SNAPWebD and SNAPSync: A Web Desktop and Transparent *sync* of NFS and Standalone System Logical Volumes

Anupama Potluri[1], Krishna Vutukuri[1], and Garvit Sharma[2]

[1] School of Computer and Information Sciences, University of Hyderabad Hyderabad, India
[2] The LNM Institute of Information Technology, Jaipur, India
apcs@uohyd.ernet.in, {sskrishnavutukuri,garvits45}@gmail.com

Abstract. The web browser is the most ubiquitous software today on all kinds of devices including PCs, tablets and mobile phones. Access to high bandwidth network services has lead to the development of cloud computing where services such as applications or desktops can be accessed from remote servers through the Internet. In this paper, we present our work in developing a Web Desktop for the typical environments found in many corporates and educational institutions. These environments consist of Network File System (NFS) mounted home directories of users authenticated using Lightweight Directory Access Protocol (LDAP). *SNAPWebD*, our web desktop, which is a free and open source software, will allow users of such organizations to access their home directories and remote applications through the web browser. *SNAPSync* is the software that allows transparent syncing of the home directory from the cloud to local or NFS volumes of a user and vice versa. Together, these two applications allow a user to access their home directory from the cloud using any available device and also sync the changes transparently to local home systems. This keeps the information consistent across multiple systems without user intervention while allowing anytime, anywhere access through the web desktop.

1 Introduction

The Web Browser is an ubiquitous piece of software on almost all devices of communication today including PCs, Laptops, Tablets, Mobile phones etc.. This, combined with the availability of high bandwidth network access using 3G/4G mobile networks, has spurred the growth of *cloud computing*. Cloud computing is transparent access to various services in the network anywhere, anytime. One of the services that is gaining a lot of momentum is a desktop environment or an operating system through the browser called Web Desktop [1] and WebOS respectively [2–5]. This extends independence from hardware and specific operating systems for users and helps portability. There has also been development of software that keeps some of the files synchronized between the local and the

R. Natarajan (Ed.): ICDCIT 2014, LNCS 8337, pp. 105–110, 2014.

cloud versions such as *UbuntuOne* [6], *4sync* [7], *eyeSync* [4] and *GoogleDrive* [8].

In this paper, we present our work on providing a Web Desktop that is extremely lightweight and involves no installation of any software on the client system. It allows access to Lightweight Directory Access Protocol (LDAP) authenticated user accounts hosted on Network File System (NFS) logical volumes. We believe that this will be ideal for small offices and educational institutions. The IT challenge in implementing this software in an organization is minimal. While the web desktop allows access to users of their home accounts from any system at any time, typically, most users also have a laptop that they use for continuing the work at home rather than directly on the cloud. It is extremely useful to have all versions of the home directories synchronized with each other. Towards this end, we developed *SNAPSync*, that uses *rsync* as a library to synchronize the home directories. Many of the existing applications that allow synchronization only synchronize those files that exist in special directories/folders on the local system and not entire home directories.

The rest of the paper is organized as follows: the architecture and implementation of *SNAPWebD* is given in Section 2. The design of *SNAPSync* is given in Section 3. We conclude with Section 4 on enhancements planned in the future for these applications.

2 *SNAPWebD*

Virtualization of a desktop has long been used in Unix environments through software such as Virtual Network Computing (VNC) [9, 10]. Recently, software has been developed to access desktops through the web browser giving platform and OS independence. These are called Web Desktops. However, none of them have the capability to directly access already existing user accounts through the web browser. Many need installation of software on client side and use protocols such as Remote Desktop Protocol (RDP) [11] which is quite complex. *SNAPWebD*[12] overcomes all of these limitations.

SNAPWebD architecture consists of the following minimum components: an LDAP server to authenticate the users, an NFS server to host the users' home directories, a web server that hosts *SNAPWebD* software and a client system that is used by the user to access the home directory via the web browser. In addition, if an organization hosts an application server from which the applications are transparently accessed, this is also supported by the web desktop. The basic architecture of *SNAPWebD* is given in Fig. 1. The web server is the only system where our software needs to be installed. All the other components of the IT infrastructure of an organization are untouched by our application. The web desktop software we have developed is also extremely lightweight in that it consists of a maximum of 2000 lines of JSP and JQuery code. Thus, the job of the IT administrator to enable the web desktop in an organization is highly simplified.

SNAPWebD works as follows: when the user types in the URL for *SNAPWebD*, the login page is displayed. The user's details are accepted and authenticated

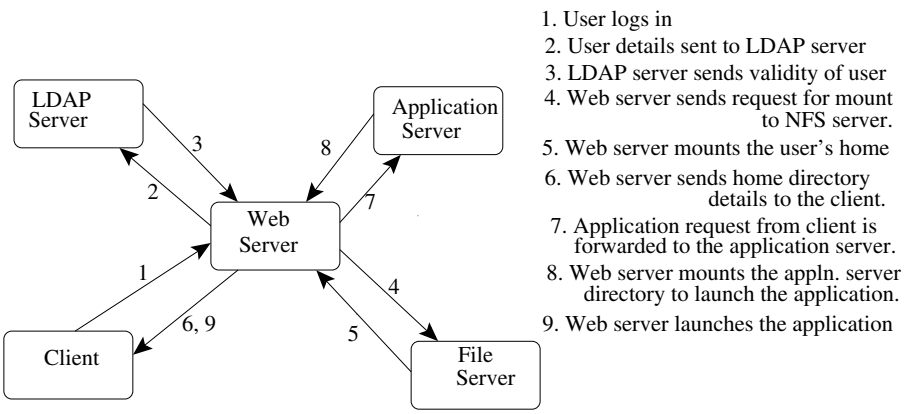

Fig. 1. Proposed Web Desktop Architecture

using the LDAP server. The LDAP server returns the home directory information for the user who has logged in. Using the home information, *SNAPWebD* software automatically and transparently mounts the corresponding NFS logical volume, if needed. Then, the contents of the home directory are displayed on the web browser. Similarly, whenever a user needs to launch an application, *SNAPWebD* software searches for the corresponding application in the local system, the web server, the NFS file server and then the NFS application server in that order. This is done so as to enhance the performance. As long as an application is available on the local system, there is no need to connect to remote servers on the network and run the application on the remote system. This helps performance in three ways: firstly, there is no network load. Secondly, there is no load on the remote server in terms of the number of running processes. Thirdly, working with a local process is much faster than with a remote process. We use the application on a web server so as not to have further network traffic between the web and file/application servers. On the other hand, if the file server volume is already mounted, accessing the application from this volume is more efficient. Only if the application is not found in any of the already mounted partitions does the software load a new partition and use it. However, this order can be customized or the institution can load balance by having different types of applications in different application servers. To run an application on the remote system we use the following Secure SHell (SSH) command: **sshpass -p "password" ssh user-name@NFS Server-IP "application-command"**. The application display (xserver display) should be exported to the client system before we run the application to open its window in the local system. This is done by using the **DISPLAY** environment variable and exporting it to the client IP address (**DISPLAY=Client-IP:0**).

We maintain a list of standard extension types for files and the applications associated with these file types in our software. We can also use a given application to open a file. For example, to open a file called "ex.pdf", we invoke the application as follows: **sshpass -p "krishna.48" ssh -o StrictHostKeyChecking=no Krishna@10.5.0.95 "DISPLAY=10.5.1.0:0;okular ex.pdf"**.

We provide other interesting features in our web desktop. One feature is a lightweight terminal which can be used as a command line interface for many of the standard actions. Another feature is the maintenance of the history of actions taken since the login, such as opening the lightweight terminal, launching an application etc.. All the commands executed in the lightweight terminal are also stored and displayed when history button in the left frame of the page is clicked. A third feature is the ability to download files from the remote file server to the local system or other locally mounted media or devices as well as to upload the files from local system to the remote file server. We also provide a search feature for the web desktop. Three types of searches are possible: firstly, we can search for an application which launches the application, if found; otherwise, it displays an error message. Secondly, we can search for files based on the name of the files using wildcard characters as is true for typical command line search. The third search feature is a rudimentary search on the content of the files. The given string is searched for in the files of the home directory recursively and all the files that contain the given string are displayed in the left frame of the page.

Web Desktop to a Specified Remote Computer: The final feature of *SNAPWebD* is that it allows access to a local user account on a given system identified by its IP address or a logical name. We assume that this system runs the NFS server software. In this case, *SNAPWebD* software authenticates the user locally without using the LDAP server. Once the user is authenticated, the home directory of the user on that system is mounted on the web server and the user can access the files of that system. A user can access an NFS account and a local account on a different standalone system simultaneously using the web desktop. Files can be transferred from one system to the other by downloading from one system to the local system and then uploading it to the other.

A table comparing the features supported by various commercial and free software web desktop products and **SNAPWebD** is given in Table 1.

3 *SNAPSync*

SNAPSync[13] uses *rsync* as a library to perform the synchronization across various systems. It runs as a daemon on systems which need to be synchronized. We need to configure the remote system(s) and directory information to which the home directory on the local system needs to be synchronized. Once this configuration is complete, the synchronization is entirely transparent to the user. *SNAPSync* differs from other synchronization products as follows: it allows for home directory to be synchronized and not only those files and directories in a specially marked directory. Further, unlike *UbuntuOne* and many other synchronization tools, it maintains a journal. This helps in two ways. Firstly, if a user has made many changes while being offline, on connection to the Internet, the overhead associated with searching for modified files is removed. Secondly, a periodic update combines multiple edits to the same file into a single synchronization event. We use $inotify$ just as *UbuntuOne* does but only to make a journal entry.

Table 1. Comparison of Features supported by SNAPWebD and Other Products

Features	**Products**			
	VNC	**eyeOS**	**OVD**	**SNAPWebD**
Display files and directories in a browser	NO	YES	YES	**YES**
Can a user get his personal desktop	YES	NO	NO	**YES**
Opening a file when the associated application does not exist in the remote system hosting the file	NO	NO	NO	**YES**
Ability to upload a file to the remote host from local system	NO	YES	YES	**YES**
Ability to download a file to the local host from the remote system	NO	YES	YES	**YES**
Lightweight terminal (execute all types of non-interactive commands)	NO	NO	NO	**YES**
Complete terminal (execute all types of commands)	YES	NO	NO	**YES**
Search for an application in user's remote system	YES	NO	NO	**YES**
Search for a file based on name	NO	NO	YES	**YES**
Search for a file based on content	NO	NO	NO	**YES**
Store the history of users' actions since login time	NO	NO	NO	**YES**
Accessing a specified standalone system (using its IP address)	NO	NO	NO	**YES**

The very first time that *SNAPSync* runs, it simply dumps the entire contents to the remote system if this directory does not exist in the remote system. After that, every time a change is done to the local system, a journal is maintained of the changes made. Periodically, the *SNAPSync* daemon checks the journal and synchronizes the modified files. The result of the operation is stored in a log file to indicate success or failure and the causes of failure. If the synchronization is a success, the corresponding entry in the journal file is deleted. If a *force sync* command is issued, it attempts to flush all the changes currently in the journal to the remote system. This is useful before shutdown of a system to ensure that all changes are synchronized in the remote system. If the user shuts down the system without this option, the journal is still retained. Hence, the changes are synchronized in the future whenever both the local and remote systems are online.

The user may also use *SNAPWebD* to modify the contents of the files in the cloud from a third party system or a mobile device. When the user returns to the home or office and goes online, we need to pull the changes from the cloud into the local system. This is the first step of the *SNAPSync* daemon. Thus, by using both *SNAPWebD* and *SNAPSync*, we can manipulate the home directory contents of users as well as synchronize them in a lightweight and transparent manner.

4 Conclusions and Future Work

We have developed a web desktop, *SNAPWebD*, that allows users to access their home directories hosted on NFS servers using the browser. While there are other web desktop applications – both free and commercial products – available, none of these work with existing infrastructure of most institutions/corporations where users' accounts are hosted on NFS servers. In *SNAPWebD*, we do not need any special software to be installed on the client system. One additional feature is the ability to access a user's account on any standalone remote system with *SNAPWebD* given its name or IP address. *SNAPSync* allows the user to pull changes to the home directory from the cloud to local systems and vice versa. This differs from other products in that there is no special directory whose contents are synchronized. A journal maintained of all changes made while the system is offline or online is used to flush the changes. A periodic synchronization leads to a more efficient utilization of bandwidth.

However, at this point of time, *SNAPWebD* and *SNAPSync* work only with Linux systems. In future, we plan to extend them to work with other operating systems as well as connect to VMs and allow access to VMs of users. We also need to explore the security considerations of *SNAPWebD*.

References

1. Web Desktop, http://en.wikipedia.org/wiki/Web_desktop
2. Silvestri, G.A.: Citrix XenDesktop 5.6 Cookbook. PACKT Publishing, Birmingham (2013)
3. Citrix: How Desktop Delivery Works, http://www.1st-computer-networks.co.uk/citrix-how-desktop-delivery-works.php
4. Norte, J.C.: Hybrid virtualization: optimal management of a companys IT resources (January 2012), http://resources.eyeos.com/ENG_virtualization_whitepaper.pdf
5. Open Virtual Desktop, http://www.ulteo.com/home/
6. Ubuntu One, https://wiki.ubuntu.com/UbuntuOne/TechnicalDetails
7. 4sync, http://en.wikipedia.org/wiki/4sync
8. Google Drive, https://support.google.com/drive/answer/2424384?hl=en&ref_topic=14942
9. VNC, http://en.wikipedia.org/wiki/Virtual_Network_Computing
10. Virtual Network Computing, http://virtuallab.tufreiberg.de/p2p/p2p/vnc/ug/howitworks.html
11. Remote desktop protocl, http://support.microsoft.com/kb/186607
12. Vutukuri, K.: SNAPWebD: a web desktop for Standalone desktop or Nfs server using APache (SNAP) web server, https://github.com/sskrishna/SNAPWebD
13. Sharma, G.: SNAPSync, https://github.com/garvitlnmiit/SNAPSync