
- I350 does up to 7 Virtual Functions per port
- We can map a VM interface directly to a Virtual Function

Linux containers (chroot on steroids)

- No additional running copies of the full OS
  - Conserves RAM
- Separate namespace, filesystem, network, privileges

What do we measure?

We want to compare the behaviour of common authoritative servers in different environments.

Knot (1.4.5), NSD (4.0.3) and BIND (9.9.5)

Baseline on bare OS (Ubuntu 13.10 & 14.04)

Containers, unconstrained and with various CPU pinnings

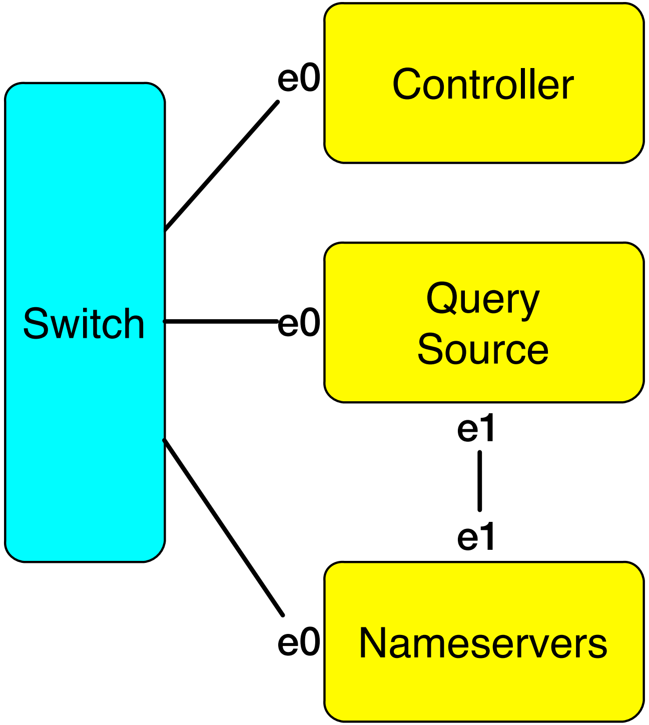
VMs, bridged versus sr-iov

Stability of results

Computers seem to be complicated enough that they are no longer deterministic in their behaviour

- many factors, little details count, double check
- different setups scale differently
  - CPU type
  - number of hosted zones

Lab network



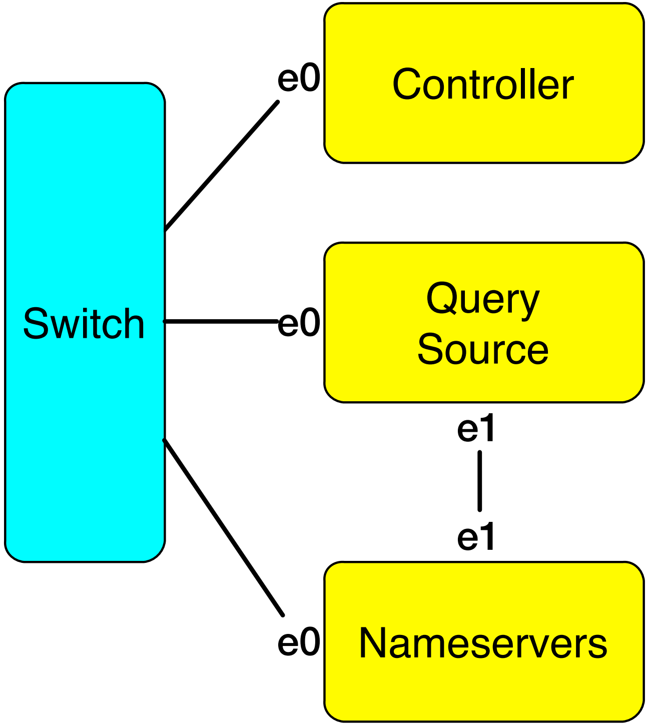
Install server

Test scripts run here

dnsperf runs here

VMs, containers, nameservers run here

Lab configuration



Ubuntu 12.04 LTS

Test control scripts ssh as root to test boxes

Ubuntu 14.04 LTS

dnsperf 2.0.0.0-1

Ubuntu 14.04 LTS

QEMU 2.0.0 (KVM), Docker 0.8.1

BIND-9.9.5-W1, NSD-4.0.3, Knot DNS 1.4.5

Lab operation

Automated reinstall of test servers between tests.

Nameserver build, Docker image creation, VM configuration all fully scripted.

Tests fully scripted, ie:

```
foreach server (bind nsd knot)
    foreach workers (1..8)
       foreach pinning (0 0,4 0,1,4 0,1,4,5)
          start nameserver_container
          run dnsperf
```

Testing VMs

We assume that we should not run high performance nameservers in VMs.

New technologies like sr-iov challenge this assumption.

We compare performance of a nameserver running inside a VM attached to the network with

- Bridge
- SR-IOV Virtual Function

VM with a plain old bridge

Bridge devices mapped to physical NIC ports

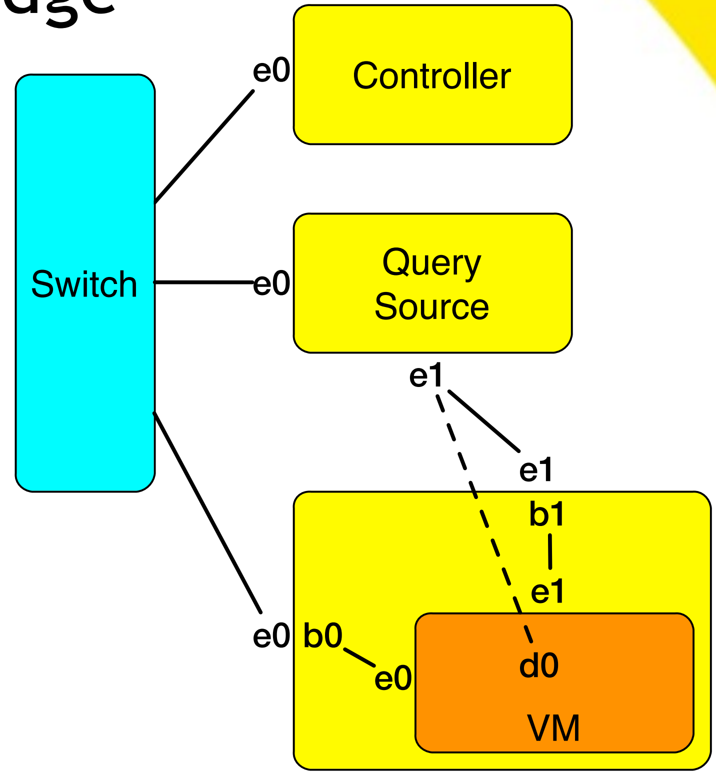
b0 -> e0 , b1 -> e1

VM interfaces mapped to bridges

e0 -> b0 , e1 -> b1

VM interfaces can talk on the physical LANs

Dummy interface inside the VM with the service address



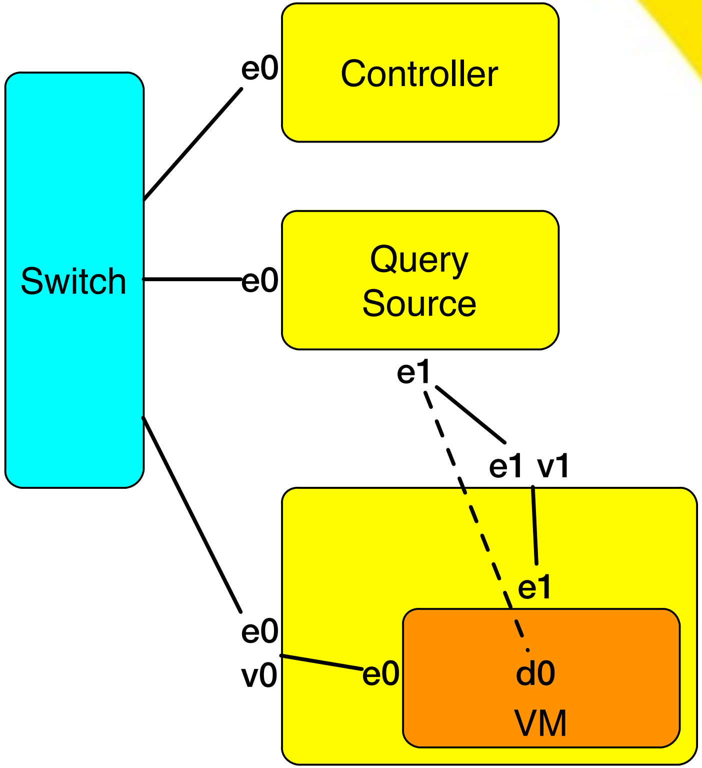
VM with sr-iov

No bridge

VM interfaces mapped directly to NIC Virtual Functions

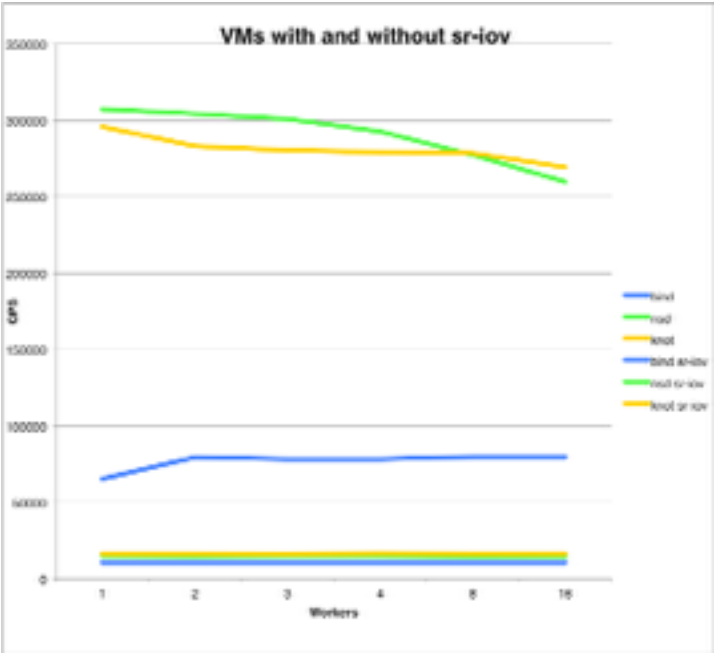
VM interfaces can talk on the physical LANs

Dummy interface inside the VM with the service address



More about sr-iov: http://blog.scottlowe.org/2009/12/02/what-is-sr-iov/

Bridge vs sr-iov



sr-iov gives a massive performance increase!

Testing Containers

Using Docker.io

- Easy container builds and management
- Add a nameserver and dependencies, atop a base image
- Easy to start/stop
- Easy to ship updates across the network

Dyn

Docker with NAT

Docker's default behaviour:

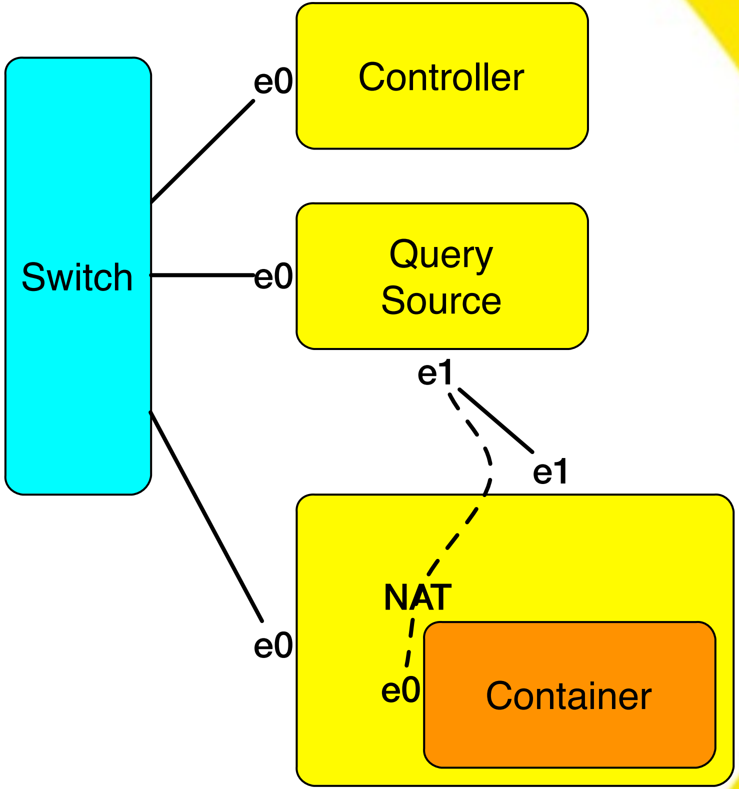
Container's eth0 has a dynamic private address.

Public service address:port mapped onto that with iptables NAT.

iptables NAT is stateful

Bitter experience suggests that state in front of a high traffic nameserver is bad.

We don't actually need to track state here.



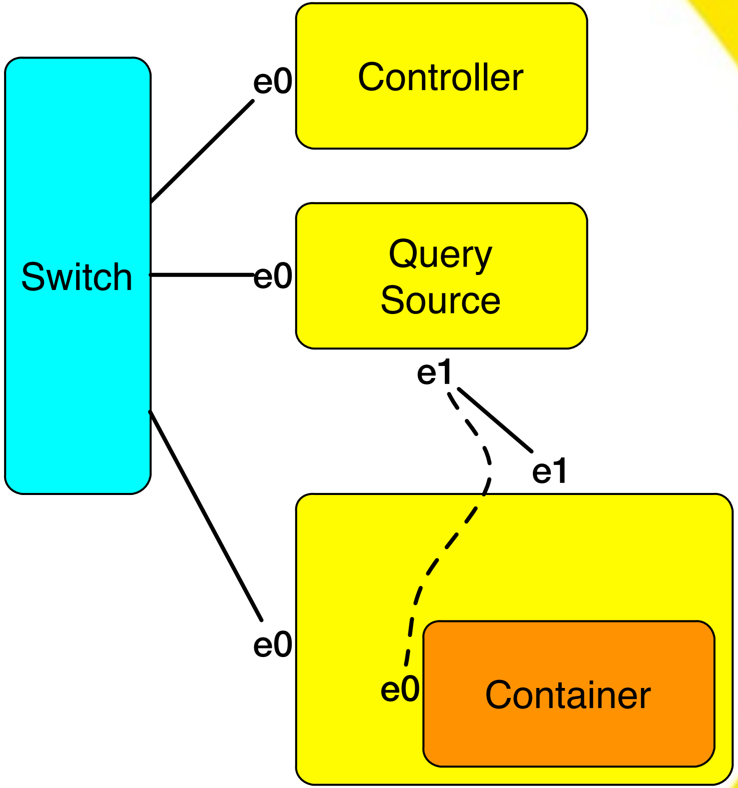
Docker without NAT

If we configure the service address inside the container we don't need NAT.

Docker doesn't do this.

We can't run ifconfig inside a Docker container without disabling it's security.

However, the ip tool can do network namespaces, we can add an alias to the containers e0 from outside the container!



Container Resource Constraints

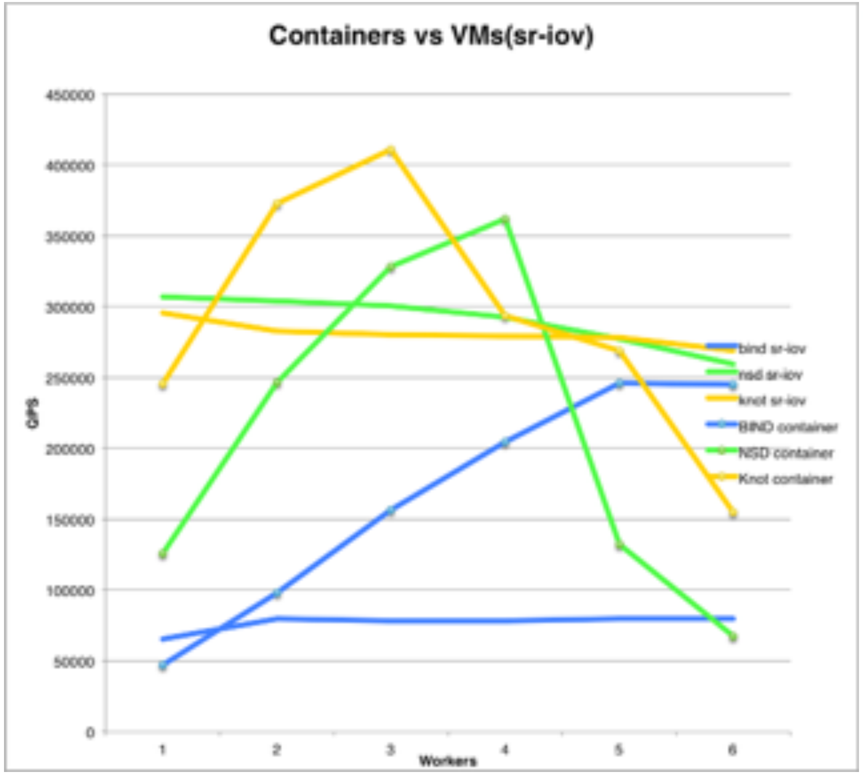
Docker lets us

- Define a memory limit for a container
- Assign CPU priority to a container
- Pin a container to specific CPU/core/thread

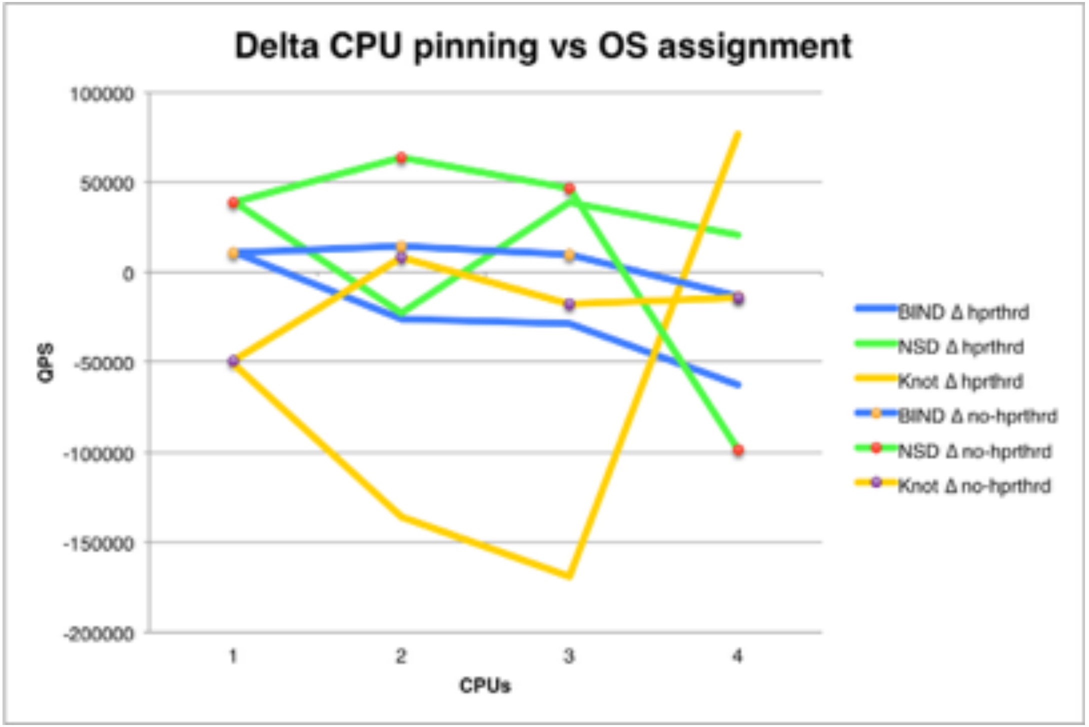
We compare performance of nameservers with different configurations of CPU pinning / number of workers.

VMs (sr-iov) vs Containers

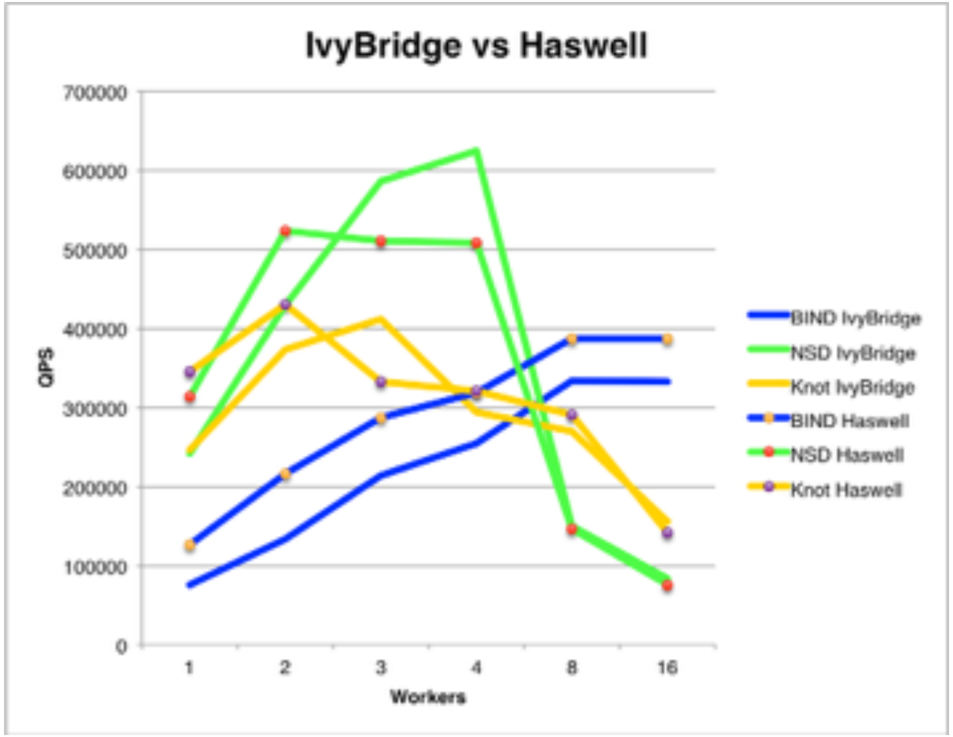
- Containers are slower for 1 CPU
- Containers scale better



Containers with CPU pinning



Haswell vs IvyBridge



Questions?

Dyn