Journal of Physics: Conference Series **513** (2014) 032088 doi:10.1088/1742-6596/513/3/032088

Using ssh and sshfs to virtualize Grid job submission with RCondor

I Sfiligoi and J M Dost

University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093, USA

isfiligoi@ucsd.edu

Abstract. The HTCondor based glideinWMS has become the product of choice for exploiting Grid resources for many communities. Unfortunately, its default operational model expects users to log into a machine running a HTCondor schedd before being able to submit their jobs. Many users would instead prefer to use their local workstation for everything. A product that addresses this problem is RCondor, a module delivered with the HTCondor package. RCondor provides command line tools that simulate the behavior of a local HTCondor installation, while using ssh under the hood to execute commands on the remote node instead. RCondor also interfaces with sshfs, virtualizing access to remote files, thus giving the user the impression of a truly local HTCondor installation. This paper presents a detailed description of RCondor, as well as comparing it to the other methods currently available for accessing remote HTCondor schedds.

1. Introduction

In recent years, glideinWMS[1] has become the product of choice for exploiting Grid resources for many communities[2], especially on the Open Science Grid[3]. Its separation of resource provisioning from user job scheduling has proven to drastically simplify the handling of heterogeneous environments, as it is typically the case in the Grid.

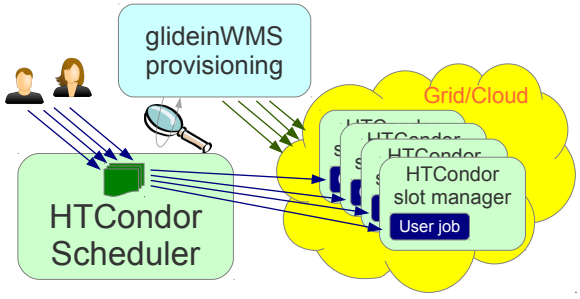


Figure 1. The glideinWMS layers

The glideinWMS relies on HTCondor (previously known as Condor)[4] for the scheduling of jobs to provisioned resources, as shown in figure 1, and this is the only layer that final users directly interact with. As explained in section 2, however, the HTCondor architecture assumes that all participants run a persistent service, which makes it impractical for most mobile users. Users thus

typically rely on a third party to host the HTCondor submission infrastructure for them, requiring them to first log into the remote node before doing any real work. The service used for remote access is typically *ssh* (Secure Shell)[5], with the *gsissh*[6] variant allowing for authentication with Grid credentials.

However, working in a remote environment can be annoying. The software selection is likely very different. Network latencies can make interactive use very sluggish. Graphic User Interfaces (GUIs) are rarely available. In summary, users usually prefer to work locally, as much as possible.

Hence the idea: let's keep ssh but hide it from the users. Most ssh clients have support for scripting and *sshfs*[7] can be used to provide virtalized access to the remote files. So we created the RCondor package, now part of the standard HTCondor distribution, that provides commands that are equivalent to the HTCondor ones, but use ssh under the hood to transparently connect to the hosting node on demand. From the user point of view, thus, it as is if HTCondor was really local. More details are available in section 3.

Ours is of course not the first attempt to provide remote access to HTCondor services. An overview of the alternative systems is provided in section 4, where we also make our case that our approach is the most useful one for a large variety of users.

2. The HTCondor architecture and the typical deployment scenario

The HTCondor architecture is based around a set of services in peer-to-peer (P2P) configuration[8], as shown in figure 2. Matched resources consumer and resources provider services talk to each on a regular basis. If any one service stops responding for an extended period of time, the default threshold being 20 minutes, that service is considered dead by the other party, and any work related to that pairing will be aborted. It is thus paramount that all services maintain a high level of network availability.

The service that the users interact with is called the *schedd*, or Scheduling Daemon. The schedd is a persistent service, which accepts jobs from the users and then coordinates with the other HTCondor services for them to finish in the shortest amount of time. Its operational model also calls for job submissions to be local to the service itself, a large reason being the need for the access to the storage area where the input files required by the jobs are read from, and the output files are stored in. An overview is available in figure 3. Please note that there must be at least one schedd service in the system, but for scalability reasons there may be several of them.

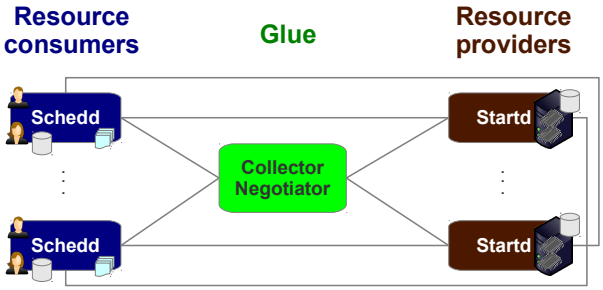


Figure 2. The HTCondor architecture

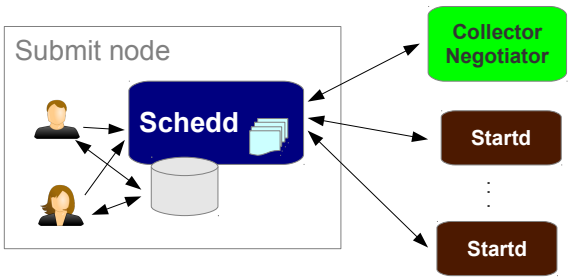


Figure 3. The submission node

This implies that there is no such thing as a *light HTCondor client*. The submission infrastructure is effectively a server setup, requiring both high uptime and incoming network connectivity. Neither functionality is suitable for the workstations of users who tend to be quite mobile nowadays, e.g. their laptops and tablets. As a consequence, users typically do not run HTCondor tools on their local machines, but rely on a third-party service provider to run it for them, using remote access tools, typically *ssh*, to bridge the gap, as shown in figure 4.

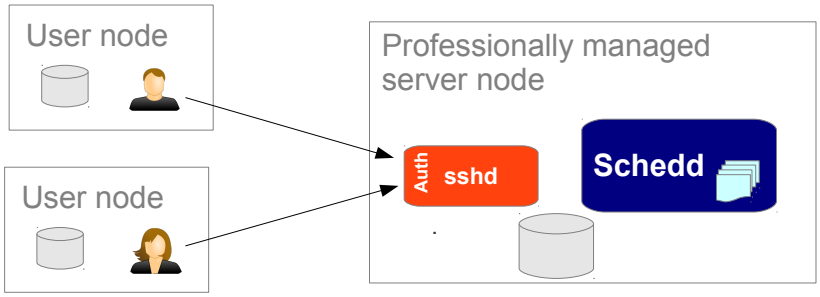


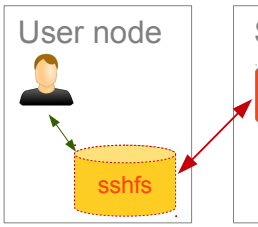
Figure 4. Typical HTCondor deployment

As mentioned in the introduction, this setup comes at the expense of ease of use for the users of the system.

3. The RCondor package

The work presented in this paper stemmed from the desire to bring local look-and-feel to the users of HTCondor, while still keeping the HTCondor schedd on a remote, professionally managed server. Keeping the existing authentication and authorization mechanism in place, for ease of maintenance, was also high on the desiderata list.

The resulting solution built on the realization that the ssh mechanism is used by most deployments is actually very powerful and flexible, on top of being very mature and stable. Most ssh clients also have good support for scripting. Finally, ssh can be used to provide virtualized access to remote file systems as well, though the use of the sshfs Linux package, as shown in figure 5.



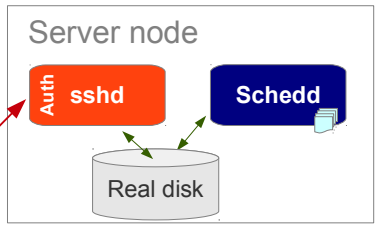


Figure 5. Virtualizing file access with sshfs

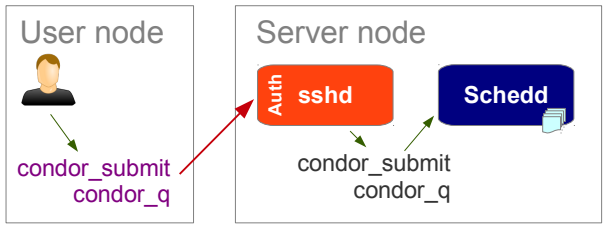


Figure 6. RCondor provides wrappers around the standard HTCondor command line tools

The RCondor package, now standard part of the HTCondor distribution, provides a set of command line tools that mimic the behavior of standard HTCondor tools, but require no local HTCondor daemons to be installed and running. The provided tools are effectively just wrappers around the real HTCondor commands that get executed on the server node through ssh, as schematized in figure 6. They are also aware of sshfs being used, thus converting absolute paths when needed. Integrated sshfs mounting and unmounting tools are also provided.

The configuration of the RCondor package is pretty straightforward. A typical user just needs to provide the server node it wants to communicate with. However, advanced users can optionally further customize their experience, from specifying the desired sshfs mountpoint to customizing the invoked ssh client.

4. Related work

There are three major alternative options for users who want to use HTCondor from their mobile workstations.

The HTCondor schedd itself does support remote submission, if properly configured. The user could use HTCondor in this mode using only a true HTCondor-provided client, as shown in figure 7. However, using this operational mode has four severe drawbacks:

a) The HTCondor clients operating in remote mode do not provide the same level of monitoring as when operating in local mode. In particular, there is no job log file, which is essential for some add-on tools, like DAGMan.
b) Users must explicitly poll the schedd to get updates on the job progression and to fetch any produced output files. This can put a substantial load on the schedd, impacting its scalability limits.
c) The authentication and authorization configuration is HTCondor specific, yet it has to be synchronized with the system security options of the hosting server. This makes it hard to maintain and is error prone. It may also violate policy rules of some sites.
d) The data staging happens independently for each job submission. This can be a problem for jobs that share a significant subset of the input files.

The second option is the so-called HTCondor-C mode. Here, the user installs and runs a local schedd, but this schedd does not talk directly to the daemons managing the CPUs, i.e. the startds. Instead, the local schedd just accepts the jobs from the user, and forwards them to the schedd on the server node, as shown in figure 8. The protocol between the local workstation and the server schedd is essentially the same, so problems (a) and (b) above remain. However, the local schedd now coordinates that communication, making it both more scalable and more user friendly. The user also gets full HTCondor functionality, e.g. it can use DAGMan. It does, however, introduce a new set of problems:

e) The syntax for submitting jobs is completely different, and arguably much more complex, than the one used for "vanilla HTCondor" jobs.
f) The user now runs a daemon process, which must be properly configured. With proper configuration it is a pretty safe process, but with only minor misconfiguration it can allow for remote execution of arbitrary code.

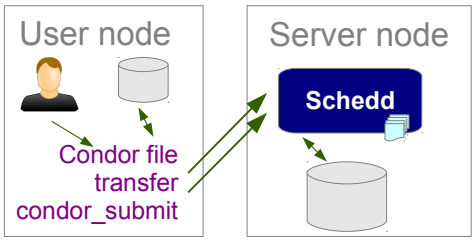


Figure 7. HTCondor remote submission

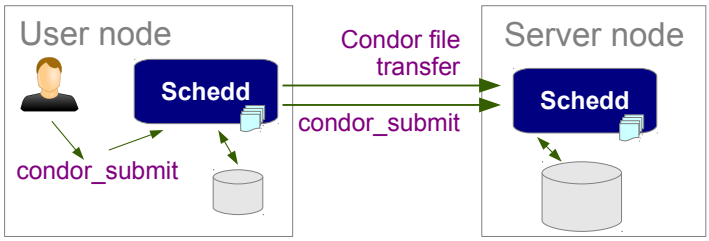


Figure 8. HTCondor-C mode

The third option is provided by BOSCO[9], which is basically a helper wrapper around HTCondor-G. This setup is similar to the Condor-C scenario, with the user running a local schedd. However, the local schedd uses ssh to talk to the remote schedd, as shown in figure 9. This solves problem (c) above. Problems (d), (e) and (f) are still present, though.

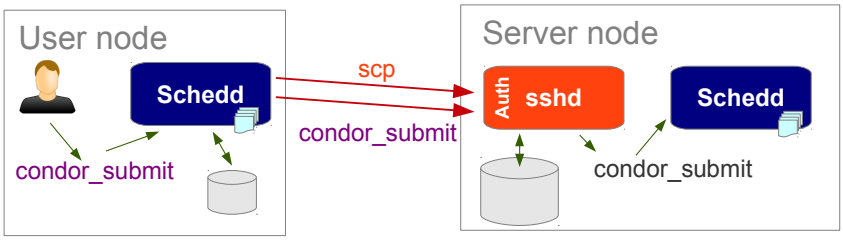


Figure 9. HTCondor setup under BOSCO

We believe that our approach is to be preferred over the other three methods, as it provides a true client implementation with all the expected HTCondor functionality. The only use case our approach does not cover is submitting jobs while off the network; in this case, a local schedd who accepts the

request immediately would be needed. It is our belief that for most users the benefits of our approach outweigh the lack of this functionality.

5. Future work

There has been interesting work going on in the CMS community about using HTCondor over ssh[10]. One particularly interesting aspect of their work was the optimization of the user experience when dealing with high network latencies and unreliable ssh clients. We plan on incorporating their methods into the RCondor package in the near future.

HTCondor has recently added the support for python bindings for communicating with the HTCondor daemons. We thus want to explore how difficult would it be to provide python bindings that operate on top of ssh, and add this capability to the RCondor package, if feasible.

6. Conclusions

Most Grid computing nowadays is based on pilot infrastructures, with glideinWMS being the most used one in the Open Science Grid. However, since glideinWMS uses HTCondor as the user facing interface, it requires a server installation on the submission side, making it is very hard for users to interface with it from mobile workstations.

The solution proposed in this paper is to use the RCondor package to seamlessly communicate to a remote glideinWMS installation over an ssh tunnel. The users thus do not need to run any server software in their local environment, making it easy to deploy and suitable for mobile workstations.

The RCondor package is an optional component distributed with all recent versions of HTCondor.

Acknowledgements

This work was partially sponsored by the US National Science Foundation (NSF) under Grants No. PHY-0612805, PHY-1148698 and PHY-1120138.

References

[1] Sfiligoi I, Bradley D C, Holzman B, Mhashilkar P, Padhi S and Würthwein F 2009 *Comp. Sci. and Info. Eng., 2009 WRI World Cong. on* **2** 428-432 doi:10.1109/CSIE.2009.950

[2] Sfiligoi I, Würthwein F, Dost J M, MacNeill I, Holzman B and Mhashilkar P 2011 Reducing the human cost of grid computing with glideinWMS *Proc. Cloud Computing 2011* (Rome, Italy) 217-221 ISBN 978-1-61208-153-3 http://www.thinkmind.org/index.php?view=article& articleid=cloud_computing_2011_8_40_20068

[3] Pordes R et al. 2007 *J. Phys.: Conf. Ser.* **78** 012057 doi:10.1088/1742-6596/78/1/012057

[4] Thain D, Tannenbaum T and Livny M 2005 *Concurrency and Computation: Practice and Experience* **17 2-4** 323-356 doi:10.1002/cpe.938

[5] Ylonen T et al. 2006 The secure shell (SSH) transport layer protocol *IETF RFC* **4253** http://www.ietf.org/rfc/rfc4253.txt

[6] Butler R et al. 2000 A national-scale authentication infrastructure *Computer* **33 12** 60-66 doi:10.1109/2.889094

[7] Hoskins M 2006 SSHFS: super easy file access over ssh *Linux Journal* **8904** http://www. linuxjournal.com/article/8904

[8] Bradley D, Sfiligoi I, Padhi S, Frey J and Tannenbaum T 2010 *J. Phys.: Conf. Ser.* **219** 062036 doi:10.1088/1742-6596/219/6/062036

[9] Weitzel et al. 2013 Accessing opportunistic resources with Bosco *Int. Conf. on Computing in High Energy and Nuclear Physics 2013* Amsterdam, Netherlands

[10] Belforte S et al. 2013 Using ssh as portal - The CMS CRAB over glideinWMS experience *Int. Conf. on Computing in High Energy and Nuclear Physics 2013* Amsterdam, Netherlands