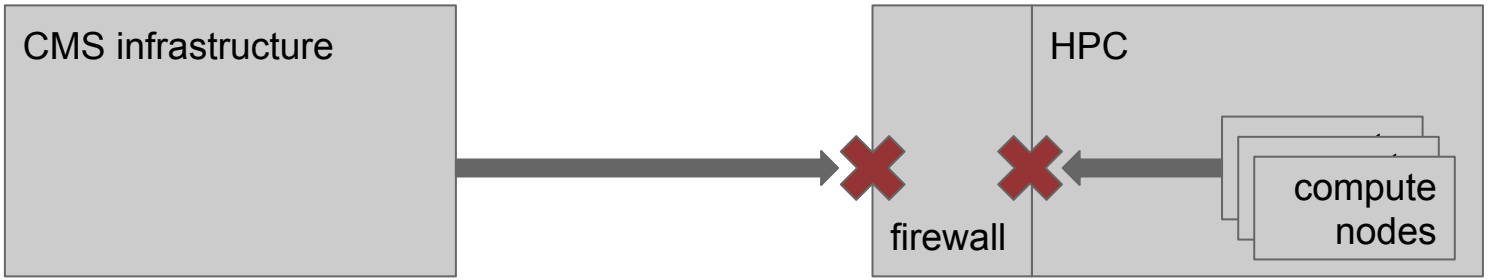
Harnessing HPC resources for CMS jobs using a Virtual Private Network

Ben Tovar, Brian Bockelman, Michael Hildreth,
Kevin Lannon, Douglas Thain

Problem Statement

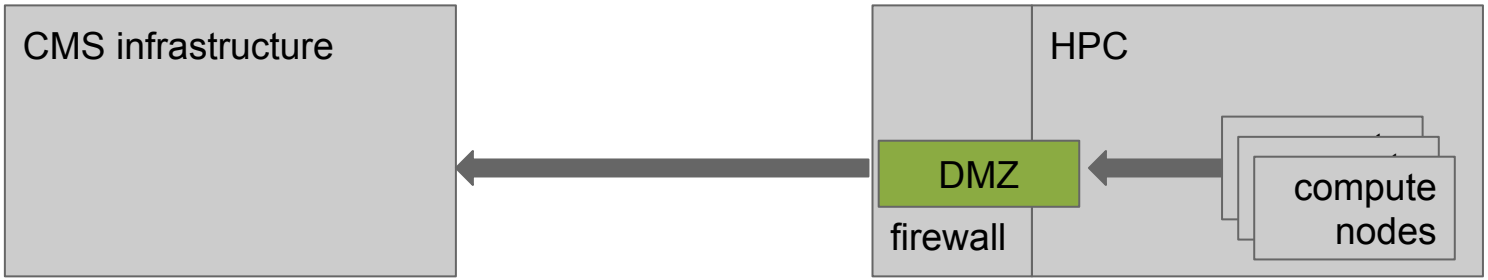
- Resources available (e.g. HPC's) for CMS may be behind firewalls.
  - How can we connect these resources to the rest of the CMS production infrastructure?
  - How can we provide network resource limits guarantees to system administrators?



Key Observation

Take advantage of Demilitarized Zones (DMZ's) to link HPC's resources to CMS.

A DMZ provides an isolated interface to the outside network.

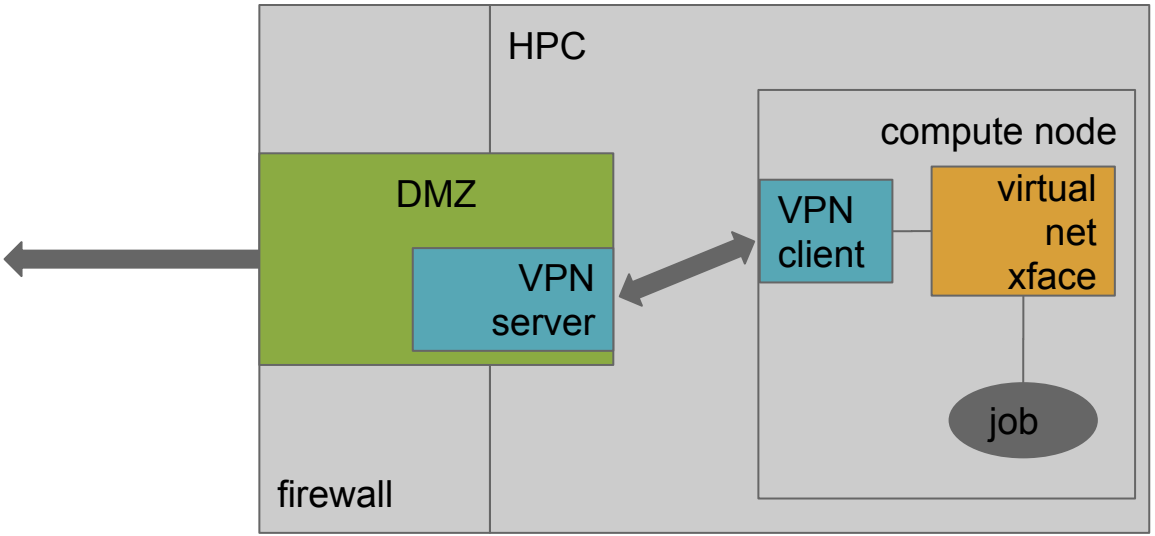


Solution Requirements

1. jobs must run on compute nodes without administrator privileges. As far as the HPC is concerned, the job is just another user job
2. the (CMS) job description should remain unchanged
3. network usage limits guarantees should be given and enforced.

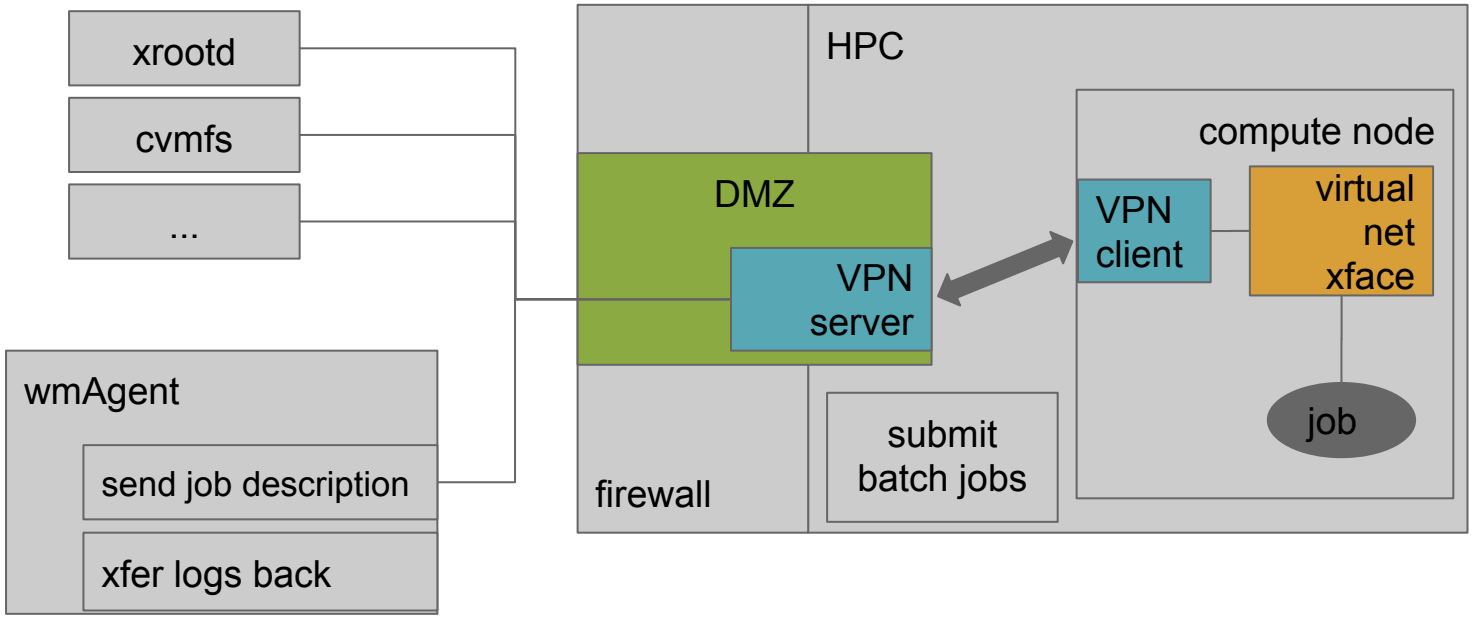
Construct a Virtual Private Network (VPN):

- VPN server runs in the DMZ
- VPN server provides IPs, DNS, and routes to VPN clients managing a virtual network interface at the compute nodes.
- All network traffic of the compute node is managed by the VPN.



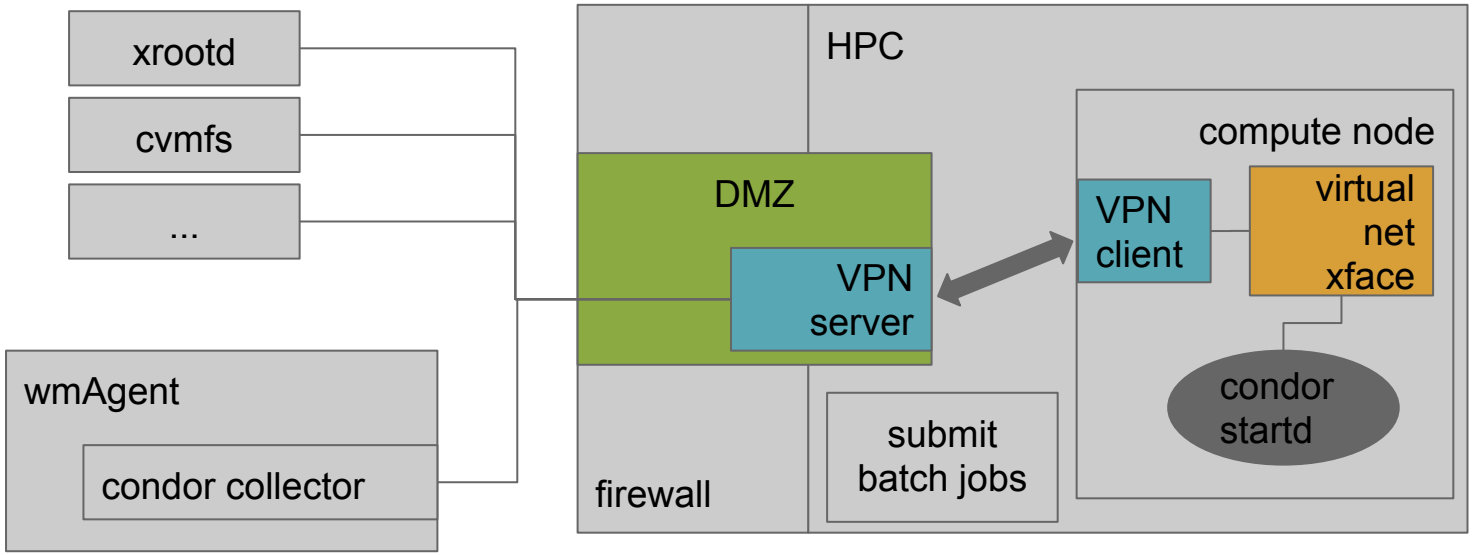
Possible Uses for CMS

a) Running production jobs directly



Possible Uses for CMS

b) jobs call back as condor startd's (unmodified wmAgent)

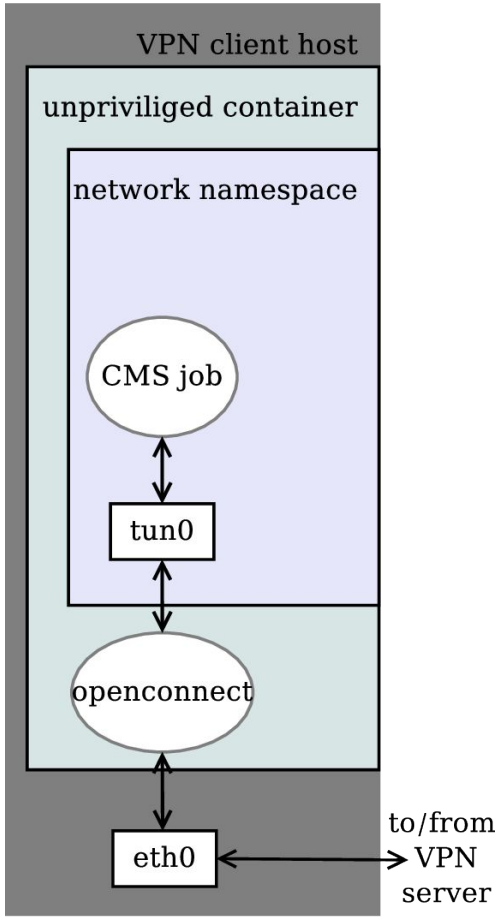


Implementation - VPN client

openconnect --script vpnns --attach

tun virtual net xface:
simulate network layer
manipulate IP packets

It can be controlled with tools such as tc.

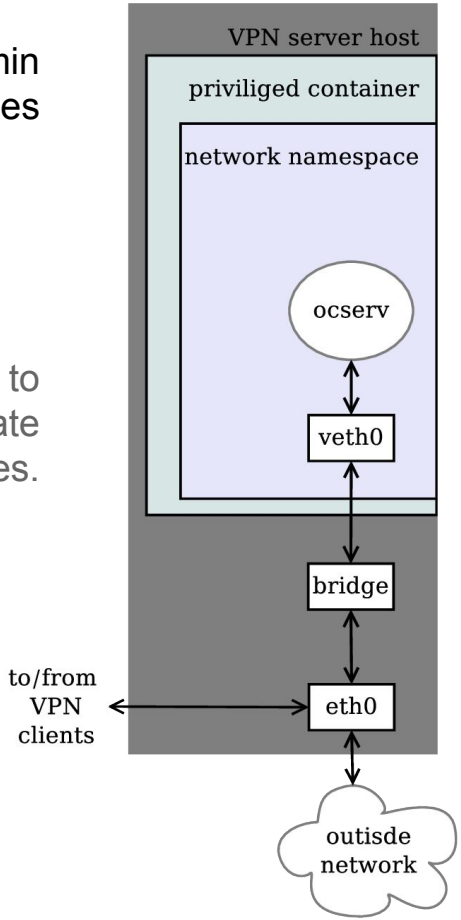


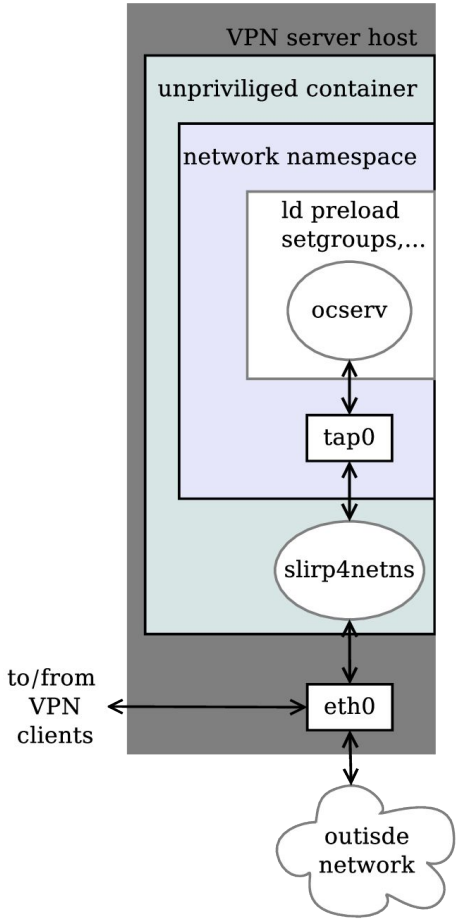
- If namespaces not enabled, fall back to LDPRELOAD and tsocks (see M. Mariotti, D. Spiga, and T. Boccali)

Implementation - VPN server

without admin privileges

ocserv needs to manipulate Ethernet frames.





without any admin privileges

tap interface to simulate link layer.

slirp4netns provides a full network stack in userspace

IO Tests

bandwidth in Mbps

network links: 1Gbps

from VPN server to VPN clients (iperf)

xfers	server stack	VPN client	avg per transfer	avg concurrent
1	kernel	no	871.98	871.98
		namespace	333.59	333.59
		socks	196.61	196.61
	user	namespace	196.06	196.06
2	kernel	no	470.99	926.84
		namespace	298.68	597.36
		socks	190.54	381.09
	user	namespace	103.24	206.48
10	kernel	no	94.87	890.56
		namespace	88.26	882.64
		socks	86.88	868.87
	user	namespace	19.96	199.67
20	kernel	no	47.63	846.01
		namespace	44.24	884.90
		socks	43.56	871.35
	user	namespace	7.24	144.83

tcp over tcp (no dtls openconnect script)

Monte Carlo production jobs test

concurrent jobs	VPN client	cvmfs	median runtime (minutes)
50	namespace	cvmfsexec	45.13
	namespace	fuse	41.67
	no	fuse	41.82
100	namespace	cvmfsexec	47.75
	namespace	fuse	50.71
	no	fuse	48.90
200	namespace	cvmfsexec	59.91
	namespace	fuse	60.78
	no	fuse	60.20

(Jobs ran at opportunistic resources at a Notre Dame cluster)

Code & Questions

https://github.com/cooperative-computing-lab/userlevel-vpn-tun-tap

(container recipes, configuration tips, launching scripts)

```
$ git clone https://github.com/cvmfs/cvmfsexec.git
$ cd cvmfsexec
$ ./makedist -s -m rhel7-x86_64 osg
$ ./makedist -s -o /tmp/singularity-cmvfsexec
$ cd ..
$ export SINGCVMFS_REPOSITORIES=cms.cern.ch,atlas.cern.ch,oasis.opensciencegrid.org
$ ./launch-vpn-client --image context/openconnect-container/vpncms-client.sif \
     --server MACHINE_WHERE_OCSERV_RUNS:8443 \
     --servercert pin-sha256:XXXXXXX... \
     --user myvpnuser \
     --passwd myvpnpasswd \
     --vpn-mode ns \
     --singularity /tmp/singularity-cmvfsexec \
     -- ls /cvmfs/cms.cern.ch
```

questions: btovar@nd.edu