PROCEEDINGS OF SCIENCE

A possible solution for HEP processing on network secluded Computing Nodes

Mirko Mariotti[1]
INFN
Sezione di Perugia, Via Alessandro Pascoli 23c, 06123 Perugia (ITALY)
E-mail: mirko.mariotti@unipg.it

Daniele Spiga
INFN
Sezione di Perugia, Via Alessandro Pascoli 23c, 06123 Perugia (ITALY)
E-mail: daniele.spiga@pg.infn.it

Tommaso Boccali
INFN
Sezione di Pisa,Largo Bruno Pontecorvo 3, 56124 Pisa (ITALY)
E-mail: tommaso.boccali@pi.infn.it

The computing needs of LHC experiments in the next decades (the so-called High Luminosity LHCm HL-LHC) are expected to increase substantially, due to the concurrent increases in the accelerator luminosity, in the selection rates and in the detectors' complexity. Many Funding Agencies are aiming to a consolidation of the national LHC computing infrastructures, via a merge with other large scale computing facilites such as HPC and Cloud centers. The LHC Experiments have started long ago tests and production activities on such centers, with intermittent success. The biggest obstacle comes from their typical stricter network policies with respect to our standard centers, which do not allow an easy merge with the distributed LHC computing infrastructure. A possible solution for such centers is presented here, able to satisfy three main goals: be user deployable, be a catch-all solution for all the protocols and services, and be transparent to the experiment software stack. It is based on the integration of existing tools like `tsocks`, `tunsocks`, `openconnect`, `cvmfsexec` and `singularity`. We present results from an early experimentation, which positively show how the solution is indeed usable. Large scale testing on thousands of nodes is the next step in our agenda.

[1] Speaker and corresponding author

PoS(ISGC2021)002

PROCEEDINGS OF SCIENCE

International Symposium on Grids & Clouds 2020, ISGC2021
22-26 March, 2021
Academia Sinica, Taipei, Taiwan (online)

1. Introduction and the definition of the problem

In order to cope with the future needs of LHC computing during the High Luminosity (HL-LHC) phase, the scientific communities are developing a series of new approaches and solutions spanning several contexts, from the adoption of new optimized data formats up to the integration of possibly any type of computing infrastructures into the well know WLCG Grid resources. Many Funding Agencies are aiming to a consolidation of the national computing capacity exploiting also HPC centers; while Grid resources are homogeneous in many respects such as the used Grid middleware, architecture, setups of the centers etc, HPC installations can be very different because HPC systems are highly not standard facilities, designed and developed having in mind use cases largely different from High Energy Physics ones, making the integration process not a trivial task.

From a technical perspective HPC centers differ on a variety of specialized hardware setups, storage setups and policy wise strict usage rules apply for security reasons. In turn, this means that in order to integrate HPC resources into the highly automatized workload management setups of the CMS computing system, several challenges need to be addressed in order to possibly accommodate systems with less common operating systems, lower memory availability per core, absence of local scratch disk space on compute nodes and limited if not absent outbound network connectivity.

The latter represents the biggest obstacle with respect to our standard facilities, since only a fraction of HPC centers allow for outgoing connections from the computing nodes. This affects all the steps typically performed by CMS jobs that expect to access external services at run time to perform the following steps:

1. The communication with the central CMS queue (the HTCondor Global Pool) in order to deliver the actual payloads;
2. The access to data taking conditions (from a Oracle DB accessible sia a hierarchy of squid servers);
3. The access to the experiment software;

4. The access to remote input data and the stageout of produced output data.

1.1 Known solutions

Several R&D activities are currently ongoing in CMS in order to mitigate the limitations of the limited/no connection from worker nodes. The solution under development and investigation are:

- A shared FS based communication path for HTCondor daemons, taking the place of network socket communications. This is also known as the HTCondor-splitstarter [1] based solution, which relies on a shared file system which is mounted on the execute nodes and on the login machines, made available via `sshfs`. This is currently the solution CMS is investigating in a few centers such as BSC in Spain and Theta in US.
- A user-level network based solution, for example relying on Squid proxies that, if properly configured, can provide TCP tunneling capabilities. A client can connect to them via a HTTP CONNECT call, and instruct them to open a TCP connection to route traffic to an arbitrary other host and port. The HTTP CONNECT tunnel is only for TCP connections (no UDP) hence while still useful for tunneling specific traffic, it does not solve all the problems.
- Virtual Private Network (VPN) based solutions. In local setups where compute nodes allow for user namespaces, the setup can run entirely on user space with no root permissions required. This solution is not suitable if namespaces are not enabled.

In this paper we describe a solution based on `tsocks` [2], a networking tool designed to trap system calls and allow rerouting TCP+UDP connections via multiple means. The motivations for this choices are to provide a solution that:

- can be deployed (without hacking the system) by a standard user;
- covers all the possible connections (UDP, TCP) and protocols (XrootD, SRM, HTTP(S), SSH, …);
- is not intrusive from the Experiment software perspectives (no recompilation, no changes of configuration, no need to ask for special pilots or workflows);
- can be the pillar for a universal edge service working below the application layer.

PoS(ISGC2021)002

2. The implementation

An universal tool, working below the application layer has been developed as an edge service to overcome the network limitation in a typical HPC node, as described above.

The overall diagram of the project is illustrated in Figure 1. The only requirement from a network point of view is that the nodes trust and can reach, via TCP, a specific endpoint that will serve as a fan-out service. All the other routing configurations of the nodes are irrelevant except for this point to point connection, called VPN in the following. This endpoint is the first component of the proposed implementation. Its consumer, the VPN client, is the second and has the purpose of creating a proxy socket within the node and connecting it to the VPN. The two solutions tested to create this kind of VPN will be described later; one is entirely based on ssh and the other on `openconnect` [3] and `tunsocks` [4].

The resulting socks proxy is the network entry point for the applications within the nodes using a library called libtsocks.

Finally, the `singularity` [5] containerization system and the `cvmfsexec` [6] tool allow to pack all the software and to access remote cvmfs repositories respectively.

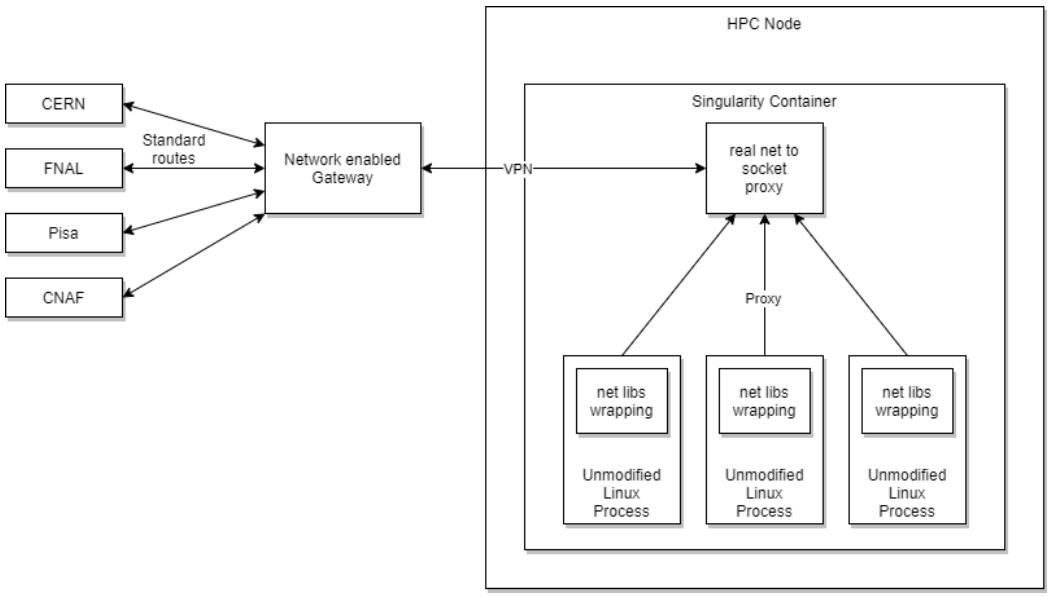

Figure 1: Overall diagram of the proposed implementation.

2.1 Tsocks

The most important tool of the proposed implementation is the net wrapping library: `tsocks`. Its purpose is to change the behaviour of the network related os calls.

PoS(ISGC2021)002

By default, they send network traffic to and from a NIC present in the system either real or a virtual one, following the kernel routing table. With the `tsocks` modification, system calls interact instead with a socks proxy. This is necessary because the creation of a virtual NIC requires superuser privileges and is not doable in this scenario; the launch of an user space application that opens a proxy endpoint can be made by a standard user instead[2].

`Tsocks` is present as part of the standard distribution is Debian derivative. In Centos based distribution, it has to be patched and recompiled from sources[3].

Its configuration is managed by a `.conf` file, and it is possible to specify which proxy use for specific routes and which are the native ones. Figure 2 shows an example of `tsocks` configuration.

`Tsocks` can be used either wrapped in a specific executable, or via the `LIBRARY_PRELOAD` mechanism of the `libtsocks` library to wrap a full shell and its children processes.

```
local = 192.168.0.0/255.255.255.0
local = 10.0.0.0/255.0.0.0
server = 192.168.0.1
server_port = 1080
path {
      reaches =
      150.0.0.0/255.255.0.0
      reaches =
      150.1.0.0:80/255.255.0.0
      server = 10.1.7.25
      server_type = 5
}
```

Figure 2: `Tsocks` *configuration example*

2.2 Proxy via SSH

The first way to create the VPN is `ssh` with the `-D` option. In this configuration, the fan-out node could be any system that is reachable via `ssh` from an HPC compute node. It could be any host, but usually a bastion host or a login node are already reachable by the center compute nodes. The `sshd` daemon on this system will act as a

[2] We are not arguing here whether standard End User agreements would still veto this type of solutions; we just want to underline how technically the solution is operating within the standard privileges of a user.

[3] We wish to thank E.Sindrilaru, F.Furano, M.Simon, P.Paparrigopoulos, L.Mascett (CERN) for their assistance in this phase.

PoS(ISGC2021)002

point of exit towards the internet and the ssh on the HPC node will create the proxy for the `tsocks` operations, as shown in figure 3.

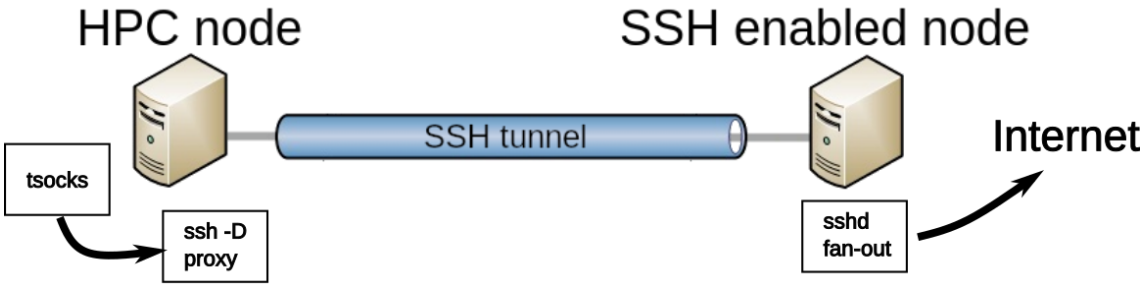

Figure 3: `ssh -d` *VPN*

`ssh -D` can also be used directly on the exit node using itself as a login machine (ssh -D localhost) and pointing the `tsocks` endpoint to its IP address (in this case also the -g option is needed) as shown in figure 4.

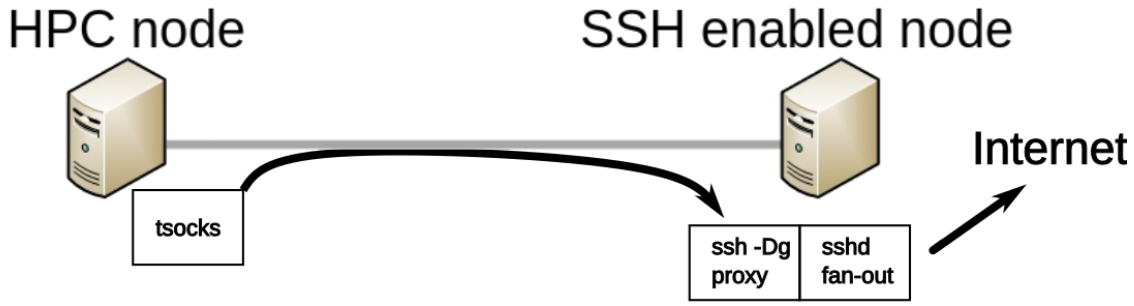

Figure 4: ssh -d localhost

2.3 Proxy via `tunsocks` and `openconnect`

The second way to build the VPN connection is using, instead of `ssh`, a proper SSL-VPN software like `openconnect`. By default it runs with administrator privileges, and creates a virtual NIC upon connection established. It also can be used with standard user privileges providing a "`tun-script`", a program to forward the network traffic without the need of a virtual interface. The tested solution uses `tunsocks` as `tun-script` for `openconnect`. `Tunsocks` creates a proxy, just like `ssh -D`, but interfacing with an SSL-VPN solution like `openconnect` as shown in figure 5.

PoS(ISGC2021)002

Figure 5: tunsocks and openconnect

This solution could also be deployed with an hardware edge system, like those from Cisco. `Openconnect` is compatible with the `Anyconnect` protocol that can be found in a wide range of routers. Figure 6 shows an example of the invocation of the `openconnect` command.

```
echo "some password" | /usr/sbin/openconnect --passwd-on-stdin
--no-dtls -u [user] --script-tun --script "/usr/bin/tunsocks -D 5555"
[edge node ip]  --servercert
pin-sha256:kgZNs8k8lbK0cI6IketuiITLcqRxpul1Xa+WD+l7fPY=
```

Figure 7: `openconnect` script example

2.4 The two approaches in comparison

The two approaches share some common traits. The edge service can be anywhere, it is not necessarily close to the HPC node; the only requirement is that it is reachable from it. Moreover, there can be more than one such edge systems, for example in DNS round robin, to improve the scalability and the reliability of the connections. `Tsocks` can also be used wrapped in another `tsocks` connection. Complex chains can be created this way.

Table 1 shows the differences between the two possibilities.

Openconnect + tunsock	ssh
The installation of software is needed on both on the edge node and on the compute	It is probably already installed and available on the edge node and on HPC node

PoS(ISGC2021)002

node	
The edge service needs to be configured with routes and users.	There is no configuration needed.
A native VPN solution is expected to scale better when handling many connection than a remote access software like SSH	Not scaling as well as a dedicated VPN
The edge service can be configured in several ways to have a fine-grained control of the network connections of the users	`ssh` does not give the possibility of selecting network routes. If this is a necessity some other third-party tools has to be used
Fine-grained control of accounts using the edge service.	The solution is available only for the user that logins to the HPC node. There is no support for multiple users if the edge service is the login node. If the edge service is a dedicated system, the access control of users rely on `ssh` and is less flexible than the `ocserver` one.
Direct in hardware support (`anyconnect` devices) is possible	

Table 1: Differences between `ssh` *and* `openconnect`

2.5 CVMFSexec

`cvmfsexec` is a tool for mounting cvmfs as an unprivileged user. The CernVM File System provides a universal way to distribute software among High Energy Physics (HEP) sites. While Cvmfs require a privileged user to be used, with `cvmfsexec` the same functionality can be brought to standard users file spaces.

2.6 Singularity

`Singularity` is a computer program that performs operating-system-level virtualization also known as containerization. It brings containers and reproducibility to scientific computing and the (HPC) world. Differently from other containerization

PoS(ISGC2021)002

solutions, it is present in many HPC clusters because it does not need superuser privileges or daemons to run. For this reason and to pack all the proposed configurations in a single object a singularity recipe has been written. Starting from the recipe a container image can be built, deployed and started on HCP nodes.

3. Tests

In order to verify that a complete HEP typical payload runs smoothly with the proposed solution, tests have been run at CINECA (the HPC/PRACE node for Italy). CINECA hosts a number of HPC most of which are suitable for the test. The chosen payload has been the standard CMS grid job. CMS is is one of the LHC experiments and its job processing over distributed computing needs in many areas the possibility to connect from the computing node to remote central services such as Condor Startd to Schedd connections, CVMFS for software releases, Xrootd/WebDAV/SRM for input output, access to VOMSes, access to CMS central services (Frontier, WMAgent, etc). Outgoing connections are multiprotocol and in a standard edge service scenario would need multiple proxies or connectors while here we try to encapsulate all of them with a single user level tunnel.

Before trying a full run the single services have been tested. A brief report of these tests are reported in table 2.

Access to the experiment software	Using the `cvmfsexec` script, the mount and unmount of remote software repositories have been tested. Also the execution of code works smoothly
Data transfers	Transfer of data have been tested using SRM (`srmcp`) and Xrootd (`xrdcp`), in both directions
HTcondor operations	Using the `manual_glidein_startup` the daemon communications have been tested to work properly

Table 2: Single services tested

A more extensive test has been performed on CINECA Galileo, an HPC cluster with no CVMFS, no external networking, no suid/sudo rights. A set of analysis crab jobs have been sent through the standard CMS tools to workers prepared with all the proposed setup. The result is shown in Figure 8 and shows a 100% success rate.

PoS(ISGC2021)002

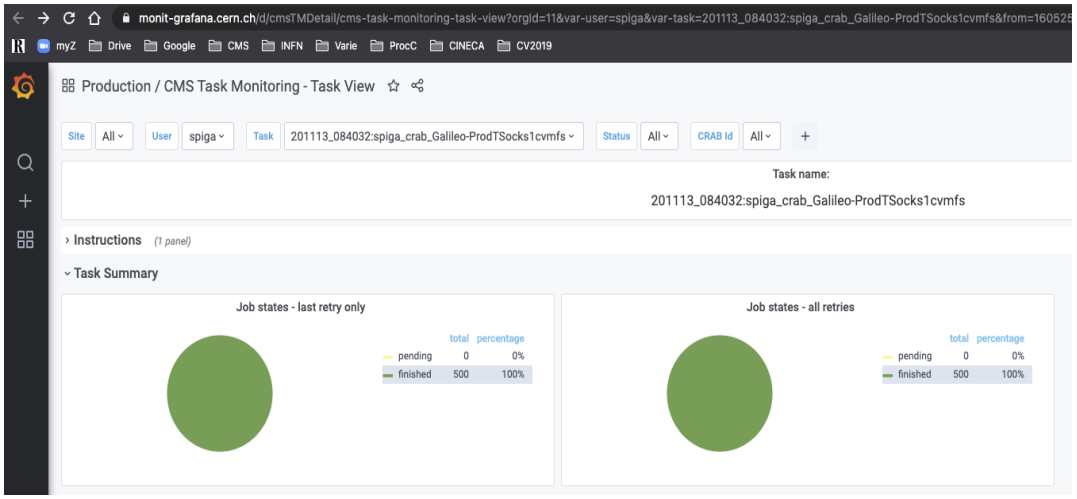

Figure 8: Success rate of encapsulated CMS jobs

A last and even more extreme test has been conducted in CINECA Marconi 100 [7]. The cluster has a Power9 architecture and just local connectivity on nodes (not even a default gateway). Nodes are reachable via SLURM though, and can login back to the login node. Using the `ssh` setup previously described, it has been possible to run the standard validation test of CMS payload, even using the V100 GPU present on nodes, as shown in Fig 9.

```
Singularity> 11634.502_TTbar_14TeV+2021_Patatrack_PixelOnlyGPU+TTbar_14TeV_TuneCP5
_GenSim+Digi+Reco+HARVEST Step0-PASSED Step1-PASSED Step2-PASSED Step3-PASSED  - t
ime date Wed Oct 14 11:29:31 2020-date Wed Oct 14 11:01:24 2020; exit: 0 0 0 0
1 1 1 1 tests passed, 0 0 0 0 failed
```

Figure 9: Standard validation test result on Marconi 100

Larger Scale tests conducted after the ISGC Conference, using the more controlled environment of CMS Monte Carlo official processing toolchain, have shown a non measurable degradation of CU efficiency when running within `tsocks` when compared with a standard routing solution. While more measurements are needed on this front, the preliminary measurements are very encouraging.

4. Conclusions

A user deployable solution based on the integration of existing tools like `tsocks`, `tunsocks`, `openconnect`, `cvmfsexec` and `singularity` has been presented together with end-to-end tests performed on a real setup, showing that the solution can actually be used in "restricted network" situations to execute CMS analysis jobs. The presented solution is protocol and service independent and it can allow to

PoS(ISGC2021)002

build a possible “universal edge service” not limited to a single application. Although a single tunnelling host could suffice for a large cluster to execute non data intensive processing (Monte Carlo generation workflows), clearly a wider deployment could scale to allow for data intensive processing.

The authors are aware a similar effort is being carried on by Ben Tovar at Notre Dame, based on using linux namespaces and a user level VPN setup and `openconnect+ocproxy` [8] `+tsocks` as a fallback when namespaces are not enabled.

References

[1] https://indico.cern.ch/event/936993/contributions/4022104/

[2] http://tsocks.sourceforge.net/

[3] http://www.infradead.org/openconnect/

[4] https://github.com/russdill/tunsocks

[5] https://sylabs.io/singularity/

[6] https://github.com/cvmfs/cvmfsexec

[7] https://www.hpc.cineca.it/hardware/marconi100

[8] http://manpages.ubuntu.com/manpages/bionic/man1/ocproxy.1.html