Enabling Research Network Connectivity to Clouds with Virtual Router Technology

R Seuster, K Casteels, CR Leavett-Brown, M Paterson and RJ Sobie
University of Victoria, 3800 Finnerty Road, BC, V8P 5C2, Canada
E-mail: Rolf.Seuster@uvic.ca

Abstract. The use of opportunistic cloud resources by HEP experiments has significantly increased over the past few years. Clouds that are owned or managed by the HEP community are connected to the LHCONE network or the research network with global access to HEP computing resources. Private clouds, such as those supported by non-HEP research funds are generally connected to the international research network; however, commercial clouds are either not connected to the research network or only connect to research sites within their national boundaries. Since research network connectivity is a requirement for HEP applications, we need to find a solution that provides a high-speed connection. We are studying a solution with a virtual router that will address the use case when a commercial cloud has research network connectivity in a limited region. In this situation, we host a virtual router in our HEP site and require that all traffic from the commercial site transit through the virtual router. Although this may increase the network path and also the load on the HEP site, it is a workable solution that would enable the use of the remote cloud for low I/O applications. We are exploring some simple open-source solutions. In this paper, we present the results of our studies and how it will benefit our use of private and public clouds for HEP computing.

1. Introduction

Clouds are providing a growing fraction of computing resources to the research community. These can be dedicated, localized clouds which are typically owned by a university or a research institution. On the other side are opportunistic clouds. Some of these providers open their clouds for research, for example by either providing grants or sometimes early access during the commissioning or access during periods of low usage. To gain access to a suitable number of machines, users might need to combine several clouds for their usage. We at the University of Victoria are in this situation, where we provide computing on clouds for currently two experiments, the ATLAS collaboration at the Large Hadron Collider at CERN in Switzerland and the Belle II collaboration at the asymmetric electron-positron collider at KEK in Japan. Jobs run on in the order of 10 clouds, from a local small cluster for development at our university, to resources provided by Compute Canada\(^1\), other research clouds worldwide.

For both experiments, jobs submitted into our system appear to be running locally at our university although they in fact run on computers far away, some even on a different continent. This can have big consequences on these jobs, the distance and therefore the network latency to services can be severely different between sites. Depending on the type of network, there

\(^1\) http://computecanada.ca
might also be issues with connectivity mostly to research networks, see Fig. 1. This can lead to problems where traffic for upload or download of data for the jobs might be charged by the cloud provider or, less severe, host names to services might not resolve. Changing to public DNS servers can usually mitigate the last problem.

Figure 1. Potential problem with distributed clouds, where jobs can run far away from the controlling home institute. Data paths on the network are undefined, and might incur fees. Connections also might be cut, as they often show unpredictable patterns quite different from commercial traffic.

2. The Problem
One of our clouds has a particular problem with connectivity to external networks. Often the job scheduler, HTCondor \(^2\) we use, loses connectivity to its master daemon running locally at our university, leading to HTCondor thinking this job is lost. Most likely, the extensive but legitimate traffic from virtual machine to different storage elements can trigger the denial-of-service algorithms of an edge router, and connections are cut. About 15 minutes later connections can often be restored, but during this time HTCondor looses connection to the job and declares it as 'lost'.

3. Previous and Related Work
Similar work on enabling research network connectivity for mainly commercial clouds has been presented for example at previous LHCOPN-LHCONE meetings [1] and [2]. Their works also utilize software tunneling protocols than this study but also include modifications to network routing and other hardware solutions: changes within the cloud infrastructure that we explicitly wanted to avoid in this study.

4. Network Tunnels to improve Connectivity
In this paper we investigate a simple software solution for providing a network tunnel back to the home institute, to avoid these problems. First, we compare three different solutions with respect to their impact on network bandwidth. Later, we describe some experience we gained running of these solutions in production.

The basic idea is that we relay all external traffic back to the home institute via a tunnel. For the job it indeed looks like the job would run at the home institute, except for different distances

\(^2\) http://htcondor.org
and latencies to storage elements and services (Fig. 2) external traffic out of the cloud will only be going to the home institute. This means that in our case, the traffic will definitely stay on the research network. Furthermore, any additional required exceptions and white-listing can be kept to an absolute minimum, as the endpoint of all external traffic visible from the cloud’s point of view is another research institution, our home institute.

Figure 2. A tunnel back to the home institute can provide several advantages. The network connectivity is well. For jobs it appears as they would run at the home institute. All jobs will either produce only local traffic or external traffic to a single site, exceptions and white-lists can be kept to an absolute minimum.

4.1. Test-bed
We evaluated the upload and download performance between two servers at our university, one real server and one virtual machine both connected with 10G network. The MTU of both network interfaces was at 1500 and no tuning of kernel parameters was done.

To simulate long distance connections, artificial latencies were introduced by the ‘netem’ \(^3\) package from the Linux kernel: It can emulate various network related features. For example, it can emulate a fixed or variable network delays or an configurable amount of packet loss. Latencies can follow simple distributions. Some basic correlations between the lost packets can also be emulated: A packet following a lost packet will have a configurable, higher probability to be lost as well.

We emulated fixed latencies of 0ms, 2ms, 5ms, 10ms, 20ms, 30ms, 40ms, 50ms, 100ms and 200ms. These latencies roughly corresponding to data transfers found typically on campus (0-2ms), in close vicinity (5-20ms), in the same global region (30-50ms) or across continents (100-200ms).

We used two different programs for the data transfer. The first program was ‘scp’, a well established and performant. It provides a good security, and with its counterpart ‘ssh’ it is used daily by a huge amount of users.

The other programs from the ‘davix’ project\(^4\) are open source software which is developed at CERN. It provides single clients for the protocols HTTP, WebDAV, Microsoft Azure and Amazon S3. For authentication, it supports the X509 proxy certificate extensions which is commonly used in HEP community.

\(^3\) \url{https://wiki.linuxfoundation.org/networking/netem}
\(^4\) \url{https://dmc.web.cern.ch/projects/davix/home}
4.2. Software Solutions for Tunneling
Several software products can create with tunnels which provide us with the required research network connectivity:

- **OpenVPN** The open source version of OpenVPN \(^5\) is a full-featured SSL VPN solution that accommodates a wide range of configurations, including remote access, site-to-site VPN, Wi-Fi security, and enterprise-scale remote access solutions with load balancing, fail-over and fine-grained access-controls. Most of these features were not used in our tests, but we appreciated the fairly easy installation, also by the clear instruction given and many good examples for configuration in the documentation. Many jobs can connect to one client which can have advantages or disadvantages. A single connection from a dedicated client server on the cloud back to the home institute could be beneficial in terms of setup and performance degradation of the virtual machines. On the other side, if this single connection fails, potentially a lot of jobs would be affected at the same time.

A full compilation of the software was required, because the mismatch of versions on the two servers did not work well together. Additionally, some TCP or UDP ports needed be opened on firewalls, even when using the default ports.

- **GRE** Generic Routing Encapsulation \(^6\) is a tunnel protocol developed by Cisco. It can encapsulate a wide variety of network layer protocols inside virtual point-to-point links over an Internet Protocol network and is supported by many operating systems. Typically no further installation is required, because GRE is supported by the Linux kernel since many versions. One can find easily plenty of examples, however some might be outdated. Another big advantage is that one can connect directly to physical router instead of Linux box as endpoint, improving performance and avoid the need for a server as endpoint at the home institute. Also here, the firewall might needs to be opened for IP protocol 47, which is used for GRE traffic. In our case this was required, because one endpoint was a virtual machine running in OpenStack, which blocks this protocol by default. root access is required on both endpoints.

- **sshuttle** \(^7\) It is a python script layer over ssh to dynamically create port forwarding to relay all TCP and (optionally) DNS traffic over ssh tunnels. Also here, we appreciated the very easy installation. In the firewalls, the ssh port is usually already open, so no changes needed. The underlying protocol is the established and well known ssh program and protocol. A slight disadvantage is that some installation is required both on server and client. However, the installation on the client will be done by sshuttle itself when the tunnel is being opened. This prohibits closing the ssh access for the user running the tunnel to a bare minimum and requires more thoughts on securing the tunnel, as for establishing the tunnel, secrets must be exchanged. To keep the right secrets hidden from others is a little bit involved. We’ll highlight a few points further down.

4.3. Performance for uploading with secure copy ‘scp’
Although not commonly used for data transfers, the ssh protocol can provide a good idea of the relative performances of the different tunnel solutions and estimate the overhead they incur.

The results for the four different connections are presented in Fig. 3. Direct connection means that data was transferred just by a direct scp connection between the two Linux servers. For the other three connections scp was going through tunnels opened by GRE, OpenVPN or sshuttle. GRE provides a very high performing connection with close to no overhead compared to the direct connection. sshuttle has a significant overhead for small latencies but reaches a similar

---

\(^5\) https://openvpn.net/index.php/open-source/overview.html

\(^6\) https://en.wikipedia.org/wiki/Generic_Routing_Encapsulation

\(^7\) https://github.com/apenwarr/sshuttle
Figure 3. In our setup, the GRE performs nearly at native speed, labelled at “direct connection”. The other solutions studied in this paper fall behind, “ssshuttle” only for small latencies, “OpenVPN” at small and at large latencies. For intermediate latencies typical for connections within a country all solutions reach similar performances.

performance for latencies above 10ms than the direct connection. OpenVPN falls behind for small and large latencies but keeps up well with the other solutions for latencies around 10-30ms.

4.4. Performance for downloading with davix-get
As mentioned above, the davix-get is an open-source software provided by CERN. This software is closer to what’s in use by the experiments. Future versions of the jobs will use this as one of the many solution provided. Future versions of our applications will likely use this software as one of their options to upload and download the data the application requires and produces. For downloading, we could not establish reliably a GRE tunnel with a decent performance. Our network card required far from optimal settings for GRE to function, so we decided to omit these results in this study. Here, the OpenVPN solution falls behind by an order of magnitude in performance compared to the other two solutions, the direct connection and the sssshuttle solution. Both provide in this setup a very similar performance. davix-get out-performs scp by far and provides a performance that is almost independent of the network latency up to latencies around 50ms indicating that it provides its own receiving buffer of about 5MB.

4.5. Security Concerns
The three solutions provide different levels of security. Most of the concerns are about intruders faking a new virtual machine to connect to the endpoint at the local university and gaining access to the university’s network and research network with its somewhat loosened security protocols based on mutual trust.

• **OpenVPN** is an established product and protocol which uses server and client certificates. It does not provide encryption, which would likely influence its performance negatively. private IP addresses are certainly possible, which can provide additional security.
Figure 4. Using a different protocol for downloading data boosts the performance of “sshuttle” to native speed. “OpenVPN” falls behind by an order of magnitude in Bandwidth. Due to a known bug in a network driver, “GRE” was not performing optimally and we decided to leave its results out of this study.

- **GRE** is an established product and protocol which also does not provide encryption. It creates point to point connections where both sides need each other’s IP addresses of the other end during setup, private IP addresses are possible, this is one reason why GRE tunnels have been designed. Since both ends need the other IP address, it is hard to fake a new virtual machine wanting to connect. This on the other hand can also make it cumbersome to setup a connection automatically and programmatic as the clients IP address needs to be known at setup. Transfering this information across might cause a delay.

- **sshuttle** is based on an established product and protocol (ssh), which encrypts the traffic. This makes it a priori a very secure solution. However, a ssh key needs to be transferred either into the virtual machine or from the virtual machine back to the server to establish this tunnel. This might cause security concerns on campus. There are solutions which mitigate this problem, which are briefly mentioned below.

5. First Experience in Production Environment
The intrinsic security of the sshuttle solution together with the relatively easy installation of the sshuttle software pleased us, so we decided to test this solution further in a production environment.

5.1. Authentication
Before a connection can be established, one ssh key must be transferred securely between the two endpoints of the tunnel. The keys for the user who establishes the tunnel can either be generated at the home institute and transferred into the virtual machine or generated locally on the virtual machine during boot and transfer back to the home institute.

For establishing a password-less ssh connection, ssh keys are used. Two keys exist, a private
and a public key. The public key can be shared freely, but protecting the private key is a must. Together they provide access via the tunnel without the need for a password.

If the keys are generated at the home institute both of them need to be shipped out together with the virtual machine. They will be valid for at least the lifetime of the virtual machine, longer if no automated cleanup is done. If they are intercepted, they can provide full access to the user running the tunnel back to the server at the home institute. This was regarded as too dangerous and discarded.

Instead the user keys are generated on the virtual machine during boot-up. For opening the ssh tunnel, only the public key needs to be transferred back to the home institute. The private key stays on the virtual machine and is never exposed.

The public key from the user on the virtual machine is transferred back to the home institute via so called job ads within the condor job scheduler. These are normally used to report back things like memory and CPU usage, but can be configured to send arbitrary data back. This data can either be static, so never changes or dynamic, like CPU time used.

This procedure requires a little bit more care in establishing the ssh tunnel because the public ssh key for the virtual machine connecting to the home institute must for sure be installed on the server when the virtual machine initiates the connection. We found that generous timeouts can help in developing the framework and using a limited number of automated retries in production make this system very stable in production.

Preventing intruders from pretending their machine is part of the cluster can be done in our case by using GSI authentication to protect the communication between all condor daemons.

The current list of public ssh keys in use is inquired regularly from HTCondor and added to a pre-defined list of keys on the ssh tunnel server. To be able to do this, HTCondor was installed on the tunnel server and configured to connect to the condor master running on another host in the same cluster.

The bottom green line in Fig. 5 shows the CPU load on the ssh tunnel server when virtual machines are connected and doubled to 10 virtual machines in the middle of the plot. The CPU load correlates with the number of transferred bytes shown in the upper plot, red for received bytes and blue for transmitted bytes. The overall CPU load however seems not to vary linearly with the number of connected tunnels, the average load over the first 8 hours in this plot is around 1.7, and rises to 2.4 for the last 8 hours in this plot.

To reduce the overall transferred volume, a squid proxy can be installed on this cloud, possibly also utilizing the tunnel back to the home institute.

As mentioned earlier, this cloud had frequent disconnections of the network. In principle this can also affect the connection through the tunnel. To circumvent lost connections, each virtual machine checks regularly if the tunnel is still functioning and restarts the tunnel if needed. Fig. 6 shows the restarts needed on 10 virtual machines which ran for about one week. There is no clear pattern visible, where for example at a certain time all 10 connections are lost. The reconnects are spread in a random fashion, making the clear identification of the reason more difficult.

6. Conclusion

All three software products for tunneling provide viable solutions for solving cloud connectivity problems, and can be implemented on short time scales. OpenVPN typically provides the worse performance, but can be a good and convenient choice on nearby clouds. sshuttle gives a competitive performance to a direct connection except for very close-by clouds where it falls behind. GRE performs very well, but can require modifications to firewalls. In our setup, the network card prevented us operating GRE with good performance for downloads, further investigation is required. davix clearly outperforms secure copy, however tuning network parameters in the Linux kernel likely mitigate the effect. sshuttle has proven to be a reliable
Figure 5. A clear correlation of the CPU load (green, bottom plot) on the ssh tunnel server and the transferred bytes (red for received bytes and blue for transmitted bytes, upper plot) through the tunnel. At the beginning of the time shown in this plot, 5 virtual machines were connected which was doubled to 10 in the middle of the plot.

Figure 6. The yellow bands show the lifetime of the virtual machines, the black bars indicate the times when the tunnel required a re-start. There’s no clear pattern visible, so the root cause for these frequent reconnects is still unclear.

solution in small scale production with up to 50 virtual machines and we are in the process of scaling that to larger sizes.

References
[1] LHCOPN-LHCONE meeting in Taipei, Taiwan https://indico.cern.ch/event/461511/
[2] LHCOPN-LHCONE meeting in Helsinki, Finland https://indico.cern.ch/event/527372/